TRANSFER LEARNING FOR MULTI-CHANNEL TIME-SERIES HUMAN ACTIVITY RECOGNITION

Dissertation
zur Erlangung des Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

FERNANDO MOYA RUEDA

Dortmund
2023

Tag der mündlichen Prüfung:	26.09.2023
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr.-Ing. Gernot A. Fink Prof. Dr.-Ing. Thomas Kirste

TRANSFER LEARNING FOR MULTI-CHANNEL TIME-SERIES HUMAN ACTIVITY RECOGNITION

FERNANDO MOYA RUEDA

2023

AKNOWLEDGMENTS

I see this thesis as a societal result, supported by the work of many people; among them,...

I am extremely grateful to Prof. Dr.-Ing. Gernot A. Fink for his supervision of my work over the last few years. Gernot! You gave me a space in your group, allowing me to freely explore research. I deeply cherish your guidance and support that helped me point in the right direction for developing this thesis. In German, the word "Doktorvater" is quite interesting to translate into Spanish; still, it is very fitting of how you have been to me.

Furthermore, I would like to thank Prof. Dr.-Ing. Thomas Kirste. Thank you for revising my thesis and participating in the doctoral committee. Your feedback was crucial for bringing some scattered ideas I had into the experimental part of my thesis. I cherish your time when discussing my work. Besides, it was crucial to me visiting the workshop you organised back in 2017, as it was one of my first steps toward this work. From your group, I would also like to thank Dr. Kristina Yordanova and Dr. Stefan Lüdtke. Thank you both for supporting my research by collaborating on interesting works in the field of human activity recognition! Thank you very much, Prof. Dr. Falk Howar, for being my mentor in the structured doctoral program! I appreciate the very positive support I received from you. I also want to thank Prof. Dr. Mario Botsch and Prof. Dr. Erich Schubert for participating in the doctoral committee.

I am very grateful to my colleague, Dr.-Ing. René Grzeszick! You have been and still are a great support over the last few years. I still do remember my first interview with you and Dr.-Ing. Leonard Rothacker, when applying for a position as scientific support at the chair. Somehow, that moment was the start of this journey. Thank you very much for the opportunity!

Dr.-Ing. Christopher Reining! Since our first discussion regarding HAR in industrial settings in the FLW building, you have become a great partner. I immensely appreciate the support and friendship!

I sincerely thank my Patrec group colleagues, Dr.-Ing. Axel Plinge, Dr.-Ing. Leonard Rothacker, Dr.-Ing. René Grzeszick, Dr.-Ing. Sebastian Sudholt, Dr.-Ing. Thorsten Wilhelm, Eugen Rusakov, Fabian Wolf, Dr.-Ing. Philipp Oberdiek, Kai Brandenbusch, Dominik Kossman, Oliver Tuselmann, Arthur Matei, Nilah Ravi Nair, and Tim Raven. It was wonderful sharing with you at the office, on academic trips, and in Wenkers. Furthermore, I thank Erik Altermann for his collaboration developing the annotation tool for human activity

recognition, and Shrutarv Awasthi for your engagement extending some experiments in the field.

This is the culmination of a long journey, where my family back in Colombia and friends have always been present. I appreciate all the support, love, and patience you all have given me in the last few years. To my brother Julian David, we have shared the world since we were born, and I am delighted to keep sharing beautiful things with you, like this triumph.

Thanks to my family in Germany, Ana María, Juliana, Juan David, María Isabel, Felipe, Olaf, Lars, Neha, Lennart, Pau, Diana, Filip, and Nicole. You become the chosen family.

To my other teachers, ...

Abuelita María del Rosario and Tía Olga Robertina, thanks for all the motivation and support. I will miss you!

And finally, to mi Madre querida, Clara Moya Rueda, thank you for everything!

NOTATION

$a, b, c, \ldots$	scalar
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	vector; all vectors are column vectors.
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	matrix or tensor
$A, B, C, \ldots$	univariate random variable
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$	multivariate random variable
$\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots$	integer value for, e.g., counts, cardinalities or dimensionalities
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	set
$\lvert\mathbf{A}\rvert$	the cardinality of set $\mathbf{A}$
$\mathbf{A}^{(i)}$	the i-th element in set $\mathbf{A}$
$\bar{\mathbf{A}}$	average value of the set $\mathbf{A}$, i.e., $\frac{1}{\lvert\mathbf{A}\rvert}\sum_{i=1}^{\lvert\mathbf{A}\rvert}\mathbf{A}^{(i)}$
$\mathbf{a}^\top$	the transpose of vector $\mathbf{a}$
$\hat{a}, \hat{\mathbf{a}}$	estimate or prediction of a scalar or vector
a_i	the i-th element of vector $\mathbf{a}$
$a_{i,j}$	the element of the matrix $\mathbf{A}$ at row i and column j
$\operatorname{diag}(\mathbf{A})$	the main diagonal of $\mathbf{A}$
$a_i^{(j)}$	the i-th element of the j-th vector in set $\mathbf{S} = \{\mathbf{a}^{(j)}\}_{j=1}^{n}$
$f(x), f(\mathbf{x})$	scalar function with scalar or vector argument; for all functions that are only defined on scalars, a vectorial argument indicates element-wise application of the function
$\mathbf{f}(\mathbf{x})$	vector function with vector argument
$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i}$	the partial derivative of $\mathbf{f}$ with respect to x_i
$\mathbb{E}[X]$	the expected value of the univariate random variable X
$\langle \mathbf{a}, \mathbf{b} \rangle$	inner product between the vectors $\mathbf{a}$ and $\mathbf{b}$
$[[\mathsf{P}]] = 1$	Probability of 1 when P using Iverson's bracket notation
$\lVert\cdot\rVert$	the Euclidean norm $\lVert\cdot\rVert_2$

$\{O\}$	frame attached to point $\{O\}$
${}^{\{O\}}_{\{J\}}\mathbf{a}_j(t)$	acceleration $\mathbf{a}$ of a point j located in frame attached to body $\{J\}$ with respect to an origin frame $\{O\}$

CONTENTS

1 INTRODUCTION

Methods of Human Activity Recognition (HAR) have been developed for the purpose of automatically classifying recordings of human movements into a set of activities. Capturing, evaluating and analysing sequential data to accurately recognise human activities is critical for many applications in pervasive and ubiquitous computing applications, e.g., in applications such as mobile- or ambient-assisted living, smart-homes, Activities of Daily Living (ADL), health support and rehabilitation, sports, automotive surveillance, and industry 4.0 [CGD+13, CFK13, BBS14, ZNY+14, FPZ16, HHP16, LLSL16, OR16, CGS18, HMSB18, HVL18, Rei21]. For example, HAR is of special interest for optimisation in those industries where manual work remains dominant.

Human Activity Recognition (HAR) takes as inputs signals from videos [FPZ16, KY18] or from multi-channel Time-Series (multi-channel Time-Series), e.g., human joint measurements from marker-based Motion Capturing System (marker-based MoCap) [Rei21], and inertial measurements from On-body Devices (OBDs) [ZNY+14, RC15, YNS+15, OR16]. This thesis focuses on HAR from multi-channel Time-Series, and will be addressed as multi-channel time-series Human Activity Recognition (M-HAR). OBDs have become relevant as they extend the potential of HAR beyond constrained or laboratory settings.

On-body Device (OBD) devices are suitable for M-HAR as activities can be tracked and performed in a natural environment [HMS16, ZXZ+17, HVL18]. Besides, these devices are not affected by occlusion and do not show human identities easily [HMSB18]. OBDs are low-power devices that are highly reliable, non-invasive and easy to use. They should assist anywhere and anytime by observing activities egocentrically [BBS14, ZNY+14, MMD17, TDF+18]. Additionally, OBD are relatively inexpensive in comparison with marker-based MoCap scenarios, which comprise several cameras, motion capture suits and expensive licensed software.

OBDs contain 3D inertial sensors, measuring derived physical quantities in three dimensions. Examples of such sensors are accelerometers and gyroscopes. OBDs might also contain sensors capturing environmental quantities such as magnetometers measuring the magnetic field, or barometers for air pressure. Besides, these devices could also contain human vital sensors such as temperature and heart rate monitors [RC15, OR16, HHP16,

TDF+18]. The latter are relevant for medical applications, e.g., in rehabilitation, preventing accidents, and supporting the elderly [BBS14, HMSB18].

Multi-channel time-series Human Activity Recognition (M-HAR) is, in general, a challenging classification task. Human activities and movements show a large variation. Humans carry out in similar manner activities that are semantically very distinctive; conversely, they carry out similar activities in many different ways. Furthermore, M-HAR datasets suffer from the class imbalance problem, where there are more samples of certain activities than others [FMHF16, OR16]. This problem strongly depends on the annotation; e.g., at MotionMiners GmbH, walking and standing are usually ignored as they are not crucial for application purposes. Moreover, there are non-standard definitions of human activities for annotation [BBS14, ZNY+14].

Based on the assumption that activities are a combination of body movements presenting specific patterns, a M-HAR system seeks to use these patterns to classify body movements with different techniques. A traditional M-HAR pipeline segments sequences, then extracts relevant hand-crafted features from the segmented sequences, and finally trains a classifier for assigning specific activity labels to the sequences [BBS14, FMHF16].

Methods based on Deep Neural Networks (DNNs) are prevalent for M-HAR. DNN learn more discriminative features of human activities in contrast to statistical pattern-recognition approaches [YNS+15, ZNY+14]. The advantage of DNNs is that they combine the learning of feature extractors and classifiers in an end-to-end approach, minimising a common loss or cost function. Different configurations of such networks have been introduced, e.g., Temporal Convolutional Neural Network (tCNN) [ZNY+14, RC15, YNS+15, MSR+17, CGS18, RRR18, YLSR18, CZY+21] and Recurrent Neural Network (RNN) [HHP16, OR16]. They also aggregate information from the entire sequence, e.g., transformers [Mah20, BV21, SK21]. Convolutional Neural Networks (CNNs) and RNNs learn the non-linear and temporal relations of basic, complex and highly dynamic human movements. They learn non-linear features directly from raw inertial data. These relations are discriminative regarding human activities, and at the same time, they are invariant to amplitude, and temporal distortions of the sensor measurements [HHP16].

The performance of DNN has not significantly increased as in other fields such as image classification or segmentation. DNN present a low sample efficiency as they learn the temporal structure from activities completely from data. Considering supervised M-HAR, scarcity of annotated M-HAR data is the primary concern [KHC10, DPBR20]. Annotated data from human behaviour is scarce and costly to obtain. The annotation process demands enormous resources. Additionally, annotation reliability varies, because they can be subject to human errors or unclear and non-elaborated annotation protocols [FMHF16, KHC10, Rei21].

Transfer Learning (TL) has been used for coping with a limited amount of annotated data, as well as, overfitting, zero-shot learning or classification of unseen human activities, and the class-imbalance problem [OMR16, AR17, GY17, CZY+21], e.g., object recognition [KSH12], object detection [RHGS15], face recognition [PVZ15], and word spotting [GSF18, WF22]. Transfer learning can alleviate the problem of scarcity of annotated data. Learnt parameters and feature representations from a certain source domain are transferred to a target domain. The target and source domains are related [PY10, GY17].

Fine-tuning is a common parameter transfer learning strategy for DNN. Filters from the convolutional layers are trained on a large source dataset. Then, these filters are taken as initialisation for an architecture to be adapted to a related target task. The Fully Connected Layers (FCs) are the only ones trained from scratch [OMR16]. However, Transfer Learning for Multi-channel Time-Series HAR (Transfer Learning for M-HAR) can be hindered by the enormous variation of recording settings. Authors use different recording rates, sensor resolutions, device positioning, or intrinsic device characteristics.

Sharing feature representations is a type of transfer learning method. Motivated by the success of semantic attributes for representing classes in the context of image or scene classification [LNH09] and document analysis [SF18, RRMF18], human activities can be likewise represented by such a collection of semantic attributes. These attributes describe semantically and coarsely human activities, e.g., "gait-cycle" and "grabbing with two hands" represent the "pushing a cart" activity [ZJC17, Rei21]. Besides, they can share a set of similar human activities, e.g., "walking" and "running" contain "gait-cycle" as a common attribute. Moreover, attributes are suitable for recognition tasks where the data is imbalanced, or training and testing sets are disjoint, e.g., zero-shot learning.

This thesis proposes a method for Transfer Learning for M-HAR using attribute representations and parameter transfer. Firstly, it introduces the search and use of attribute representation that favourably represent signal segments for recognising human activities; currently, M-HAR datasets composed of multi-channel Time-Series from OBDs and marker-based MoCaps lack human-annotated attributes. Moreover, it presents a DNN for predicting attributes, including temporal convolutions and an OBD-centred design.

In addition, a method for data-based Transfer Learning is proposed. The method takes advantage of a large human-pose dataset as a source domain. DNNs are fine-tuned using inertial measurements from OBDs. These networks will process sequences of movements from the human limbs, either from poses or inertial measurements. Furthermore, synthetic-inertial measurements will be derived from sequences of human poses either from marker-based MoCap or video-based HAR and pose-based HAR datasets. The latter will specifically use the annotations of pixel-coordinate of human poses as multi-channel Time-Series data. All these synthetic measurements will then be deployed as a source domain for transfer learning.

1.1 CONTRIBUTIONS

The methods and findings presented in this thesis have partially been published at scientific conferences in the activity recognition and pattern recognition communities. The research has been carried out in the *Pattern Recognition Group* at TU Dortmund University under the supervision of Prof. Dr.-Ing. Gernot A. Fink. This thesis was also supported by Deutsche Forschungsgemeinschaft (DFG) in the context of the project Fi799/10-2 *Transfer Learning for Human Activity Recognition in Logistics*. In the following, the contributions are discussed in the context of these publications.

Temporal Convolutional Neural Networks for M-HAR

The contribution [MRGF+18] presents an extensive evaluation of the IMU Temporal Convolutional Neural Network (IMU-tCNN) architecture for M-HAR. Different configurations concerning the network's size, number of layers, number of units, maxout units, and learning settings were presented. In addition, the network has been deployed on two different benchmark datasets for M-HAR in ADL scenarios. The IMU-tCNN was introduced in a joint work with the co-authors Dr. René Grzeszick, and Jan Marius Lenk in [GLMR+17] for M-HAR applied to the order-picking process, using a dataset provided by the co-author Dr. Sascha Feldhorst in [FMHF16]. A new tCNN architecture was introduced for predicting order-picking activities [FMHF16]. The IMU-tCNN processes segments of multi-channel Time-Series from different OBDs, e.g., located on the hands and chest. The architecture uses parallel computing blocks to process the sequences and derive an intermediate feature representation per device. Feature representations from all the devices are fused to compute a global representation. These parallel blocks introduce an invariance with respect to human limbs, as human movements vary independently to the human limbs.

In [MRF18], the usage of attribute representations was introduced for M-HAR. A search for suitable attributes that represents signal segments is presented. As such representations did not exist, only annotations concerning coarse human activities are available. The authors presented a method for learning attributes that better represent time-series segments for solving M-HAR problems. This method encodes attributes as binary vectors. By using an evolutionary algorithm, it finds a representation that performs well for classification. The evolutionary algorithm initially assigns a representation to human activities, evaluates the representation using the performance on the validation set as a fitness metric, and mutates the representations with the best fitness values. The authors used three different deep architectures, a tCNN, a IMU-tCNN, Attribute Temporal Convolutional Neural Network (Attr-tCNN) and Attribute IMU-Temporal

Convolutional Neural Network (Attr-IMU-tCNN), which contain a sigmoid layer for predicting attributes. The Attr-IMU-tCNN from [GLMR+17, MRGF+18] using the learned attribute-representation showed the state-of-the-art performance of M-HAR for these two benchmark tasks and the order-picking dataset [FMHF16] when published.

Transfer Learning for M-HAR

The work in [MRF21] explores transfer learning for solving M-HAR among two different data sources: human joint poses and inertial measurements. It proposes to create a synthetic inertial measurements dataset, and to use it as a source for training a tCNN. The synthetic measurements correspond to the derivatives of human poses along time. The Logistic Activity Recognition Challenge (LARa) dataset, proposed in [NRMR+20], for M-HAR with human joint-poses is deployed as a source domain. Learned convolutional layers were utilized for initializing architectures deployed on benchmarking OBDs datasets, improving classification performance in three different benchmark datasets. The classification accuracy improves for the activities that are shared among the datasets. This approach can be considered as transfer learning across three target domains with different physical, but related, measurements, different number of OBDs, and recording rates.

The work in [AMRF22] extends Transfer Learning for Human Activity Recognition (Transfer Learning for HAR) using Synthetic On-body Device (SOBD) computed from pixel coordinates of human joints of datasets intended for video-based Human Activity Recognition (video-based HAR) in the wild. This transfer learning approach allows exploiting large collection of existing data with a variate range of activities. Classification performance improves when fine-tuning with a small amount of the target data.

Semi-Automatic Annotations for HAR

The author of this thesis collaborated strongly with the Lehrstuhl für Förder- und Lagerwesen (FLW) from the TU Dortmund University in works regarding practical applications of his research, summarised in the next paragraphs.

The joint contribution with Dr. Christopher Reining (FLW) in [RMRtF18] proposed a framework to reduce the annotation effort for creating an OBD dataset using a marker-based MoCap system as a reference for the order-picking process. The authors created a dataset composed of synchronously-recorded OBD and marker-based MoCap measurements in a laboratory scenario. In addition, it introduced proper annotation labels by defining logistic process steps, human activities and basic human limb movements in order-picking scenarios. The proposed dataset includes recordings from eight participants, each performing eight coarse activities. Sequences from OBDs are labelled using the

predictions from a IMU-tCNN, which has been trained using the marker-based MoCaps dataset, as both datasets are synchronized.

Attribute-based representations have been deeply explored on Zero-Shot Multi-channel Time-Series HAR (Zero-Shot M-HAR) in the joint contribution with Dr. Christopher Reining (FLW) in [RSH+18]. Particularly, in the manual order-picking process, attribute representations were expected to be beneficial for dealing with the versatility of activities. This contribution compared the performance of the Attr-tCNN and Attr-IMU-tCNN trained using different attribute representations. It evaluated their quantitative performance and quality from the practical application perspective. The marker-based MoCaps dataset from [RMRtF18] has been extended further, having more activities and participants. The dataset consisted of two parts—seen and unseen activities. The first part was used for training a Attr-tCNN, while the latter was used for testing. The unseen activities were described using attributes they shared with the seen activities. Despite having fewer attributes, Human-Labeled Attribute (HLA) representations performed better than a random one, created following the conclusions in [MRF18]. A semantic relation between attributes and activities enhances M-HAR not only quantitatively with regards to performance but also ensures a transfer of the attributes between activities by domain experts. In this preliminary work, the mapping between activity classes and attribute representations was one-to-one. Therefore, the deep architecture learned a multi-class classification problem. However, this direct mapping is not unique to a warehouse scenario.

The joint contributions in [RMRtF18, RSH+18] only considered the order-picking scenario. The performance of the classifiers based on annotated OBDs data was also not tested. Furthermore, the performance on unseen activities showed rather poor results, suggesting that the recorded dataset is class-imbalanced, not general and hardly discriminative. Besides, no proper approach for deriving attribute representations was available. Nonetheless, these contributions [RMRtF18, RSH+18] constituted initial attempts for creating OBDs data for M-HAR, i.e., the LARa dataset.

The manual annotation performance for creating a dataset for M-HAR is addressed in the joint contribution with Dr. Christopher Reining and Friedrich Niemann (FLW) in [RMRN+20]. Specifically, it evaluates the manual annotation of the LARa dataset in terms of expenditure of time for labelling and annotation consistency. A single domain expert revises initial annotations to measure its effect on the overall between-individual annotation consistency. Activity class labels and semantic attributes were annotated using the marker-based MoCap skeleton visualization. Within- and Between-individual annotation consistency based on the Cohen's kappa coefficient showed that annotators are moderately consistent when labelling recordings. Within-consistency is higher than between-consistency as minor disagreements among annotators are present. After an

annotation revision by a single-domain expert, the consistency for activity classes and attributes improved. However, this improvement is relatively small in comparison to the revision effort. This finding shows that more strict and clear guidelines for the first annotation shall be considered. This contribution presents the first version of the annotation tool for LARa dataset.

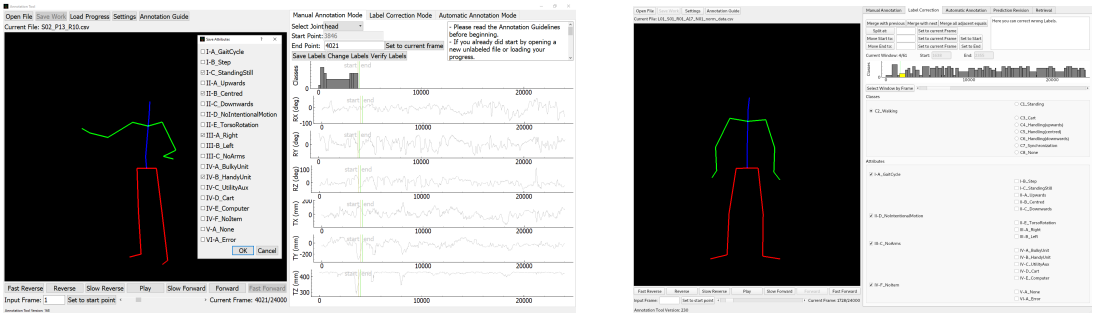
The joint contribution with Hülya Avsar, Erik Altermann, and Dr. Christopher Reining (FLW) in [AAR+21] evaluates the semi-automated annotation method, proposed in [RMRtF18], using the LARa dataset. The semi-automated approach uses a combination of predictions from a Attr-tCNN and manual revision for annotating human activities and their attribute representations. The contribution also compares the semi-automated annotation with the manual annotation. It also explores some adaptations of the Attr-tCNN, including training with recording samples of the testing subjects or using their deep representations. This annotation approach reduces the annotation effort, keeping quality similar to manual annotation [RMRN+20].

In the contribution [AMRRF22], the Human Activity Retrieval (HARetr) is proposed as a semi-automated annotation method for M-HAR. HARetr provides an ordered list of segmented windows similar to an activity class query in attribute space. An annotator provides the activity class query. It uses the attribute representation computed by a Attr-tCNN. Then, the annotator accepts or rejects the proposed activity label and its attributes. The HARetr allows annotating windows for a particular activity, reducing the mental burden of the annotator. The annotator concentrates on a single activity. Likewise, the revision process is guaranteed while annotating.

Along the aforementioned contributions, an annotation tool for M-HAR using marker-based MoCap recordings by means of a skeleton visualization is presented. This tool includes different annotation modi: a manual annotation modus used in [RMRN+20], where an annotator selects an activity class label and a set of attributes for a certain window of the sequences as Figure 1.1.1a shows; a manual revision as Figure 1.1.1b shows; a semi-automated modus used in [AAR+21] deploying the Attr-tCNN from [MRF18]; finally, a HARetr used in [AMRRF22]. This annotation tool is provided by Erik Altermann and the author of this thesis, and it is available in [MRA22].

Fine-grained Annotated Dataset for M-HAR

The joint contribution with Dr. Christopher Reining and Friedrich Niemann (FLW) in [RNMR+19] provides a systematic literature review of M-HAR for Production and Logistics. The review focuses on M-HAR using a marker-based MoCap system and OBDs. It provides an overview of M-HAR methods, i.e., statistical pattern recognition and DNNs. The work follows a very structured three steps-revision method. It uses a selection process



(a) Manual Annotation. (b) Manual Revision.

Figure 1.1.1: Annotation Tool for LARa dataset.

based on content criteria in the keywords, titles, abstract and full-text revision. Finally, a list of 52 publications results from the selection process. The systematic review organises the 52 resulting publications into different categorisations: Domain, Activity classes, Attachment, Dataset, Data preparation, Statistical methods and Deep Learning (DL).

The joint contribution in [NRMR+20] presents the LARa dataset for M-HAR in Production and Logistics. The LARa dataset composes of measurements of humans performing activities in logistics. Real-world warehousing scenarios are phisically recreated in a laboratory environment, ensuring natural motion and resemblance to reality, as Figure 1.1.2 shows. The laboratory scenario is the *Innovationlab Hybrid Services in Logistics* at the TU Dortmund University, and the Fraunhofer IML [RVBZH17]. The LARa dataset contains measurements of marker-based MoCap and two sets of OBDs from 14 humans performing eight activities from three scenarios in the intra-logistics. LARa contains sample-based fine-grained annotations of activities and attributes, following the conclusions in [RMRN+20]. The annotation process follows these steps: manual labelling by twelve annotators, a consistency revision of the annotated attribute representations comparing with expert-given semantic definition of the activities and its attributes, and a manual revision by four annotators. The LARa dataset is labelled using the annotation tool first presented in [RMRN+20], and improved for this contribution, showing its potential for annotating a large and demanding dataset. This dataset contains two versions: version (1) includes recordings of 14 subjects and is available in [NRMR+20]; version (2) provides a revised annotation of LARa, recordings of two additional subjects, and a series of its annotations from the manual to the second-round revision of the dataset, which is available in [NRMR+22].

In collaboration with *MotionMiners GmbH*, this thesis presents an additional dataset for M-HAR in the intralogistics. This dataset contains OBD recordings of three subjects

Figure 1.1.2: The LARa dataset includes recordings from marker-based MoCap and two sets of OBDs of 16 subjects performing activities in the intra-logistics.

from three real order-picking scenarios of warehousing. *MotionMiners GmbH* provided manual annotation of activity classes using a video-based annotation tool, where OBD and video recordings are synchronised using the camera's flash of a smartphone. Subjects wear three OBDs, one on each wrist and one on the torso. These OBDs are similar to one of the OBD sets from LARa dataset. Thus, this dataset will be used as a testing scenario for experimentation.

Knowledge-based and Data-driven M-HAR

The author of this thesis collaborated with the research group of Prof. Dr.-Ing. Thomas Kirste from the Rostock University in works regarding combining knowledge-based and data-driven M-HAR.

The joint work [MRLS+19] proposes a *hybrid* M-HAR method that combines Computational State-Space Models (CSSMs) and neural networks. The CSSM encodes prior knowledge about the high-level, causal structure of the task domain, and a tCNN acts as observation model, relating the sensor data to CSSM states. This work shows the general feasibility of such a hybrid method, although the proposed combination was simplistic and ad-hoc.

Additionally, the joint work [LMRA+21] investigates how information about high-level process steps can be integrated into a data-driven M-HAR system to increase recognition performance. Specifically, a Attr-tCNN computing an attribute representation from the OBD data is combined with a shallow classifier that estimates activities from predicted attributes and process step information.

1.2 STRUCTURE

The thesis is organized as follows: Chapter 2 revises the concepts of neural networks, specifically, Multi-Layer Perceptron (MLP) and its training algorithm; then, focusing on DNNs, i.e., CNN, and RNN. It also presents the concept of M-HAR describing a M-HAR system starting from data recording to statistical pattern recognition. In addition, it introduces the concept of Transfer Learning for M-HAR. Chapter 3 deepens into the related works of DNN for HAR, video-based HAR, and pose-based Human Activity Recognition (pose-based HAR). Approaches for video-based HAR and pose-based HAR introduce the concept of synthetic data for pose-based HAR, which becomes the basis for the synthetic data for M-HAR proposed in the method. Moreover, the chapter overviews related works for Transfer Learning for HAR using CNNs. Chapter 4 introduces attribute representation using DNN, giving an example on document analysis, where an attribute-based CNN was first documented. Furthermore, it introduces attribute representation for video-based HAR. Chapter 5 proposes the method of this thesis, namely, a Transfer Learning for M-HAR system that uses an attribute-based DNN and SOBDs for improving classification on target scenarios. Besides, it proposes a method for finding semantic attributes in the target scenarios. Chapter 6 shows the experiments on different benchmark OBD datasets for Transfer Learning for M-HAR. The chapter introduces the datasets and explains the implementation details. Then, it presents the evaluation in four steps: first, an evaluation of the proposed DNN architecture to be transferable to the benchmark datasets; second, the SOBD datasets from a rich pose dataset and pose-based HAR datasets; third, a search for semantic attributes using the proposed DNN, and an attribute representation; finally, the Transfer Learning for HAR combining the learnt attributes, the attributed-based DNN, and the SOBD under different transferability scenarios on the target datasets. Furthermore, the results are discussed. Chapter 7 presents a practical application for annotating OBD data for M-HAR using the proposed deep model and attribute representation. It defines the concept of HARetr. Finally, Chapter 8 concludes this work.

PUBLICATIONS

[AAR+21] Avsar, Huelya ; Altermann, Erik ; Reining, Christopher ; Moya Rueda, Fernando ; Fink, Gernot A.: Benchmarking Annotation Procedures for Multi-channel Time Series HAR Dataset. In: *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. Kassel, Germany, 2021, S. 453–458

[AMRF22] Awasthi, Shrutarv ; Moya Rueda, Fernando ; Fink, Gernot A.: Video-based Pose-Estimation Data as Source for Transfer Learning in Human Activity Recognition. In: *2022 26th International Conference on Pattern Recognition (ICPR)*. Montreal, QC, Canada : IEEE, 2022, S. 4514–4521

[AMRRF22] Altermann, Erik ; Moya Rueda, Fernando ; Rusakov, Eugen ; Fink, Gernot A.: Retrieval-based Annotation of Multi-channel Time-Series Data for HAR. In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. Pisa, Italy : IEEE, 2022. – ISBN 978–1–66541–647–4, S. 212–217

[GLMR+17] Grzeszick, Rene ; Lenk, Jan M. ; Moya Rueda, Fernando ; Fink, Gernot A. ; Feldhorst, Sascha ; ten Hompel, Michael: Deep Neural Network Based Human Activity Recognition for the Order Picking Process. In: *Proceedings of the 4th International Workshop on Sensor-Based Activity Recognition and Interaction*, Association for Computing Machinery, 2017 (iWOAR '17). – ISBN 978–1–4503–5223–9, S. 6

[LMRA+21] Lüdtke, Stefan ; Moya Rueda, Fernando ; Ahmed, Waqas ; Fink, Gernot A. ; Kirste, Thomas: Human Activity Recognition using Attribute-Based Neural Networks and Context Information. In: *3rd International Workshop on Deep Learning for Human Activity Recognition*. Montreal, August 2021

[MRA22] Moya Rueda, Fernando ; Altermann, Erik: *Annotation Tool for Logistic Activity Recognition Challenge (LARa) Github*. https://github.com/wilfer9008/Annotation_Tool_LARa. Version: 2022

[MRF18] Moya Rueda, Fernando ; Fink, Gernot A.: Learning Attribute Representation for Human Activity Recognition. In: *4th International Conference on Pattern Recognition (ICPR)*. Beijing, China : IEEE, 2018, S. 523–528

[MRF21] Moya Rueda, Fernando ; Fink, Gernot A.: From Human Pose to On-body Devices for Human Activity Recognition. In: *Proc. Int. Conf. on Pattern Recognition*. Milan, Italy : IEEE, 2021. – ISBN 978–1–72818–808–9, S. 10066–10073

[MRGF17] Moya Rueda, Fernando ; Grzeszick, René ; Fink, Gernot A.: Neuron Pruning for Compressing Deep Networks Using Maxout Architectures. In: *Proc. 39th German Conf. Pattern Recognition*. Basel, Switzerland : Springer International Publishing, 2017. – ISBN 978–3–319–66709–6, S. 177–188

[MRGF+18] Moya Rueda, Fernando ; Grzeszick, René ; Fink, Gernot A. ; Feldhorst, Sascha ; ten Hompel, Michael: Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors. In: *Informatics* 5 (2018), Nr. 2. http://dx.doi.org/10.3390/informatics5020026. – DOI 10.3390/informatics5020026. – ISSN 2227–9709

[MRLS+19] Moya Rueda, Fernando ; Lüdtke, Stefan ; Schröder, Max ; Yordanova, Kristina ; Kirste, Thomas ; Fink, Gernot A.: Combining Symbolic Reasoning and Deep Learning for Human Activity Recognition. In: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2019, S. 22–27

[NRMR+20] Niemann, Friedrich ; Reining, Christopher ; Moya Rueda, Fernando ; Nair, Nilah R. ; Steffens, Janine A. ; Fink, Gernot A. ; ten Hompel, Michael: LARa: Creating a Dataset for Human Activity Recognition in Logistics Using Semantic Attributes. In: *Sensors* 20 (2020), Nr. 15. http://dx.doi.org/10.3390/s20154083. – DOI 10.3390/s20154083. – ISSN 1424–8220

[RMRN+20] Reining, Christopher ; Moya Rueda, Fernando ; Niemann, Friedrich ; Fink, Gernot A. ; ten Hompel, Michael: Annotation Performance for multi-channel time series HAR Dataset in Logistics. In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Austin, TX, USA : IEEE, 2020, S. 1–6

[RMRtF18] Reining, Christopher ; Moya Rueda, Fernando ; ten Hompel, Michael ; Fink, Gernot A.: Towards a Framework for Semi-Automated Annotation of

Human Order Picking Activities Using Motion Capturing. In: *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*. Poznan, Poland : IEEE, 2018. – ISBN 978–1–5386–2371–8, 817–821

[RNMR+19] Reining, Christopher ; Niemann, Friedrich ; Moya Rueda, Fernando ; Fink, Gernot A. ; ten Hompel, Michael: Human Activity Recognition for Production and Logistics—A Systematic Literature Review. In: *Information* 10 (2019), Nr. 8. http://dx.doi.org/10.3390/info10080245. – DOI 10.3390/info10080245. – ISSN 2078–2489

[RSH+18] Reining, Christopher ; Schlangen, Michelle ; Hissmann, Leon ; ten Hompel, Michael ; Moya, Fernando ; Fink, Gernot A.: Attribute Representation for Human Activity Recognition of Manual Order Picking Activities. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. Berlin, Germany : Association for Computing Machinery, 2018 (iWOAR '18). – ISBN 978–1–4503–6487–4

LARA DATASET

[NRMR+20] Niemann, Friedrich ; Reining, Christopher ; Moya Rueda, Fernando ; Altermann, Erik ; Nair, Nilah R. ; Steffens, Janine A. ; Fink, Gernot A. ; ten Hompel, Michael: *Logistic Activity Recognition Challenge (LARa Version 01) – A Motion Capture and Inertial Measurement Dataset*. http://dx.doi.org/10.5281/zenodo.3862782. Version: Mai 2020. – Acknowledgement: The work on this publication was supported by Deutsche Forschungsgemeinschaft (DFG) in the context of the project Fi799/10-2, HO2403/14-2 "Transfer Learning for Human Activity Recognition in Logistics".

[NRMR+22] Niemann, Friedrich ; Reining, Christopher ; Moya Rueda, Fernando ; Bas, Hülya ; Altermann, Erik ; Nair, Nilah R. ; Steffens, Janine A. ; Fink, Gernot A. ; ten Hompel, Michael: *Logistic Activity Recognition Challenge (LARa Version 02) – A Motion Capture and Inertial Measurement Dataset*. http://dx.doi.org/10.5281/zenodo.5761276. Version: Februar 2022. – Acknowledgement: The work on this publication was supported by Deutsche Forschungsgemeinschaft (DFG) in the context of the project Fi799/10-2, HO2403/14-2 "Transfer Learning for Human Activity Recognition in Logistics".

FUNDAMENTALS 2

HAR is the task of automatically giving particular activity labels to sensor recordings of human movements. HAR processes signals from videos, marker-based Motion Capturing System (marker-based MoCap), or multi-channel Time-Series, e.g., measurements from On-body Devices. The latter ones are very important as they make HAR a potential tool beyond constrained or laboratory settings. OBDs are not affected by occlusion, and they do not portray human identities, as in the case of video-based HAR.[1] OBDs are low-cost devices that are highly reliable, non-invasive and easy to use. OBDs contain tri-axial inertial sensors, which measure different types of derived physical quantities along three dimensions, e.g., acceleration, angular velocities, and magnetic field strength. Additionally, OBDs may contain vital human sensors, e.g., temperature and heart rate monitors. They provide a view into the movement of the subject wearing the devices. Ideally, they would assist anywhere and anytime by observing activities egocentrically. This work refers to HAR using OBD recordings, as multi-channel time-series Human Activity Recognition (M-HAR).

Methods of statistical pattern recognition were used for supervised-M-HAR. These methods follow a standard pipeline: segmentation, extraction of handcrafted features, primitives computation, dimensionality or feature reduction, and a training stage and classification. Nowadays, Deep Learning (DL) methods are prevalent for M-HAR. They combine feature extraction, feature or dimensionality reduction, and classification in a holistic approach. This thesis concentrates on DNNs, in particular, on the Temporal Convolutional Neural Network (tCNN) for solving M-HAR. The tCNNs are end-to-end architectures that combine learnable temporal convolutional filters along the time axis with non-linear operation functions, downsampling and classification.

As follows, this chapter first focuses on basic concepts of tCNN, starting from CNN architectures. Second, it gives an overview of HAR with regards to the type of data, preprocessing, annotation and statistical pattern recognition methods. Third, it presents basic concepts of transfer learning, focusing on M-HAR. Finally, it introduces attribute representations for classification as a concrete example for transfer learning.

1 OBD recordings contains also information regarding the subjects identities or their soft-biometrics [NMRRF23].

2.1 NEURAL NETWORKS

Artificial Neural Networks (ANNs) are networks of primitive functions called artificial neurons. Artificial neurons are vaguely inspired by neuroscience [Kono5, HM19]. An artificial neuron is conceived as a simplistic model of a biological neuron, modelling based on how brain cells store and process information. From biological neural networks, neurons have input signals from previous neurons' outputs, also called axons. These input signals are transmitted to the neuron body via synapses or input connections weighted by internal parameters or weights [Roj96, Kono5, HM19]. These weights are learnable, influencing the behaviour of the neuron. This influence can be excitatory or inhibitory with positive or negative weights, respectively [Roj96]. The weighted signals are accumulated in the neuron body until reaching certain threshold, firing the neuron.

Nevertheless, an artificial neuron has gone far off from the neuroscientific insight of a biological neuron. An artificial neuron or perceptron is a primitive function of N parameters that produces a numerical output. This primitive function contains an integration component, as a linear excitatory or inhibitory response, and an activation function or non-linear component. Perceptrons become a basis for more complicated deep learning models. Each input connection is associated with a weight w_n. In the neuron body, the weighted input signals are summed up. Finally, an arbitrary non-linear function uses the summation as an argument for activating the perceptron. For example, if the final summation is above a threshold value, the neuron will activate and send a signal along its output. The connection of multiple artificial neurons results in an ANN.

2.1.1 *Multi-Layer Perceptron (MLP)*

A neuron or perceptron computes the scalar product between its input $\mathbf{x} \in \mathbb{R}^N$ and its weights $\mathbf{w} \in \mathbb{R}^N$, with N the number of inputs. It adds a bias $\mathbf{b}$—considering it as a negative threshold—, and thresholds the resulting value with zero as activation function $\varphi(\cdot)$, computing an output y, as Equation 2.1.1:

$$y = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geqslant 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1.1}$$

A Single-Layer Perceptron (SLP) separates the input space into two subspaces. Instead of a comparison with a threshold value, an SLP utilizes a non-linearity function as the activation function $\varphi(\cdot)$. A SLP has an extended-input vector $\mathbf{x} \in \mathbb{R}^{N+1}$, and an extended-weight vector $\mathbf{w} \in \mathbb{R}^{N+1}$; where the bias is considered as a weight with a corresponding

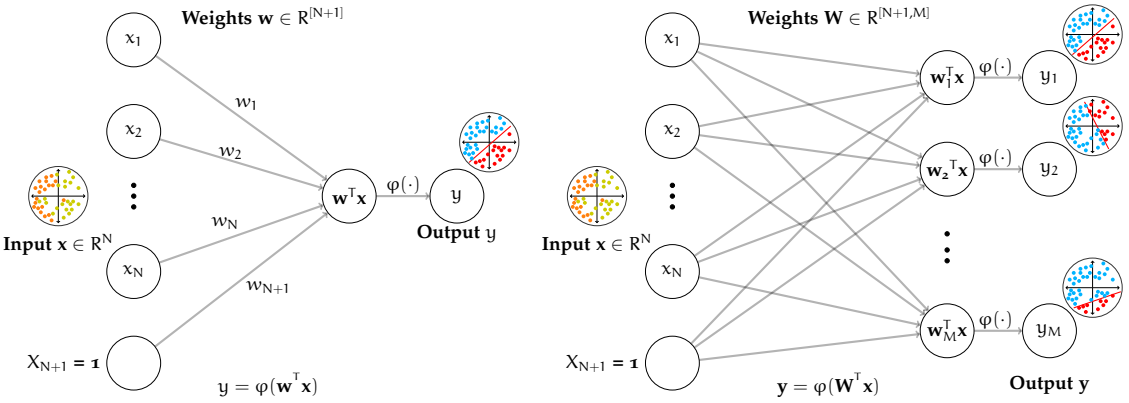


Figure 2.1.1: Single-Layer Perceptron (SLP) (left). Fully Connected Layer (FC) with M Single-Layer Perceptrons (SLPs) (right), where all neurons are fully or dense connected to the input vector.

input $x_{N+1} = 1$. This ANN is the simplest feedforward neural network and can be considered as a linear classifier. A SLP implements the function,

$$y = \varphi\left(\mathbf{w}^T\mathbf{x}\right). \tag{2.1.2}$$

An ANN with M SLPs implements a set of M linear classifiers $\mathbf{f} = \{f_m\}_{m=0}^M$, producing a M-dimensional vector $\mathbf{y} \in \mathbb{R}^M$. This ANN is called Fully Connected Layer (FC), where all the SLPs are connected to each element in the input, implementing Equation 2.1.3,

$$\mathbf{z} = \boldsymbol{\varphi}\left(\mathbf{W}^T\mathbf{x}\right), \tag{2.1.3}$$

with $\mathbf{x} = [x_1, \cdots x_N, 1]^T$, $\mathbf{W} = \{\mathbf{w}_m\}_m^M$, and $\boldsymbol{\varphi}(\cdot)$ as the activation function; see Figure 2.1.1.

SLPs are arranged in layers, connected to other perceptrons in neighbouring layers. A MLP is an ANN with an input layer, an output layer and multiple in-between multiple FCs. Their input is forwarded through the network layers to the output layer. The in-between layers are called hidden layers, as they are not observable. MLPs are able to approximate any continuous function, given enough SLPs in the hidden layers. Given a large set of annotated data—input sequence with observed annotations—, MLPs can be trained by changing their weights to approximate arbitrary functions or classify input data into any set of predefined classes. Its layers are FCs since neurons are completely connected to

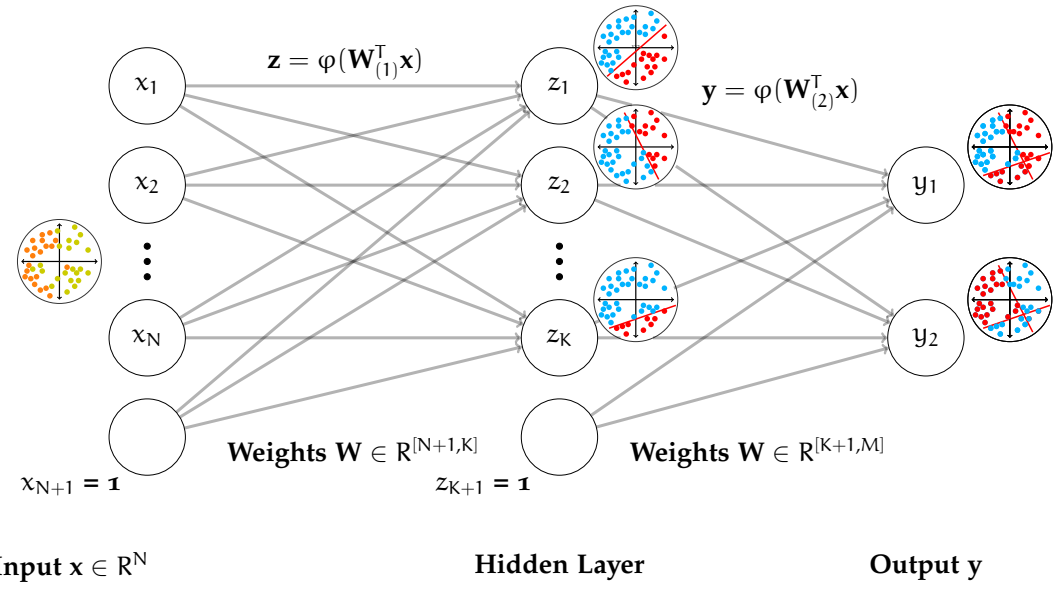


Figure 2.1.2: MLP with two hidden FCs and an output layer.

neurons from the immediate previous layer. Figure 2.1.2 shows an example of a MLP of three layers: an input layer with $N+1$ neurons with an input vector $\mathbf{x} \in \mathbb{R}^{N+1}$, a hidden layer with K neurons with parameters $\mathbf{W}_{(1)} \in \mathbb{R}^{[N+1\times K]}$, and an output layer with M neurons with parameters $\mathbf{W}_{(2)} \in \mathbb{R}^{[K+1\times M]}$ computing the output vector $\mathbf{y} \in \mathbb{R}^2$ [Roj96]—here, the vectors $\mathbf{x}$ and $\mathbf{z}$ and the weight matrices are extended for including the bias. The MLP in Figure 2.1.2 represents a chain of function compositions, where the output of the hidden layer becomes the input of the last layer. It implements the mapping $\phi : \mathbb{R}^N \to \mathbb{R}^2$ as follows,

$$\mathbf{z} = \boldsymbol{\varphi}\left(\mathbf{W}_{(1)}^{\mathsf{T}}\mathbf{x}\right) \tag{2.1.4}$$

$$\mathbf{y} = \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{\mathsf{T}}\mathbf{z}\right) = \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{\mathsf{T}}\varphi\left(\mathbf{W}_{(1)}^{\mathsf{T}}\mathbf{x}\right)\right), \tag{2.1.5}$$

with $\mathbf{x} = [x_1, x_2, \cdots x_N, 1]^{\mathsf{T}}$, $\mathbf{z} = [z_1, z_2, \cdots z_K, 1]^{\mathsf{i}}\,\mathrm{ntvar}\mathsf{T}$, and $\mathbf{y} = [y_1, y_2, \cdots y_M]^{\mathsf{T}}$. The output $\mathbf{y}$ is a function composition since its input is the output of hidden layer $\mathbf{z}$.

Activation Function

The perceptron in Equation 2.1.1 uses the step function as the non-linear activation function $\varphi(\cdot)$. This activation function compares the sum of the weighted inputs with a threshold in "0". However, this step function is discontinuous precisely at "0", being a problem for the learning algorithm for adapting the weights. ANNs utilize differentiable non-linear activation functions. These functions map their input values within a finite range. In essence, these non-linear functions are continuous and differentiable, and they are an important element in the optimisation algorithm [Roj96, GBC16]. The optimization algorithm requires the computation of gradients. The most common non-linearity functions and their derivatives are:

- **Sigmoid Function (*Sg*):** The *Sg* takes its argument and crushes it to the range of $[0, 1]$ with an interception with y-axis at $x = 0.5$, and horizontal asymptotes at $y = 1$ and $y = 0$ for large positive and negative input values, respectively.

$$\varphi(x) = \frac{1}{1 + e^{-x}} \qquad \varphi'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \varphi(x)\,(1 - \varphi(x)) \tag{2.1.6}$$

- **Tanh Function (*tanh*):** The *tanh* is a scaled, translated and zero-centred *Sg* function, mapping its input argument into the range $[-1, 1]$.

$$\varphi(x) = \tanh(x) = 2 \cdot (Sg(x) - 1) = \frac{1 - e^{-x}}{1 + e^{-x}} \qquad \varphi'(x) = (\tanh)'(x) = \left(1 - \tanh^2(x)\right) \tag{2.1.7}$$

  The *Sg* and *tanh* saturate to high and low argument values. This saturation hinders the learning process, as gradients approach to zero [GBC16].

- **Rectified Linear Unit (ReLU) Function:** The ReLU thresholds at zero,

$$\varphi(x) = \max(0, x) \qquad \boldsymbol{\varphi}'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & x \geqslant 0 \end{cases}, \tag{2.1.8}$$

  being closer to step function of the perceptron in Equation 2.1.1 [GBB11, GBC16].

The Rectified Linear Unit (ReLU) performs better than the saturating functions because it does not saturate for large argument values. Thus, the activation function obtains sparse output representations [GBB11, KSH12]. Therefore, ReLU controls the number of active neurons, causing a lesser burden in computation when optimising. Active neurons become a path between the output and the input. Along this path, the relation between the output of activated neurons and the input of the ANN is linear. Moreover, this path helps the optimisation algorithm since gradients only flow back on the path of activated neurons, and the remaining ones stay unchanged [GBB11, GBC16]. This behaviour supports the conclusion in [KSH12], showing that the ReLU is faster for training than the saturating non-linearity, e.g., *Sg* and *tanh*. Moreover, ReLU allows for ANN compression, as their weights are sparse [MRGF17]. However, the risk of having many dead neurons increases with a deficient learning algorithm, reducing generalisation.

- **Leaky ReLU Function:** The leaky ReLU includes a small slope α for negative values instead of mapping to zero as ReLUs.

$$\varphi(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & x \geqslant 0 \end{cases} \qquad \boldsymbol{\varphi}'(x) = \begin{cases} \alpha & \text{if } x < 0 \\ 1 & x \geqslant 0 \end{cases} \tag{2.1.9}$$

Gradient Descent (GD)

The learning process seeks to find a set of weights for an ANN by minimizing an objective function $\mathbf{E}(\boldsymbol{w})$ in weight-space when evaluated on a validation dataset [Roj96, GBC16]. The objective function $\mathbf{E}(\boldsymbol{w})$ computes the difference between an expected vector $\mathbf{y}$ and the computed output vectors $\hat{\mathbf{y}}$ of an ANN. For minimising $\mathbf{E}(\boldsymbol{w})$, the learning process uses an optimization algorithm, called Gradient Descent (GD). The GD finds a local minimum of an objective function in terms of the weights. For an ANN, GD repeatedly computes the gradient of an objective function $\mathbf{E}(\mathbf{w})$ with respect to the weights. Then, it adapts its parameters proportionally to the negative gradient of the objective function following,

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \Delta \mathbf{E}(\mathbf{w}), \tag{2.1.10}$$

with $\mathbf{w}_i \in \mathbb{R}^K$ as the weights of an ANN in optimization step i, $\mathbf{w}_{i+1} \in \mathbb{R}^K$ the updated weights, γ a fixed parameter, $\mathbf{E}(\mathbf{w})$ the objective function, and $\Delta\mathbf{E}(\mathbf{w})$ is

$$\Delta\mathbf{E}(\boldsymbol{w}) = \frac{\partial\mathbf{E}(\boldsymbol{w})}{\partial\boldsymbol{w}} = \left[\frac{\partial\mathbf{E}(\boldsymbol{w})}{\partial\mathbf{w}_1}, \frac{\partial\mathbf{E}(\boldsymbol{w})}{\partial\mathbf{w}_2}, \cdots \frac{\partial\mathbf{E}(\boldsymbol{w})}{\partial\mathbf{w}_K}\right]. \tag{2.1.11}$$

The parameter γ is the step length related to the proportion of the gradient for updating the weights, called the learning rate [Roj96]. It is an important parameter since it directly affects the convergence to the local minimum. Small steps to the expected direction lead to slow convergence. In contrast, large steps lead to faster progress, but with the possibility of overshooting the local minimum, producing bigger values of $\mathbf{E}(\boldsymbol{w})$. Besides, a larger γ increases the number of dead neurons when using a ReLU activation function as $\varphi(\cdot)$.

The objective function $\mathbf{E}(\mathbf{w})$ is considered with respect to a training set $\mathbf{D} = \left\{(\mathbf{x}, \mathbf{y})^{(n)}\right\}_n^N$ with N sample-observation tuples $(\mathbf{x}, \mathbf{y})$. There are three versions of GD depending on the size and use of the training set $\mathbf{D}$ of sample-observation tuples: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD) and Stochastic Batch Gradient Descent (SBGD). The BGD considers the entire set $\mathbf{D}$ as a large batch—the term batch does not imply minibatch. SGD draws a sample-observation tuple randomly from $\mathbf{D}$ updating the weights sample by sample. The SBGD or Stochastic Minibatch Gradient Descent (SMGD) considers a minibatch or, common used simply batch, B with random selected sample-observation tuples from $\mathbf{D}$. SBGD updates the weights after feeding the batch to the network [GBC16]. DL methods deploy SBGD, selecting a batch with a compromise between training time and computation capability. SBGD with small batches need more steps to reach an acceptable minimum. This limitation comes as the estimated gradient variates strongly, resulting in selecting a lower learning rate. However, the SBGD has a regularisation effect on the training process.

There are also other versions of the GD that add more hyper-parameters. Examples of these GD approaches are:

- **GD with momentum:** The GD with momentum considers the previous change of weights; see Equation 2.1.12. The momemtum helps to avoid oscillations of the gradient direction. Moreover, it minimises the risk of getting trapped at a local minimum [Kon05, Roj96].

$$\begin{aligned} \boldsymbol{\nu}_{i+1} &= \gamma_1\boldsymbol{\nu}_i - \gamma_2\Delta\mathbf{E}(\mathbf{w}) \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + \boldsymbol{\nu}_{i+1} \end{aligned}, \tag{2.1.12}$$

with $\mathbf{w}_i$ and $\mathbf{w}_{i+1}$ being the weights and updated weights of a ANN in iteration i, $\boldsymbol{\nu}_i$ and $\boldsymbol{\nu}_{i+1}$ the weight changes and updated weight changes in iteration i, γ_1 the momentum constant or rate, γ_2 the learning rate, and $\mathbf{E}(\mathbf{w})$ the objective function.

- **GD with momentum and weight decay:** penalises the weight values by γ_3, as

$$\begin{aligned} \boldsymbol{\nu}_{i+1} &= \gamma_1 \boldsymbol{\nu}_i - \gamma_2 \Delta \mathbf{E}(\mathbf{w}) - \gamma_3 \gamma_2 \boldsymbol{w}_i \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + \boldsymbol{\nu}_{i+1} \end{aligned}, \tag{2.1.13}$$

being γ_3 weight decay.

- **AdaGrad:** adapts the learning rates of each of the weights inversely proportional to the square root of the sum of the accumulated squared values of the gradient [GBC16].

$$\begin{aligned} \boldsymbol{\eta}_{i+1} &= \boldsymbol{\eta}_i + \Delta \mathbf{E}\left(\mathbf{w}\right)^2 \\ \mathbf{w}_{i+1} &= \mathbf{w}_i - \frac{\gamma_1}{\sqrt{\boldsymbol{\eta}_{i+1}}} \Delta \mathbf{E}\left(\mathbf{w}\right), \end{aligned} \tag{2.1.14}$$

with $\boldsymbol{\eta}_{i+1}$ the accumulated gradient in iteration i. Here, the $\frac{\gamma_1}{\sqrt{\boldsymbol{\eta}_{i+1}}}$ is applied element-wise.

- **RMSProp:** uses an exponentially weighted moving average for adapting the learning rates. This approach converges faster when finding a convex area around a minimum by discarding past accumulated squared gradients according to a decay rate α [GBC16].

$$\begin{aligned} \boldsymbol{\eta}_{i+1} &= \alpha \boldsymbol{\eta}_i + (1-\alpha) \Delta \mathbf{E}\left(\mathbf{w}\right)^2 \\ \mathbf{w}_{i+1} &= \mathbf{w}_i - \frac{\gamma_1}{\sqrt{\boldsymbol{\eta}_{i+1}}} \Delta \mathbf{E}\left(\mathbf{w}\right) \end{aligned}, \tag{2.1.15}$$

with $\frac{\gamma_1}{\sqrt{\boldsymbol{\eta}_{i+1}}}$ is applied element-wise.

The **AdaGrad** and **RMSProp** are approaches that adapt the learning rates for each weight, and they can be extended by including the momentum. The **RMSProp** with momentum is used frequently in the networks focused on in this thesis.

There are different objective functions $\mathbf{E}(\boldsymbol{w})$ according to the problem, e.g., classification, segmentation and regression. For example, for a batch set $\mathbf{B} \sim \mathbf{D}$ with B

sample-observation tuples $\mathbf{B} = \left\{(\mathbf{x}, \mathbf{y})^{(b)}\right\}_{b=1}^{B}$, and an output vector $\hat{\mathbf{y}}^{(b)} \in \mathbb{R}^{M}$ for batch b:

- **Total Sum Squared Error (TSSE):**

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2B} \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \sum_{m=1}^{M} \left(\mathbf{y}_m^{(b)} - \hat{\mathbf{y}}_m^{(b)}\right)^2 \tag{2.1.16}$$

- **Total Binary Cross-Entropy (TBCE):**

$$\mathbf{E}(\mathbf{w}) = \frac{1}{BM} \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \prod_{m=1}^{M} \hat{\mathbf{y}}_m \log\left(\mathbf{y}_m\right) + (1 - \hat{\mathbf{y}}_m) \log\left(1 - \mathbf{y}_m\right) \tag{2.1.17}$$

- **Total Categorical Cross-Entropy (TCCE)**

$$\mathbf{E}(\mathbf{w}) = \frac{1}{B} \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \sum_{m=1}^{M} \hat{\mathbf{y}}_m \log\left(\mathbf{y}_m\right) \tag{2.1.18}$$

EXAMPLE Consider the GD from Equation 2.1.10, the SLP in Equation 2.1.2, with a single output, i.e., with $M = 1$, and the TSSE, Equation 2.1.16, as the objective function $\mathbf{E}(\mathbf{w})$, the $\Delta\mathbf{E}(\mathbf{w})$ for a a batch $B \sim \mathbf{D}$ from training set $\mathbf{D}$ of sample-observation tuples is

$$\frac{\partial \mathbf{E}(w)}{\partial w} = \frac{\partial}{\partial w} \left(\sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \frac{1}{2} \left(\mathbf{y}^{(b)} - \varphi\left(w^{\mathsf{T}} \mathbf{x}^{(b)}\right)\right)^2 \right), \tag{2.1.19}$$

solving for $\partial\mathbf{w}$ using the chain rule

$$\frac{\partial \mathbf{E}(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \left(\mathbf{y}^{(b)} - \varphi\left(\mathbf{w}^{\mathsf{T}} \mathbf{x}^{(b)}\right)\right) \varphi'\left(\mathbf{w}^{\mathsf{T}} \mathbf{x}^{(b)}\right) \mathbf{x}^{(b)}. \tag{2.1.20}$$

$\Delta \mathbf{E}(\boldsymbol{w})$ depends on the derivative of the activation function $\boldsymbol{\varphi}'(\cdot)$, i.e., the GD behaves different for each of the non-linearity functions. On the assumption of using the sigmoid as $\boldsymbol{\varphi}(\cdot)$, Equation 2.1.6, the final gradient $\Delta \mathbf{E}(\boldsymbol{w})$ is

$$\frac{\partial \mathbf{E}(\boldsymbol{w})}{\partial \boldsymbol{w}} = - \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \left(\mathbf{y}^{(b)} - \hat{\mathbf{y}}^{(b)}\right) \hat{\mathbf{y}}^{(b)} \left(1 - \hat{\mathbf{y}}^{(b)}\right) \mathbf{x}^{(b)}. \tag{2.1.21}$$

Backpropagation

The previous example uses GD for a SLP where the input and output vectors are observed, so the objective function $\mathbf{E}(\mathbf{w})$ can be computed. However, the hidden neurons are not observable for a MLP. The backpropagation exploits the function composition of the output of a MLP, Equation 2.1.5, the chain rule for differentiating, and the GD. The backpropagation is divided into two phases: a forward step and a backward step. In the forward step, sample-tuples $(\mathbf{x}, \mathbf{y})$ are fed to the MLP. Then, the backward step computes the gradients with respect to the weights from the final layer to the initial layer—the gradients are propagated backwards by using the chain rule successively—, and it updates the weights as the GD.

EXAMPLE For better describing the backpropagation, the MLP in the Figure 2.1.2, the TSSE (Equation 2.1.16) are considered as the objective function $\mathbf{E}(\mathbf{w})$, and SBGD with a batch $B \sim \mathbf{D}$ from training set $\mathbf{D}$. First, the MLP processes an input sample $\mathbf{x} \in \mathbb{R}^N$ computing a hidden vector $\mathbf{z} \in \mathbb{R}^K$, using Equation 2.1.4, and an output vector $\hat{\mathbf{y}} \in \mathbb{R}^M$ with Equation 2.1.5.—here, the vectors $\mathbf{x}$ and $\mathbf{z}$ and the weight matrices are extended including the bias. Second, the backpropagation implements the backward step, i.e, it computes the gradients for each layer from the output layer back to the input layer, and it updates the weights following the updating rule of GD, Equation 2.1.10. The weight-updates for the MLP are:

$$\mathbf{W}_{(2)i+1} = \mathbf{W}_{(2)i} - \gamma \frac{\partial \mathbf{E}(\mathbf{W}_1, \mathbf{W}_2)}{\partial \mathbf{W}_2}, \tag{2.1.22}$$

$$\mathbf{W}_{(1)i+1} = \mathbf{W}_{(1)i} - \gamma \frac{\partial \mathbf{E}(\mathbf{W}_1, \mathbf{W}_2)}{\partial \mathbf{W}_1}, \tag{2.1.23}$$

being $\mathbf{W}_{(1)} \in \mathbb{R}^{[N,K]}$ and $\mathbf{W}_{(2)} \in \mathbb{R}^{[K,M]}$.

Each of the weight-updates considers the gradient of $\mathbf{E}(\mathbf{w})$ with respect to each of the weights, in this case all the terms in the matrices $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$. The objective function $E(\mathbf{w})$, Equation 2.1.16, in terms of $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$ becomes:

$$E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right) = \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \frac{1}{2} \sum_{m=1}^{M} \left(\mathbf{y}^{(b)} - \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(b)}\right)\right)^2, \tag{2.1.24}$$

and

$$E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right) = \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \frac{1}{2} \sum_{m=1}^{M} \left(\mathbf{y}^{(b)} - \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{T} \boldsymbol{\varphi}\left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(b)}\right)\right)\right)^2. \tag{2.1.25}$$

The objective function $E(\mathbf{w})$ with respect to $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$ Equation 2.1.24 and Equation 2.1.24 is developed in Appendix (Section A.1), resulting in the equations (Equation A.1.5 and Equation A.1.7). These equations are described with respect to local gradients as:

$$\frac{\partial E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)}{\partial \mathbf{W}_{(2)}} = - \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \boldsymbol{\delta}_{y}^{(b)} \mathbf{z}^{(b)} \tag{2.1.26}$$

$$\begin{aligned} \frac{\partial E(\mathbf{W}_{(1)}, \mathbf{W}_{(2)})}{\partial \mathbf{W}_{(1)}} &= - \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \sum_{m=1}^{M} \left[\boldsymbol{\delta}_{y_m^{(b)}} \mathbf{W}_{(2)_m}\right] \boldsymbol{\varphi}'\left(\mathbf{W}_{(1)} \cdot \mathbf{x}^{(b)}\right) \mathbf{x}^{(b)} \\ &= - \sum_{\left\{(\mathbf{X},\mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B}} \boldsymbol{\delta}_{\mathbf{z}^{(b)}} \mathbf{x}^{(b)} \end{aligned} \tag{2.1.27}$$

with,

$$\boldsymbol{\delta}_{\hat{\mathbf{y}}^{(b)}} = \left(\mathbf{y}^{(b)} - \hat{\mathbf{y}}^{(b)}\right) \boldsymbol{\varphi}'\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(b)}\right) \tag{2.1.28}$$

$$\boldsymbol{\delta}_{\mathbf{z}^{(b)}} = \sum_{m=1}^{M} \left[\boldsymbol{\delta}_{y_m^{(b)}} \mathbf{W}_{(2)_m}^{T}\right] \boldsymbol{\varphi}'\left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(b)}\right). \tag{2.1.29}$$

The local gradients Equation 2.1.28 and Equation 2.1.29 depend on the derivatives of the activation functions. As these derivatives are expressed in terms of the layer outputs, the local gradients depend on the outputs. The backpropagation first forwards the inputs computing outputs of the neurons; second, it computes local gradients per layer and feeds them backwards from the output to the input layers. In general, the backpropagation sends backwards the local gradients from the output to the input layers—the gradient of a hidden layer depends on the gradient of the next layer.

For a MLP with L layers with weight matrices $\mathbf{W}_{l,l+1}$ for $l = 1, 2..., L-1$, the TSSE as objective function, B batch $\left\{(\mathbf{X}, \mathbf{y})^{(b)}\right\}_{b=1}^{B} \in \mathbf{B} \sim \mathbf{D}$, and using the sigmoid as activation function Equation 2.1.6, the local gradients become:

$$\delta_l = \begin{cases} diag\left(\hat{\mathbf{y}}_{(l)}^{(b)}\right)\left[I_{[M,M]} - diag(\hat{\mathbf{y}}_{(l)}^{(b)})\right]\left[\mathbf{y}^{(b)} - \hat{\mathbf{y}}_{(l)}^{(b)}\right]^T & l = L \\ diag\left(\hat{\mathbf{y}}_{l}^{(b)}\right)\left[I_{k_l x k_l} - diag\left(\mathbf{z}_{l}^{(b)}\right)\right]\mathbf{W}_{l,l+1}^{T}\delta_{l+1} & l = 1, 2, ..., L-1 \end{cases} \tag{2.1.30}$$

and the weight-updates become

$$\mathbf{W}_{(l)_{i+1}} = \mathbf{W}_{(l)_i} - \gamma \delta_l^{(b)} \mathbf{z}_l^{(b)} \quad l = 1, 2, ...L. \tag{2.1.31}$$

2.1.2 *Recurrent Neurons*

A MLP assumes that all sample-tuples are mutually independent. Temporal relations from the input data becomes relevant for an ANN to model a time series, e.g., for processing inertial recordings for M-HAR. RNNs are neural networks specifically designed to process sequential data. They use recurrent connections in every neuron [HS97, GMH13, OR16]. A unit-time delayed and weighted activation is fed back to the neuron, which provides the neuron with a memory capability. This capability allows learning the temporal relations of sequential data from past activations.

Given an input sequence $\mathbf{X} = (\mathbf{x}_{t=1}, \cdots, \mathbf{x}_{t=T})$ with T number of samples, a standard recurrent neuron computes the hidden vector sequence $\mathbf{H} = (\mathbf{h}_1, \cdots, \mathbf{h}_T)$ and an output vector sequence $\mathbf{Y} = (\mathbf{y}_{t=1}, \cdots, \mathbf{y}_{t=T})$ by iterating the following equations from $t = 1$ to T following,

$$\begin{aligned} \mathbf{h}_t &= \varphi\left(\mathbf{W}_{(xh)}\mathbf{x}_t + \mathbf{W}_{(hh)}\mathbf{h}_{t-1} + \mathbf{b}_{(h)}\right) \\ \mathbf{y}_t &= \mathbf{W}_{(hy)}\mathbf{h}_t + \mathbf{b}_{(y)}, \end{aligned} \tag{2.1.32}$$

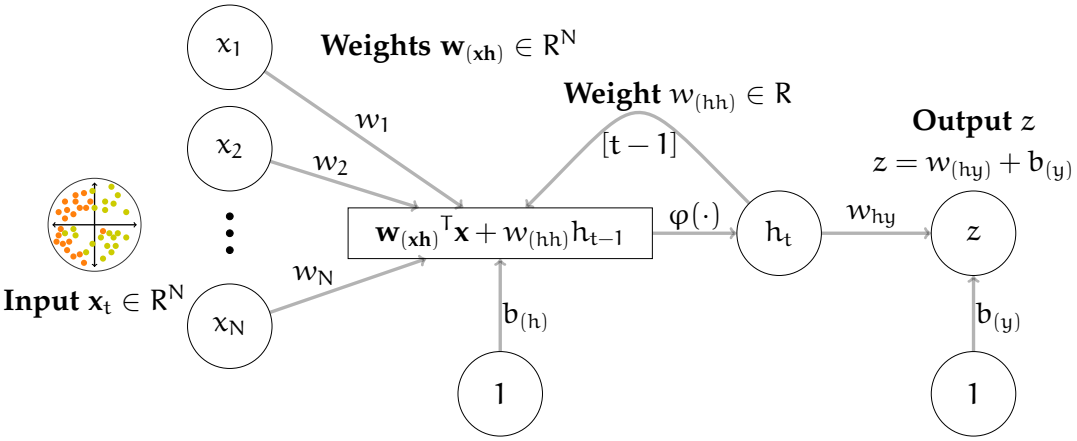


Figure 2.1.3: Recurrent neuron.

where $\mathbf{W}_{(xh)}$ is the input-hidden weight matrix, $\mathbf{W}_{(hh)}$ is the hidden-hidden weight matrix, $\mathbf{W}_{(hy)}$ is the hidden-output weight matrix, $\mathbf{b}_{(h)}$ and $\mathbf{b}_{(y)}$ are bias vectors, and $\boldsymbol{\varphi}(\cdot)$ is the hidden activation function.

Recurrent neurons, Equation 2.1.32, process sequential data from $t = 1$ to T, learning temporal correlations from the past. However, a sequence can also be processed backwards; in such case, a recurrent neuron will be able to learn temporal relations from the future, at least from a sequence of length T. A bidirectional recurrent neuron computes the forward hidden sequence $\overrightarrow{\mathbf{h}}$, the backward hidden sequence $\overleftarrow{\mathbf{h}}$ and the output sequence $\mathbf{Y}$ by iterating the backward layer from $t = T$ to $t = 1$, the forward layer from $t = 1$ to $t = T$ and then updating the output layer:

$$\begin{aligned}\overleftarrow{\mathbf{h}}_t &= \varphi\left(\mathbf{W}_{\left(x\,\overleftarrow{h}\right)}\mathbf{x}_t + \mathbf{W}_{\left(\overleftarrow{h}\,\overleftarrow{h}\right)}\overleftarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\left(\overleftarrow{h}\right)}\right)\\ \overrightarrow{\mathbf{h}}_t &= \varphi\left(\mathbf{W}_{\left(x\,\overrightarrow{h}\right)}\mathbf{x}_t + \mathbf{W}_{\left(\overrightarrow{h}\,\overrightarrow{h}\right)}\overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\left(\overrightarrow{h}\right)}\right)\end{aligned} \tag{2.1.33}$$

$$\mathbf{y}_t = \mathbf{W}_{\left(\overrightarrow{h}\,y\right)}\overrightarrow{\mathbf{h}}_t + \mathbf{W}_{\left(\overleftarrow{h}\,y\right)}\overleftarrow{\mathbf{h}}_t + \mathbf{b}_{(y)}. \tag{2.1.34}$$

Long short-term Memory (LSTM)

Long short-term Memorys (LSTMs) extend RNNs with memory cells and a gating system to learn temporal relationships of long-time sequences. The gating system uses component-

wise multiplication of the input with different gates, namely, the input, output and forget gates. Each of these gates represents a different operation for the recurrent cell. These operations are *write*, *read* and *reset* for the input, output and forget gate. LSTM units do not suffer from exploding or vanishing gradients during training [HS97]. The input provided to an LSTM updates its cell state according to the activation of the gates. The activation of the LSTM units is calculated similar to the RNNs, as in Equation 2.1.32. The hidden value $\mathbf{h}_t$ of an LSTM cell is updated at every time step t [GMH13]. A LSTM is implemented in the following composite function,

$$\begin{aligned}
\mathbf{i}_t &= \boldsymbol{\varphi}\left(\mathbf{W}_{(xi)}\mathbf{x}_t + \mathbf{W}_{(hi)}\mathbf{h}_{t-1} + \mathbf{W}_{(ci)}\mathbf{c}_{t-1} + \mathbf{b}_{(i)}\right) \\
\mathbf{f}_t &= \boldsymbol{\varphi}\left(\mathbf{W}_{(xf)}\mathbf{x}_t + \mathbf{W}_{(hf)}\mathbf{h}_{t-1} + \mathbf{W}_{(cf)}\mathbf{c}_{t-1} + \mathbf{b}_{(f)}\right) \\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh\left(\mathbf{W}_{(xi)}\mathbf{x}_t + \mathbf{W}_{(hi)}\mathbf{h}_{t-1} + \mathbf{b}_{(c)}\right) \\
\mathbf{o}_t &= \boldsymbol{\varphi}\left(\mathbf{W}_{(xo)}\mathbf{x}_t + \mathbf{W}_{(ho)}\mathbf{h}_{t-1} + \mathbf{W}_{(co)}\mathbf{c}_{t-1} + \mathbf{b}_{(o)}\right) \\
\mathbf{h}_t &= \mathbf{o}_t \tanh\left(\mathbf{c}_t\right),
\end{aligned} \tag{2.1.35}$$

where $\boldsymbol{\varphi}(\cdot)$ is the *Sg* activation function, and $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$, and $\mathbf{c}$ are respectively the input gate, forget gate, output gate, and cell activation vector, all of which are the same size as the hidden vector $\mathbf{h}$ defining the hidden value. The matrices $\mathbf{W}_{(gate\,a,\,gate\,b)}$ are weights matrices with subscripts representing relationships from $gate\,a$ to $gate\,b$. The vectors $\mathbf{b}_{(gate\,a)}$ represent the biases of gate $(gate\,a)$.

BackPropagation Through Time (BPTT)

For training a RNN such as the LSTM network, the backpropagation algorithm is extended to consider the sequential behaviour; this is called BackPropagation Through Time (BPTT). For a RNN network, $\boldsymbol{\phi}_{rnn}(\cdot)$, with K number of recurrent neurons and that processes a sequence from $t = 1$ to T. The network is unrolled in time to obtain a MLP $\boldsymbol{\phi}_{mlp}(\cdot)$ with a number of layers T, i.e., the MLP is composed of T layers, one layer for each time step $t = [1, \cdots, T]$. Besides, each neuron in $\boldsymbol{\phi}_{mlp}(\cdot)$ has an image from the corresponding neuron in $\boldsymbol{\phi}_{rnn}(\cdot)$; this image includes the weights of the neuron. This conceptualization regards the problem of training a RNN as training a feedforward network with certain constraints imposed on its weights. The BPTT approach calculates the $\frac{\partial E(\mathbf{W}_{(l)})}{\partial \mathbf{W}_{(l)}}$ in layer l from $\boldsymbol{\phi}_{rnn}(\cdot)$ by simply computing the partial derivatives of $\partial E\left(\mathbf{W}_{(l)}\right)$ with respect to each of the $t - T$ weights in $\boldsymbol{\phi}_{mlp}(\cdot)$ corresponding to the layer $l = t$ and adds them up. Hence, the problem of computing the gradient signals throughout the layers in the recurrent network $\partial E\left(\mathbf{W}_{(l)}\right)$ reduces to the backpropagation of a deep MLP $\boldsymbol{\phi}_{mlp}(\cdot)$.

2.1.3 *Convolutional Neural Networks*

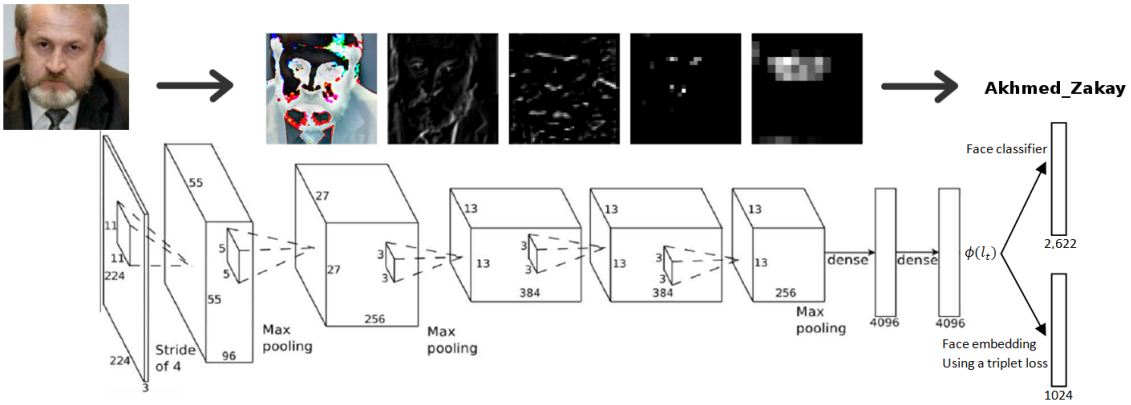


Figure 2.1.4: In image classification, pixels are assembled into hierarchical structures or features, i.e., edglets → motifs → parts → objects → scenes. This hierarchy suggests that recognition architectures for image classification should have multiple trainable extractors of these structures stacked on top of each other. CNNs respond to these needs. Image adapted from [RSA15], where a CNN classifies face images from [PVZ15].

CNNs[2] are hierarchical structures that combine convolutional operations with learnable filters and non-linear activation functions, downsampling operations, and classifiers. These structures either map their input into a more compact representation, or classify their input into a set of labels, depending on their objective function [LKF10, KSH12, PVZ15]. However, their structure is partially distinct compared to MLPs, i.e., their neurons are 3D filters that activate depending on small regions or receptive fields of their inputs [Roj96]. These neurons are called convolutional neurons. They compute a convolution operation[3] between the connected inputs and their internal parameters. Furthermore, they activate depending on convolution output and a non-linearity function [FM82]. Conceptually, the convolutional neurons are similar to a perceptron of an SLP, i.e., they compute a summation of the weighted input and use a non-linearity; see Equation 2.1.2. DNN became the standard approach in different fields, for example, in image classification [KSH17], image segmentation, word spotting from historical documents [WF22], and even in non-visual domains like acoustics.

The advantage of CNNs over ANNs with perceptrons is that they assume inputs are images (width, height, channel) [LKF10]. They allow to encode certain properties of

2 CNNs were first proposed in [FM82] and later addressed by [LBBH98], introducing the well-known LeNet-5 for handwritten digit recognition [GBC16].

3 In image processing, a convolutional operation consists of multiplying the pixel's and its neighbours' intensities by a kernel or small matrix, which is slid through the entire image along the spatial dimensions.

images into their architecture, i.e., their neurons activate when they "see" particular features extracted at different locations in the image; they are also robust against shifts in position, and feature distortions [FM82]. CNNs arrange neurons in layers with a 3D shape, called convolutional layers. Each of the layers gets a 3D-input volume, called a feature map, and transforms it into another through convolution and non-linearity operations [LKF10]. By stacking layers and downsampling their outputs, CNNs extract more complex and abstract feature maps, which, at the same time, are invariant to distortions and translations. By stacking convolutional layers and down sampling feature maps, filters in each layer will learn to give more weight and to activate to different patterns: from simple pixels to edglets (oriented edges or blob of colours), edglets to motifs, and eventually, from motifs to more complex patterns; such as parts of faces, or parts of cars. Hence, the deepest neurons activate to a combination of features from feature maps of previous convolutional layers. Since neurons are connected in a hierarchical structure, the deepest neuron receptive fields cover bigger areas from the image input. Consequently, the activation of deepest neurons is not affected by shift variations and feature deformations in the image input. These layers compute final descriptions of the input images, which can be considered global representations of images. The last part of CNNs are standard FCs, usually, a 3-layered MLP; they classify the input images using the extracted representation depending on an objective function.

Compared to a MLP that processes a 3D input volume, CNNs have less parameters—in this case assuming that the MLP is connected to all the values in the 3D input. This parameter reduction results from the convolutional operation since the layer slices the same filter through the entire feature map input. Thus, CNNs are easier to train than ANNs—at least for the same task—[KSH12]. The parameter reduction is an advantage of CNNs because their generalisation performance is improved, i.e, a learned CNN will not just model the training data, but in addition they will generalise new data more accurately than ANNs, in other words, they avoid overfitting, and the learning speed is increased [LDS89].

Architecture

A CNN is a feedforward network composed of layers, similar to an ANN that transform a feature map input to final class scores by forwarding it layer by layer. All its layers are not fully-connected, but they are generally of three types: a convolutional layer, pooling layers and a fully-connected layer. The input and output of each layer are sets of tensors called feature maps.

INPUT The input is a feature map of different dimensions depending on the problem domain. The input for image processing are 3D feature maps of size $[W, H, C]$, representing pixel intensities arranged in a matrix of $[W, H]$ for three channels, namely, red, green and blue. For video processing, the input is a 4D feature map consisting $[T, W, H, C]$ of a sequence of 3D feature maps. Usually, there is a temporal aggregation of 3D feature maps before feeding to the CNN; or 3D feature maps at different times $t = 1, \cdots, T$ are fed individually to a CNN, which performs a late fusion.[4] For sequential data processing, e.g., audio or inertial measurements, a feature map consists of 1D array of recordings per sensor, i.e., a feature map of $[W, S, 1]$.

CONVOLUTIONAL LAYERS They consists of C neurons arranged in a 3D volume. Neurons are composed of learnable filters $W \in \mathbb{R}^{[F_w, F_h, C]}$ and a bias $b \in \mathbb{R}$ that activate under a specific type of feature at some spatial position in the feature map input $\mathbf{Z}^{(l-1)}$ from layer $l-1$ producing a feature map of weighted summations $\mathbf{Z}^{(l)}$. $\mathbf{W}$ size is determined by a given receptive field $F = [F_w, F_h]$ along the width and height, and the feature map inputs depth $C = C^{(l-1)}$. Each of the neurons computes convolutions with small regions in $\mathbf{Z}^{(l-1)}$.

$$\mathbf{Z}^{(l)}_{i,j,k^{(l)}} = \varphi\left(\sum_{c=1}^{C^{(l-1)}} \sum_{f_h=-\left\lfloor \frac{F_h}{2} \right\rfloor}^{\left\lfloor \frac{F_h}{2} \right\rfloor} \sum_{f_w=-\left\lfloor \frac{F_w}{2} \right\rfloor}^{\left\lfloor \frac{F_w}{2} \right\rfloor} \mathbf{W}^{k^{(l)}}_{\frac{F_w}{2}+f_w, \frac{F_h}{2}+f_h, c} \cdot \mathbf{Z}^{(l-1)}_{i-f_w, j-f_h, c} + b^{k^{(l)}}\right) \quad \forall k = 1, \cdots C^{(l)}, \quad (2.1.36)$$

with $\varphi(\cdot)$ being the activation function.

The convolutional operation performs similar to the summation of SLPs, collecting information coming from previous feature maps. A single convolutional layer l with $C^{(l)}$ neurons or filters produces a 3D weighted feature map $\mathbf{Z}^{(l)}$ by convolving each of its filters along the spatial dimensions, width and height, of the feature map input $\mathbf{Z}^{(l-1)}$ and stacking the 2D weighted feature maps $\{\mathbf{Y}_c\}_c^{C^{(l)}}$ along the depth dimension C of the layer [LDS89, Roj96]. As convolutional layers use a filter along the spatial dimensions of the input, the filter parameters are shared along these dimensions, i.e., detecting the same feature along the spatial dimensions—, what the authors in [LDS89] called weight sharing. The size of the weighted feature map output $[W^{(l)}, H^{(l)}, C^{(l)}]$ depends on three hyper-parameters of the layer: the depth C, stride S, and zero-padding P. The C

4 Late fusion refers to merging independent preprocessed sequences from different channels of the multi-channel Time-Series data input.

parameter represents the number of channels or neurons along the depth dimension of the convolutional layer. The S parameter is the number of pixels that one slides each filter along the width and height of a feature map input, and P is the number of zeros in the border of a feature map input to fit neurons along spatial dimensions of the feature map.

$$W^{(l)} = \frac{\left(W^{(l-1)} - F_w + 2P\right)}{S} + 1 \qquad H^{(l)} = \frac{\left(W^{(l-1)} - F_h + 2P\right)}{S} + 1 \qquad C^{(l)} = C^{(l-1)} \tag{2.1.37}$$

TEMPORAL CONVOLUTIONAL LAYERS A tCNN uses temporal convolutions for processing sequential data, e.g., from recordings of microphones for audio tasks or inertial sensors for HAR. Having a N-dimensional sequence, from s sensors, a sliding window of size W moves forwards with a frame-shift of F segmenting sequences. These sequences are of size $[W, N]$. In tCNN, convolutional layers convolve their feature-map inputs with C filters along the temporal axis. Having a feature map $\mathbf{Z}^{(l-1)} \in \mathbb{R}^{[W,N,C^{(l-1)}]}$ in layer $l-1$, and a set of $C^{(l-1)}$ filters $\mathbf{W} \in \mathbb{R}^{[F,1,C^{(l-1)}]}$ and biases $b^{(l-1)}$ connecting layers $l-1$ and l, a temporal-convolution for each $s = 1, \cdots, S$ sensor follows,

$$\mathbf{Z}^{(l)}_{i,s,k^{(l)}} = \varphi\left(\sum_{c=1}^{C^{(l-1)}} \sum_{f_w=-\lfloor \frac{F_w}{2} \rfloor}^{\lfloor \frac{F_w}{2} \rfloor} \mathbf{W}^{k^{(l)}}_{\frac{F_w}{2}+f_w, \frac{F_h}{2}+f_h, c} \cdot \mathbf{Z}^{(l)}_{i-f_w,s,c} + b^{k^{(l)}}\right) \quad \forall k = 1, \cdots, C^{(l)}, \tag{2.1.38}$$

with $\varphi(\cdot)$ being the activation function.

POOLING LAYERS Pooling refers to dimensionality reduction that is usually used in CNNs to encourage spatial invariance and capacity bottleneck. They are placed among different convolutional layers to reduce the spatial size of the feature maps by downsampling.

- **Stride-based Pooling**

  It takes small squared regions of size $[F_w, F_h]$ with a stride of S_p along the spatial dimensions of the feature map input, and it performs a downsampling operation per region. This operation can be the average over the regions, or the maximum value in each region. The final size of the downsampled feature map is $[W_p, H_p, C_p]$

for a feature map input y_{act}. Max pooling shows good empirical performance for object recognition [KSH12].

$$W_p^{(l)} = \frac{(W^{(l-1)} - F_w)}{S_p} + 1 \qquad H_p^{(l)} = \frac{(W^{(l-1)} - F_h)}{S_p} + 1 \qquad C_p^{(l)} = C^{(l-1)} \quad (2.1.39)$$

- **Spectral Pooling**

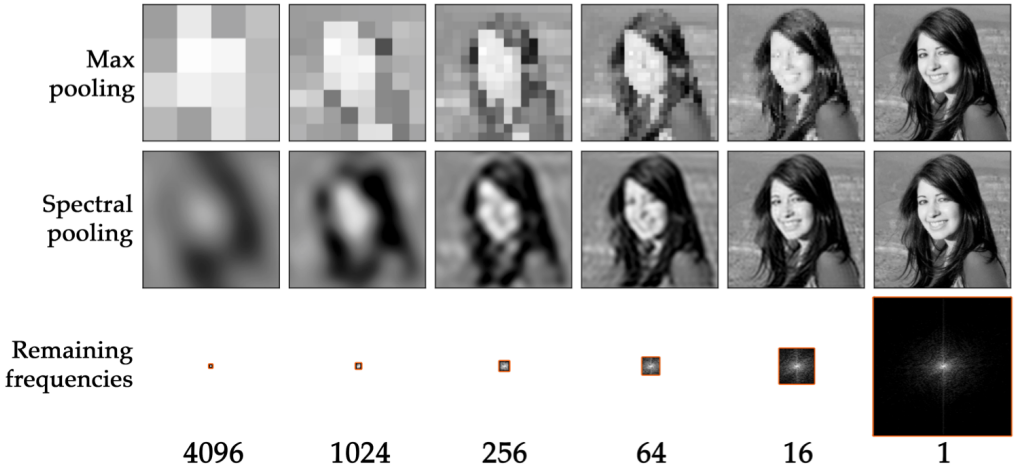


Figure 2.1.5: Comparison of max- and spectral pooling functions for different factors of dimensionality reduction. Image taken from [RSA15].

However, stride-based pooling functions exhibit a poor preservation of information. They imply a very sharp dimensionality reduction, e.g., by at least a factor of four, whenever applied to two-dimensional inputs. Besides, the maximum value in each window only reflects very local information and often does not represent the window's contents well [RSA15].

For a given feature map input $\mathbf{Z}^{(l-1)} \in \mathbf{R}^{[W,H,C]}$ and some desired output $[W_p, H_p, C]$, the Discrete Fourier Transform (DFT) of the input $\mathbf{Z}_{\mathbf{freq}}$ follows Equation 2.1.40. For image classification, and under the assumption that the DC component has been shifted to the centre of the transformed input, as Figure 2.1.5 shows, $\mathbf{Z}_{\mathbf{freq}}$ is cropped only in the $[W_p, H_p]$-size matrix. Finally, the $\mathbf{Z}_{\mathbf{freq}}^{[W_p,H_p]}$ is transformed back to the spatial domain via the Inverse Discrete Fourier Transform (IDFT), resulting in $\mathbf{Z}^{(l)} \in \mathbb{R}^{[W_p,H_p,C]}$. The spectral pooling retains more information than the stride-based pooling functions. This pooling performs linear low-pass filtering, exploiting that

images with spatial structure carry most of their information on lower frequencies. Furthermore, spectral pooling allows to select any arbitrary output dimensionality.

$$\mathbf{Z_{freq}}^{(l-1)} = \mathcal{F}\left(\mathbf{Z}^{(l-1)}\right) \tag{2.1.40}$$

$$\mathbf{Z}^{(l)} = \mathcal{F}^{-1}\left(\underset{[W_p,H_p]}{\text{crop}}\ \mathbf{Z_{freq}}^{(l-1)}\right) \tag{2.1.41}$$

- **Spatial Pyramid Pooling layer (SPP)**

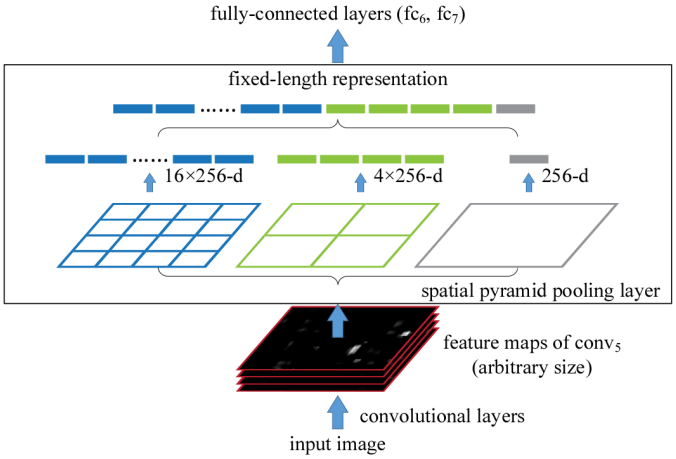


Figure 2.1.6: Spatial Pyramid Pooling (SPP) layer in a CNN. Image taken from [HZRS15].

The authors in [HZRS15] pointed out that convolutional layers do not need fixed-size feature-map inputs since they perform a convolution operation and their filters are not fully-connected to their inputs. However, the FCs need, fundamentally, fixed-size feature-map inputs. The last convolutional layer is the only one that should generate a fixed-size feature map as the first FC is connected to it. For example, the Alexnet uses $[224 \times 224 \times 3]$ cropped images for image recognition [KSH12]. For that reason, the authors in [HZRS15] replaced the last max-pooling layer with a new layer to eliminate the need for fixed-size input images for image recognition. This new layer is called the Spatial Pyramid Pooling (SPP), whose idea comes from the spatial pyramid matching used originally as an extension of Bag of Features Representation (BagFR) [LSP06].

The SPP layer divides an input from finer to coarser levels; it aggregates their information, generating local outputs; and it concatenates these outputs into an overall feature output; see Figure 2.1.6. Its advantage lies in the generation of fixed-size outputs from multisized inputs, and it maintains spatial information. More concretely in CNNs, a l-level SPP Layer (SPP Layer) layer divides a feature-map input, whose depth is C, into $C \cdot M$ spatial bins with $M = \sum_1^l 2^l$, where the level l denotes the number of divisions along the spatial dimensions of the feature-map Figure 2.1.6. Then, the SPP layers concatenate the maximum or the average value of each bin into a vector. This vector becomes the feature map of the first FC; see Figure 2.1.6. By using a SPP layer, the CNN's input can be of any scale.

- **Temporal Pyramid Pooling (TPP)**

  In case of these sequential models, partitioning along the horizontal axis is important when dealing with sequential data. The authors in [SF18] proposed the Temporal Pyramid Pooling (TPP) for improving retrieval results in word-spotting. The TPP splits the input along the horizontal axis, e.g., following the sequential writing, or the time axis for M-HAR. An L-level TPP splits its input into $2^l \, \forall \, l = [1, ..., L]$ non-overlapping horizontal cells. For a input layer $\mathbf{Z_{freq}}^{(l-1)} \in \mathbb{R}^{[W,S]}$, a cell is of size $\left[\frac{W}{2^l}, S\right] \; \forall \, l = [1, ..., L]$. Each cell covers the entire vertical axis of the sequence. The pooling is thus only done along the time axis, and each cell roughly represents features from consecutive intervals of the input data, e.g., from a word image. Stacking multiple of these pooling layers with different number of splits, along the axis of writing, results in a pyramid representation. This representation encodes the progression of sequence, hence the name Temporal Pyramid Pooling.

FULLY-CONNECTED LAYERS These layers have the same topology as an MLP, or one can see them as a convolutional layer, but their filters are fully-connected to a feature map input. Those layers perform matrix multiplication with the feature map input, add a bias, and use a non-linearity function to get activated.

2.2 HUMAN ACTIVITY RECOGNITION

HAR attempts to automatically discern the activities a subject carries out in a certain period of time. HAR processes signals from videos [FPZ16], marker-based MoCap, or multi-channel Time-Series, e.g., measurements from OBDs [RC15]. The latter ones are very important as they make HAR a potential tool beyond constrained or laboratory settings. OBDs are not affected by occlusion, and they do not portray human identities—human

identification is not possible by only regarding their multi-channel Time-Series data, as in the case of videos.[5]

HAR is, in general, a classification task. However, this classification is a challenging task due to large intra- and inter-class variations of human activities. Humans perform similar tasks differently; even a single person carries out a task differently. For example, in order picking, workers might change how they pick up a box depending on the box size and texture or the worker fatigue. Furthermore, M-HAR datasets suffer from the class imbalance problem, where there exist more samples of the frequent activities, e.g., walking or standing, than picking up an item [FMHF16, OR16]. This problem depends strongly on the annotation, e.g., at MotionMiners, the walking and standing are usually ignored as they are not crucial for application purposes.

A HAR system seeks to classify the activities of a subject based on the recording of physical quantities related to the subject's movement. The system consists of different stages: data recording, data representation, preprocessing, and classification, either by statistical feature recognition or by DNNs.

2.2.1 *Data Recording*

A HAR system acquires raw measurements of humans performing activities using different types of sensor systems. Advances in pervasive computing and sensors allowed the development of a variety of sensor modalities for recording information of humans performing activities. As a result, there exist different sensor modalities for HAR, either placed in the environment, e.g., video cameras and ambient sensors, or placed on the human body, e.g., OBDs.

Ambient Sensors

These devices are usually embedded in the subject's environment, e.g., on objects that the subject employs. They include a wide variety of sensors, such as motion detectors, door sensors, object sensors, pressure sensors, and temperature sensors [CFK13, KEK08].

Video Sequences

Video cameras provide a dense feature space for HAR, which allows for detailed recognition of activities with regards to the scene and objects a human employs for an activity.

5 However, OBD recordings might contain identity information, given the inter-person variation of activities. Besides, recording biases can provide enough information for identifying subjects, e.g., initialisation noise. Additionally, soft-biometrics such as height or gender can still be obtained from these recordings [NMRRF23].

Spatio-temporal features are extracted from video sequences for representing activities [KY18, SZ14]. The use of video cameras, however, raises privacy issues. This thesis references HAR using video data as video-based HAR.

On-Body Devices

OBDs are very common for capturing activity-related information from subjects. Placement of the devices on the body limbs strategically helps to capture important information related to a particular movement of the body limbs. Their conjunction provides insights into the overall activity performed by a subject. OBDs include 3D inertial sensors, e.g., accelerometers, gyroscopes, magnetometers, or sensors embedded in smartphones; vital devices, e.g., heart rate devices; and radio frequency identification sensors and tags [CFK13, TDF+18]. Authors have deployed a different number of OBDs on different parts of the human body, depending on the application. The authors in [FMHF16, GLMR+17] used three OBDs on the two wrists and on the torso. In contrast, the authors in [CLCG18, LYA09, ZCS19] recorded acceleration measurements from only one OBD on the waist. The positioning of sensors plays an important role in M-HAR, and it is a limiting factor for many real applications. The sensors' positioning is often largely driven by user acceptance rather than optimality of M-HAR performance [TDF+18].

OBD systems have a variety of recording configurations, e.g., sampling rate and amplitude of devices. For example, dataset authors have used recordings rates in the range of 1 to 300 Hz, as [RNMR+19]. Unfortunately, there is no consensus in the community on what is the best choice for these configuration parameters given the different types of activities or applications [TDF+18]. There are cases where lower sampling frequencies may be preferable for long experiments to reduce the energy consumption, but sacrificing higher frequency aspects of human motions [TDF+18].

Contrastively, authors pre-processed OBD recordings, creating a different data representation. The authors in [LYC17, LC11, MHB+16] used the magnitude of the acceleration from the three dimensional components $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$. Moreover, the authors in [TLLY18] proposed using the logarithm magnitude of two-dimensional DFT of OBDs. They used this magnitude as an input image for training a CNN.

Motion Capturing System

Other option for M-HAR is using multi-channel Time-Series from recordings of human joint poses, e.g., from marker-based MoCap. A pose is a combination of position and orientation information. Marker-based MoCap systems are for example the Kinect [CFK13], or the Vicon system [NRMR+20]. The authors in [DLGY12] represented marker-based

MoCap data by using the 3D-joint poses specifying a posture. They divided the human joints into five groups, according to the main human body parts, Left Arm (LA), Right Arm (RA), Left Leg (LL), Right Leg (RL), Neck or Torso (NT). Similarly, the authors in [KY18] divided the human joint pixel coordinates from video datasets into five groups for video-based HAR; see Section 3.2.

The authors in [RSH+18, NRMR+20] interpreted the joint poses marker-based MoCap as multi-channel Time-Series, where each component $[x, y, z]$ of the spatial dimension from the joint pose is considered individually. This approach is similar to how acceleration data are handled in this thesis. The authors in [VFD+17] used a geometrical and parametric representation of the joint poses and their velocities and accelerations, i.e., the quaternions and Euler angles representations of human poses, along with their velocities and accelerations of human joints, as input data.

Domain and Task for HAR

Let define the term *domain* for HAR, following [CFK13, PY10].

Definition 1. ***Domain:*** *For HAR, the domain $\mathcal{D}$ is defined by a multi-channel Time-Series space $\mathcal{X}^{[W,S]}$, which may represent the S-dimensional space with S the number of sensors in the OBDs, measuring physical quantities within a given time window W and a marginal probability distribution over all possible measurements $\mathbf{p}(\mathbf{X})$. A domain $\mathcal{D}$ is a tuple $(\mathcal{X}, \mathbf{p}(\mathbf{X}))$, where $\mathbf{X} = \{\mathbf{X}^{(n)}\}_{n=0}^{N} \in \mathcal{X}^S$ is a set of N recordings samples from multi-channel Time-Series space $\mathcal{X}$.*

2.2.2 *Preprocessing*

Pre-processing is necessary due to the different characteristics of sensors in OBDs, namely, recording sampling rates, physical measurement units, random noise or malfunctioning. For example, low- and high-pass filtering separates acceleration components from gravity, the human body, and noise. The authors in [FK16, STBO17] used a median filter for smoothing the input signal, e.g., an acceleration profile.

The authors in [KLLK10] used a third-order average filter for reducing random noise. Low- and high-pass filters have been used to eliminate noise or separate the acceleration into two components: one due to body movements and one due to gravity. The authors in [LYA09, BPT14, WGWH18] argued that the low-frequency component of the acceleration is due to gravity, and the high-frequency component to the dynamic motion of the human body. The authors in [SR12, SSH13, AI15] computed the gravity component by averaging the acceleration measurements. This average can be seen as taking the zero-frequency component. Afterwards, the body acceleration was computed by subtracting the gravity

component from acceleration measurements. The authors in [STBO17] separated these two components with a low-pass filter. A low-pass Butterworth filter is used for such separation in [NTJ18].

In contrast, a low-pass filter [AI15, KKB14], a third-order low-pass Butterworth filter [FK16], and an average filter [GCBCJG14] are used for reducing noise. The authors in [NTJ18] deployed noise filters before segmentation by sampling fix-sized windows, but there was no explanation of the particular method. The authors in [DBLG14] used a discrete low-pass filter scaling past samples and reducing noise.

The authors in [GW15, RC16] used a zero-mean and unit-variance normalisation, and the authors in [GLMR+17, MRGF+18] used a max-min normalisation to the range of $[0, 1]$, as there are differences among the units and scales of the measurements from the OBDs. The authors in [CX15] did not use any preprocessing and forwarded the raw data of a single OBD to a CNN.

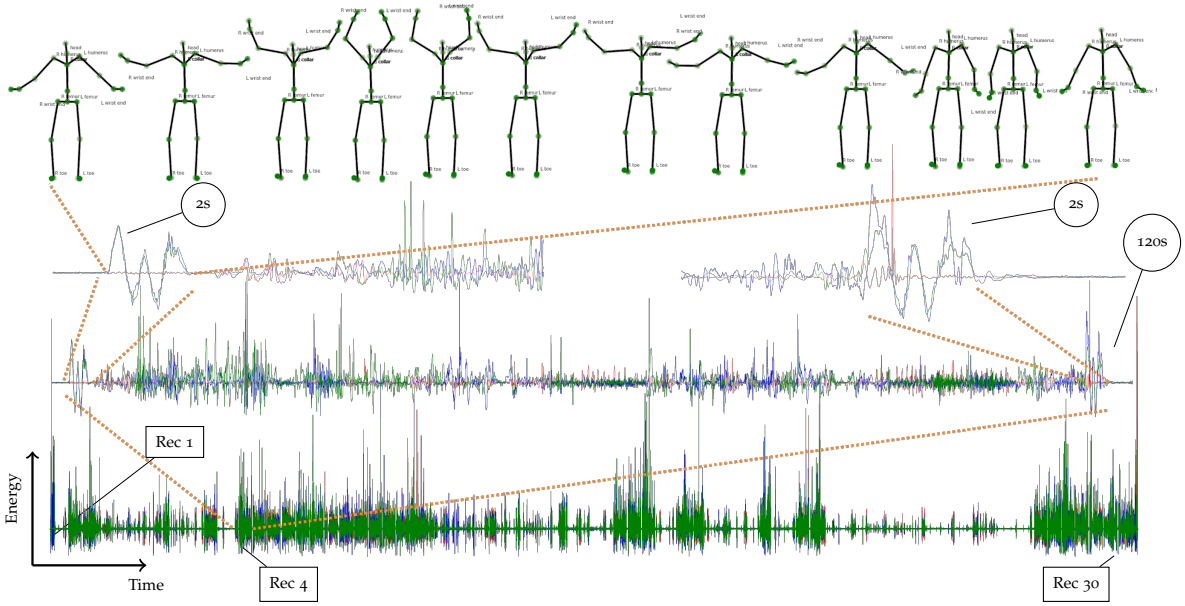


Figure 2.2.1: The author of this thesis in [NRMR+20][NRMR+22] synchronised 120s-long OBDs and marker-based MoCaps recordings using a *synchronisation* activity with an specific movement. The subject raises the arms up and down, keeping a standing position. The profile shows the energy of the 3D linear and angular acceleration channels from three OBDs, located on the right and left arm and on the belly. The authors shifted the recordings according to the starting and ending of the synchronisation activity per OBD. In this way, annotations based on marker-based MoCaps were used for annotating 30 recordings per subject in the LARa dataset.

The authors in [GD14, KKB14, OMR16, ZO18] normalised the extracted features before the training stage. For example, the authors in [ZO18] normalised the extracted features to the range of $[0, 1]$, and in [DBLG14] to $[-1, 1]$.

Synchronisation steps are required as sensors of OBDs are recorded asynchronously. The authors in [FMHF16, GLMR+17, NRMR+20] included a synchronisation activity, where the subject moves unstintingly with respect to the target activities. Annotators, then, mark the starting time of the synchronisation activity per sensor or channel of OBDs, as Figure 2.2.1 shows. Recordings are re-sampled according to a unique time using a piece-wise spline interpolation. The authors in [ZO18] ignored differences in recording sample times of devices, considering their target tasks, i.e., activities and durations. They considered only the recordings according to the time of the sensor recordings with the shortest ending time.

Typically, features are computed in each of the accelerometer directions independently, although in some cases, features that combine the axes are also used [TDF+18].

2.2.3 *Statistical Pattern Recognition for HAR*

Methods of statistical pattern recognition were used for supervised-HAR. These methods follow a standard pipeline: segmentation, extraction of handcrafted features, primitives computation or dimensionality of feature reduction, and classification. Usually, segmentation is carried out by employing a sliding-window approach on the sensor measurements along the time axis. Handcrafted features are statistical features extracted from segmented windows, either from the time or frequency domain. The handcrafted features should capture the intrinsic characteristics of a certain human action. Next, a feature reduction is deployed to reduce the dimensionality of the feature space, keeping the discriminant properties of the features. These features are aggregated using Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Kernel Discriminant Analysis (KDA) for dimensionality reduction and finally used for training a set of parameters of a classifier. These features are used for training a classifier in which its parameters are learnt. In the training stage, a classifier is trained by using the extracted features, or the reduced ones, and the ground-truth activity labels. Finally, the trained classifier assigns activity classes to unknown sequences from its extracted features. Examples of classifiers are Naïve Bayes (NB), Support Vector Machines (SVMs), Random Forest (RF), Dynamic Time Warping (DTW), and Hidden Markov Model (HMM) [FMHF16, OET+14]. To that end, annotated sensor measurements with specific activity class labels are necessary.

Segmentation

Segmentation extracts a sequence of continuous measurements of the pre-processed data that are likely to portray a human activity, rather than classifying every single data sample [RNMR+19, TDF+18]. In M-HAR, the sliding-window approach is a prevalent method for creating segments for being processed by a classifier. A window moves over the multi-channel Time-Series data by a given step, extracting or cutting segments. The window size directly controls the delay of the recognition system. The step size is selected according to segmentation precision—considering that short activities can be skipped or omitted—, computational effort and the number of training samples. The longer the window, the more information is used for predicting an activity; however, short-duration activities are skipped [TDF+18]. Using a short window length enables fast inference of the user's current activity and ensures the detection can rapidly adapt to changes [TDF+18]. The authors [RNMR+19] revised the commonly used recording rates, window sizes and steps. Recording rates are of the range 1 to 300] Hz, window sizes in the range of 0.67 to 25 s. and overlapping in the range of 5 to 75%. They noticed that the higher the sampling rate, the smaller the window size can be used for having more fine-grained predictions, e.g., small human limb movements. Publications using small sampling rates handle long-duration activities that can be seen as compositions of short-duration activities. Differently, there are approaches for segmenting sequences using additional measurements or events, e.g., eye movement, audio and speech [BBS14, CFP+16].

The authors in [SZ14, KY18, LPT18] used a number of F consecutive RGB and optical flow images for video-based HAR. The authors in [KY18] selected the F frames that follow the last frame where human joint pixel coordinates are all visible.

Feature Extraction

The feature extraction is an important stage of statistical pattern recognition pipeline. It allows representing data compactly being at the same time discriminative, which helps with later classification stages. A set of D functions $\mathbf{g}_d(\cdot) : \mathcal{X} \to \mathcal{Z}$ calculates a set $\mathbf{Z} = \{\mathbf{z}^{(d)}\}_{d=0}^{D}$ with D feature vectors from a segmented sequence $\mathbf{X} \in \mathcal{X}^{[W,S]}$, with W as the window size and S as the number of sensors, mapping the input to a feature space $\mathcal{Z}$. Typically, features are computed in each of the accelerometer directions independently, although in some cases, features that combine the axes are also used [TDF+18].

There exist two main groups, statistical and application-based features. Time-domain features on the measurement profiles and frequency-domain features focus on the periodic structure of the measurements [ZO18]. The DFT is applied to the raw- or preprocessed signals to acquire the estimated spectral density of the time series. These features are, for

example, the mean, variance, correlation, and the local slope of the sensor data fitted by first-order linear regression for the time domain or the entropy, energy, and coherence extracted from the Fourier Transform in the frequency domain [CGD+13, BBS14, FMHF16, TDF+18]. Application-based features refer to features that were created for a particular application or dataset. These features are based on geometric, structural and kinematic relations. The surveys from [RNMR+19, SAEM19] present a list of the most used statistical and application-based features.

Feature Reduction

The higher the dimensionality of the feature space, the more training data is needed for model parameter estimation and the more computationally intensive the classification is. Feature reduction seeks to minimize memory, computational power, and bandwidth requirements [BBS14], i.e., $\mathbf{h}(\cdot) : \mathcal{Z}^{R} \rightarrow \mathcal{Z}^{Q} \mid Q \ll R$. The authors in [AB10, ABT10, GW15, LYA09, VFD+17] deployed PCA for reducing the dimensionality of their features. PCA is a holistic method that considers its inputs as points in a high-dimensional feature space and it finds a lower-dimensional space along the highest variance of the features, where distance-based classification becomes easier—assuming that all the features in the lower space are not mutually correlated. LDA is another holistic method that tries to overcome that PCA does not consider the intra-class variation, i.e., any difference in classes. LDA finds an optimal projection of the input data to a linear subspace with directions, which maximise the ratio between the inter- and intra-class variations of a set of features. Quadratic Discriminant Analysis (QDA) was deployed by [SR12], which is similar to LDA assuming that the class-conditional densities are normally distributed, and the covariance of all the classes is equal. The authors in [BBS14, KLLK10] used KDA, which is a non-linear discriminating approach based on kernel techniques to find non-linear discriminating features. The authors in [ZO18] followed the Recursive Feature Elimination (RFE) for finding the best set of features. The RFE can be seen as a dense parameter search, which iteratively selects or rejects a set of features after training and deploying a classifier. The authors in [DLGY12] proposed a BagFR using k-means clustering. They used the k-means clustering algorithm to partition the multi-channel Time-Series sequences of human poses into M motion clusters. Additionally, the authors in [GW15, CPR11] utilized a Random Projection (RP). The authors in [AB10, ZO18] used a Sequential Forward Feature Selection (SFFS) and Sequential Backward Feature Selection (SBFS), which respectively add or delete features, one at a time, evaluating the classification performance. The authors in [TDF+18, ZJC17] deployed dictionary learning or sparse coding for performing feature selection. They learn a set of dictionary vectors such that a set of features $\mathbf{Z}$ can be represented as a linear combination of these vectors, finding a basis similar to PCA.

However, they introduce an additional sparsity constraint setting some of the parameters of the basis to zero.

Classifiers

Let us define the term *task* for HAR, following [CFK13, PY10].

Definition 2. ***Task:*** *For HAR, a task $\mathcal{T}$ is a tuple $(\mathcal{Y}, \mathbf{f}(\cdot))$ with $\mathcal{Y}$, a label space from a set of Y activity classes for some give domain $\mathcal{D}$, and an objective predictive function or conditional probability distribution $\mathbf{f}(\cdot)$ consisting of the probability of assigning a label $y_i \in \mathcal{Y}$ given the observed sample $\mathbf{X}_i \in \mathcal{X}$, i.e., $\mathbf{p}(\mathcal{Y}|\mathcal{X})$. The function $f(\cdot)$ is not given, but can be learned from the annotated training data, consisting of N tuples $\{(\mathbf{X}, y)^n\}_{n=0}^N$ where $\mathbf{X}^n \in \mathcal{X},\ y^n \in \mathcal{Y}$.*

The classifier $\mathbf{f}(\cdot)$ maps the extracted features $\mathcal{Z}^R$ or the lower-dimensionality ones $\mathcal{Z}^Q$ to a class representation $\mathcal{Y}$, i.e., $\mathbf{f}(\cdot) : \mathcal{Z}^R \rightarrow \mathcal{Y}$. Its parameters are trained using the extracted features of an annotated dataset, with N annotated samples $\{(\mathbf{X}, y)^n\}_{n=0}^N$ from a source domain $\mathcal{D}_{source}$, minimizing the classification error. Finally, for new samples $\mathbf{X}_{new} \in \mathcal{X}$, classifiers can be classified into template matching techniques, generative approaches and discriminative approaches.

Template matching techniques employ a K-Nearest Neighbour (KNN) classifier using a Euclidean distance [AB10, ABT10, BBS14, GW15, GD14, KKB14, STBO17, SR12, ZO18] or DTW [ABT10, MHB+16]. A KNN classifier is a non-parametric classifier that matches the activity class of the k nearest feature representation to the ones of the $\mathbf{X}_{new}$ sample.

Generative probabilistic graphical models such as HMM model and dynamic Bayesian have been used for modelling multi-channel Time-Series sequences and for smoothing recognition results of an ensemble classifier. The authors in [LC11] trained an HMM for each dimensional axis $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ of pre-processed acceleration measurements, fusing them with a weighted sum. The authors in [KEK08, BBS14, FK16, GCBCJG14] also deployed HMMs, and the authors in [WGT+11] used coupled HMMs, where a two-chain coupled HMM connects hidden states of two sequence observations from two subjects performing an activity. The authors in [RC17] used hierarchical conditional HMMs. The authors in [AB10, BBS14, CPR11, FMHF16, GW15, GD14, LPLP12] used Naïve Bayes (NB). NB models activity samples using a Gaussian Mixture Model (GMM). The authors in [LYA09] deployed a NB with a Probability Density Function (PDF) from 19 PCA-based reduced features. The authors in [ZO18, VFD+17] also used an NB, assuming that the features are mutually independent.

Discriminative approaches including SVMs [AB10, BPT14, CPR11, CX15, FMHF16, GW15, STBO17, ZO18] and Conditional Random Fields (CRFs) [GCBCJG14] have been also effective for M-HAR. Discriminative approaches minimize the error by gradient

descent. An SVM finds the hyperplane $\mathbf{w}^T\mathbf{x}_i + b$ that maximizes the margin beween the data points of different classes by optimizing the following Quadratic Programming Problem (QPP), Equation 2.2.1.

$$f(x) = \min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i, \tag{2.2.1}$$

such that $\mathbf{a}_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i$ and $\xi_i \geqslant 0$, $\forall i$, where $\mathbf{x}_i$ and $\mathbf{a}_i$ are the feature vector and the attribute vector for the i-th training sample, respectively.

The authors in [CSG+13] have trained a SVM classifier per attribute $\mathbf{a} \in \mathcal{A}$; see Section 2.4 for more details about attribute representation. The authors in [AI15] used an SVM-based binary decision tree classifier. Anguita et al. [AGO+12] proposed a hardware-friendly SVM that is meant to be deployed on smartphone devices. The authors in [VFD+17] used a Dynamic Bayesian Mixture Model (DBMM) for combining conditional probability outputs from different base classifiers, namely, an NB, anSVM, and an MLP. A weight is assigned to each base classifier, according to the learning process, using an uncertainty measure as a confidence level. An MLP, Section 2.1,was deployed by [AB10, ABT10, BPT14, CPR11, KLLK10, KWM11, LPLP12, SSH13].

Additionally, other classifiers, methods or approaches were used: Random Forest (RF) [BPT14, CPR11, FMHF16, TDF+18], Decision Tree (DT) [GW15, KWM11, LPLP12, SSH13, ZO18], Logistic Regression (LR) [BPT14, KWM11, SSH13, TDF+18], Least Squares Method (LSM) [AB10], GMM [KKB14], Template Maching (TM) [MHB+16], Correlation [MHB+16], KNN with Euclidean Distance (ED) [MHB+16].

In addition, the authors in [BPT14] combined single classifiers by averaging the activity probability predictions and majority voting. Joint Boosting (JB) [ZSMP15], Bagging [ZSMP15, LPLP12] and Stacking [ZSMP15] were also deployed.

2.3 TRANSFER LEARNING FOR HAR

A unique M-HAR framework does not apply to many real-world scenarios. Frameworks adapt baseline M-HAR methods to new scenarios [CFK13, KSR+21b]. Traditional M-HAR methods make strong assumptions that the training and testing data are drawn from the same distribution; data is in the same space, e.g., very specific sensor types. Real-world applications do not follow this assumption. Supervised M-HAR methods, e.g., deep learning methods, need large annotated and curated datasets for performing decently in a testing scenario in a variety of circumstances. In the case of OBD data, data is scarce, as the annotation process is expensive, time-consuming, tedious, and requires domain expertise,

see Chapter 7. The lack of annotated datasets raises the need to create alternatives for using, re-using and making the most of the existing annotated datasets, finding relations among datasets or performing Transfer Learning for HAR [CFK13]. Adapting experience is the capability of humans to identify deep, subtle connections in data; this experience adaptation is called transfer learning. Frameworks that adapt experience from different sources are required to improve performance on new tasks without data creation or with a minimal fraction of it [CFK13]. Transfer learning is the ability to extend the experience gained from a domain to new domains [CFK13, GY17, KSR+21b].

Transfer Learning for HAR has helped cope with the limited amount of annotated data, overfitting, and the class-imbalance problem, as classes are not normally distributed. A TL method for HAR should work under different scenarios. It should be able to re-use annotated data from a source scenario into a target scenario, with the same number and type of sensors. The target scenario has a different layout and different subjects and sensor locations, e.g., different logistic scenarios using the same devices for capturing activities of workers or intelligent houses using similar devices for elderly care. Nevertheless, a method of Transfer Learning for HAR shall also transfer information for different physical settings, sensor locations, sensor types, subjects—cultural changes, age-related differences, and gender—and activities. Besides, the amount of annotated data is also a factor to consider. Transfer Learning for HAR shall consider the representation of the experience or knowledge to be transferred [CFK13]. The authors in [CFK13] analysed Transfer Learning for HAR from four dimensions: sensor modalities, source and target domain differences, amount of annotated data in the source and target datasets, and the representation or the type of information being utilised.

Following the definition of a M-HAR domain in Subsection 2.2.1, and of a M-HAR task in Subsection 2.2.3, and [PY10, CFK13], let us define the term Transfer Learning for M-HAR.

Definition 3. ***Transfer Learning for M-HAR:*** *given a set of* Q *number of source domains* $\left\{\mathcal{D}_{source}^{(q)}\right\}_{q=0}^{Q}$*, e.g., different logistics scenarios with different OBD settings, a set of* Q *number of source tasks* $\left\{\mathcal{T}_{source}^{(q)}\right\}_{q=0}^{Q}$ *where* $\mathcal{T}_{source}^{(q)}$ *corresponds with* $\mathcal{D}_{source}^{(q)}$*; a target domain* $\mathcal{D}_{target}$*, e.g., a new logistic scenario, with a corresponding target task* $\mathcal{T}_{target}$ *transfer learning helps improving the learning of the target predictive function* $\mathbf{f}_{target}(\cdot)$ *in* $\mathcal{T}_{target}$*, where the multi-channel Time-Series spaces* $\chi_{target} \neq \chi_{source}$ *and the label spaces* $\mathcal{Y}_{target} \neq \mathcal{Y}_{source}$ *differ.*

This definition of transfer learning serves to describe many different transfer learning scenarios. These scenarios appear according on the differences between the source and the target domains in an specific application of M-HAR, e.g., for example the close-to-reality,

laboratory LARa dataset [NRMR+20] compared to the real MotionMiners datasets for intralogistics. These transfer learning scenarios can be defined:

- Source and Target sensor spaces differ:
  - Q number of source domains and a target domain have different multi-channel Time-Series spaces, $\mathcal{X}_{target} \neq \left\{\mathcal{X}^{(q)}_{sources}\right\}^{Q}_{q=0}$, e.g., under different OBDs, sensor modalities or physical space.
  - a different underlying marginal probability distribution in the feature space $\mathbf{p}(\mathbf{X}_{target}) \neq \left\{\mathbf{p}(\mathbf{X}^{(q)}_{source})\right\}^{Q}_{q=0}$, when $\mathcal{X}_{source} = \mathcal{X}_{target}$, e.g., under different recordings settings with varying time, subjects, or sampling rates.
- Source and Target tasks differ under:
  - a different label space $\mathcal{Y}_{target} \neq \left\{\mathcal{Y}^{(q)}_{source}\right\}^{q}_{q=0}$.
  - a different predictive function for labels $\mathbf{f}_{target}(\cdot) \neq \left\{\mathbf{f}_{source}(\cdot)^{(q)}\right\}^{q}_{q=0}$ under the same label space $\mathcal{Y}$, e.g., differences due to time, subjects, devices, sampling rates, activities, or labels.
- Source and Target domains differ, i.e., $\mathcal{D}_{source} \neq \mathcal{D}_{target}$.

Nevertheless, transfer learning assumes some relationship between the source and target tasks, which allows for the successful transfer of knowledge. The transferability becomes more challenging as the number or type of differences between the source and the target domains increase. In this case, transferability is much more difficult.

Transfer learning can also be distinguished depending on the availability of annotated data, either for the source and target domains, following [CFK13, PY10]. Informed Supervised Transfer Learning (IS Transfer Learning) implies that N annotated samples $\{\mathbf{X}, \mathbf{y}\}^{N}_{n=0}$ from the source domain $\mathcal{D}_{source}$ and M annotated samples $\{\mathbf{X}, \mathbf{y}\}^{M}_{m=0}$ from the target domain $\mathcal{D}_{target}$ are available. Uninformed Supervised Transfer Learning (US Transfer Learning) implies that N annotated samples are available only in the source domain $\mathcal{D}_{source}$. Informed Unsupervised Transfer Learning (IU Transfer Learning) implies that only M annotated samples $\{\mathbf{X}, \mathbf{y}\}^{M}_{m=0}$ from the target domain $\mathcal{D}_{target}$ are available. Uninformed Unsupervised Transfer Learning (UU Transfer Learning) implies that no annotated data is available for either the source or target domains [CFK13].

Considering that *Inductive learning* refers to learning techniques to learn the objective predictive function $\mathbf{f}(\cdot)$; Inductive Transfer Learning (ITL) seeks to learn $\mathbf{f}(\cdot)_{target}$ using data from a source domain, $\mathcal{D}_{source}$. It requires annotated data from the target domain

$\mathcal{D}_{target}$, even for few samples, without considering annotations of the source, $\mathcal{D}_{source}$. Here, the source and target domains are equal $\mathcal{D}_{source} = \mathcal{D}_{target}$, but their tasks differ $\mathcal{T}_{source} \neq \mathcal{T}_{target}$. Furthermore, considering that *Transductive Learning* refers to the situation where all test data are required to be seen at training time, and the learned model cannot be reused for future data. For deployment, the classifier cluster the entire training and testing data. Transductive Transfer Learning (TTL) techniques try to learn $\mathbf{f}(\cdot)_{target}$ via the relationship between samples $\mathbf{X}_{source} \in \mathcal{D}_{source}$ and $\mathbf{X}_{target} \in \mathcal{D}_{target}$. Here, the source and tasks are the same $\mathcal{T}_{source} = \mathcal{T}_{target}$, but the domains differ, $\mathcal{D}_{source} \neq \mathcal{D}_{target}$. This transfer approach does not require annotated data in the target domain, however, target data shall be available at training [CFK13, PY10].

Inductive Learning Transfer Learning for HAR can be classified into four types, namely, instance transfer, feature-representation transfer, parameter transfer, and relational-knowledge transfer, according to [PY10], as explained below.

2.3.1 *Instance Transfer*

Instance transfer reuses the source data $\{(\mathbf{X}, y)^n\}_{n=0}^{N}$ to train the target classifier. Usually, instance transfer weights the source samples based upon a given metric. It assumes that certain parts of the data in the source domain can be reused for learning the target domain by reweighting. It works well when the source and target domains are the same $\mathcal{D}_{source} = \mathcal{D}_{target}$, i.e., equal number and types of OBDs and the locations on the human body, for example, the different logistic scenarios using the same set of OBDs.

The authors in [ZHY09] used an instance-based approach to weight source instances based upon the similarity between the activity classes of the source and target data. With this, they transferred the labels from samples in the source domain to samples in the target domain using web knowledge—they used web search to extract related web pages for the activities and then applied information retrieval techniques to process the extracted web pages, computing the cosine similarity among word-vectors of activity classes—the cosine similarity results from the angle between two vectors, being 0 for orthogonal vectors and 1 otherwise; see Equation 4.1.1. Such similarities will be used later to propagate labels for domain transfer to relate the two domains. They are referred to as cross-domain transfer learning; however, this is for the case when domains of source and target are the same, having the same set of sensors, but their tasks differ $\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$, e.g., zero-shot learning.

The authors in [HSU12] developed an importance weighted least-squares probabilistic classification approach to handle Transfer Learning for M-HAR when $\mathbf{p}(\mathbf{X}_{source}) \neq \mathbf{p}(\mathbf{X}_{target})$ and $\mathbf{f}_{source}(\cdot) \neq \mathbf{f}_{target}(\cdot)$, i.e., the marginal distributions of the source and

target domains differ but their conditional distribution over the label space $\mathcal{Y}$ remains unchanged. They formulated this problem as a covariate shift problem. They focused on the sampled reweighting approach and proposed a new probabilistic classification method that is computationally very efficient. This approach combines a probabilistic classification method called least-squares probabilistic classifier with the sample reweighting approach.

2.3.2 *Feature-Representation Transfer*

Feature-representation transfer reduces the differences between the source $\mathcal{Z}_{source}$ and target $\mathcal{Z}_{target}$ feature representations. The basic idea is to learn a low-dimensional representation that is shared across related tasks, finding a good representation for the target domain [PY10]. The knowledge used for transfer across domains is encoded into the learned feature representation. It maps the source feature space to the target feature space, such as $\mathbf{k_f}(\cdot) : \mathcal{Z}_{source} \rightarrow \mathcal{Z}_{target}$, the target feature space to the source feature space, such as $\mathbf{k_g}(\cdot) : \mathcal{Z}_{target} \rightarrow \mathcal{Z}_{source}$, and the source and target feature spaces to a common feature space such as $\mathbf{k_k}(\cdot) : \mathcal{Z}_{target} \rightarrow \mathcal{Z}$ and $\mathbf{k_k}(\cdot) : \mathcal{Z}_{source} \rightarrow \mathcal{Z}$. This mapping can be computed manually [KEK08] mixing recordings from devices in similar places or for different purposes—this is for HAR using ambient sensors—or learned as part of the transfer learning algorithm [HY11]. This transfer approach works well when a lot of annotated data in the source domain is available, where supervised learning can be used to construct a feature representation.

The authors in [HY11] projected samples of the target set $\{\mathbf{X}, y\}_{target}$ to pseudo-classes corresponding to the source task $\mathcal{T}_{source}$. They assigned a pseudo-class $\mathcal{Y}_{source}$ from the most frequent source activity classes in a set of K recordings $\{\mathbf{X}^k\}_{k=1}^{K} \in \mathcal{X}_{source}$ similar to the target one. The K recordings are computed based on the DTW score between the source recordings and the target recordings. The authors map these pseudo-classes to the target activity classes measuring the distance between the source and the target activities using the Google Similarity Distance. This distance relates two activity classes based on the frequency of appearance of both activities on a given number of web pages.

Instance transfer, Subsection 2.3.1, might also be applied after mapping the feature representations from source, and target datasets to a common representation [CFK13].

2.3.3 *Relational-knowledge Transfer*

Relational-knowledge transfer requires that there exist certain relationships in the data, which can be learned and transferred across subjects. Data for activity recognition have the potential to contain such transferable relationships indicating that this may be an

important technique to pursue. Furthermore, relational-knowledge transfer applies to problems in which the data is not Independent and Identically Distributed (i.i.d.) as is traditionally assumed but can be represented through multiple relationships. The works of [ZHY09, HY11] related activity classes based on the semantic definition of the classes, the cosine similarity of word-vectors of two activity classes, and the Google Similarity Distance measuring the frequency of appearance in a given number of web pages. Subsection 3.4.2 presents a relational-knowledge method for M-HAR.

2.3.4 *Parameter Transfer*

Learned parameters are shared among the source $\mathcal{D}_{source}$ and target domains $\mathcal{D}_{target}$. For example, a common parameter transfer is learning a prior distribution shared between the source and target datasets. In another example, one technique models the source and target tasks using a GMM, which share a prior distribution. The work in [KEK08] proposed a method to learn the parameters of an HMM using labelled data from the source domain and unlabeled data from the target domain. In addition, the authors in [KEK10] extended [KEK08] learning hyperparameter priors for the HMM instead of learning the parameters directly. Differently, the authors in [DXTL10] learned a target classifier by merging a set of pre-trained classifiers from two source domains prior to learning a robust target classifier with a portion of annotated data from the target domain.

Another approach for transferring knowledge among domains is Zero-shot learning. Zero-shot learning is a classification task where the source and target tasks are disjoint, $\mathcal{T}_{source} \neq \mathcal{T}_{target}$, but have equal input space $\mathcal{D}_{source} = \mathcal{D}_{target}$, i.e., the target tasks are not directly learned. It addresses problems where the task space is large, i.e., comprising hundreds to thousands of classes, or samples of the target tasks are hard to obtain [AGFV14, LNH14, XLSA19]. A particular approach for Zero-short learning is the usage of attribute representations[6], and it is specially focused in this work for addressing HAR. Henceforth, Section 2.4 presents in more detail attribute representations for Zero-shot learning. Then, Chapter 4 deepens on attribute representations along with deep learning and finally links it for M-HAR. Finally, Subsection 3.4.1 presents parameter transfer methods using DNNs for HAR.

2.4 ATTRIBUTE REPRESENTATION FOR CLASSIFICATION

Attribute representations have helped to solve transfer learning in cases where the source and target tasks differ, e.g., for zero-shot learning; here, the source and target

6 Attribute Representation can be sorted as Feature-Representation Transfer, Subsection 2.3.2.

domains usually are equivalent. For example, zero-shot learning for object recognition [LNH09, LNH14], scene recognition and word spotting [SF18], where the domain is RGB images. Besides, using attribute representations helps in cases where annotations are hard to obtain [CGD+13], where data is highly imbalanced, and its class space is large, e.g., words of a language; see Chapter 7 for more details. High-level attribute representations are semantic descriptions that have been deployed for describing categories in object and scene recognition [LNH14, ZJC17] and in document analysis [AGFV14, SF18].

Objects are identified based on high-level descriptions, called visual semantic attributes, e.g., objects' colour or shape. Generally, visual attributes are mid-level semantic properties of categories, which are shared among categories. For example, visual attributes can be furry, yellow, or four-legged for object recognition [LNH09, LNH14], or word characters for handwritten word spotting [SF18]. They provide an effective way of solving the zero-shot classification problem [SFLP17].

Classifiers, e.g., DNNs, demand a large amount of annotated samples per class in order to provide robust predictions for new samples. However, annotating samples for all the wanted categories demands a lot of effort and resources, especially for the infrequent classes, such as specific dog breeds, car models, or articles in stores. Besides, it is not an easy task to classify classes that are not present in the training set, e.g., zero-shot classification as mentioned in Subsection 2.3.4.

There are categorisation tasks for which few samples are available. For example, for object recognition, the world contains thousands of different object classes, and image collections contain only a few of those classes [LNH14]; for handwritten word spotting or word recognition, it is unfeasible to collect training samples for every word.

Attributes transcend a specific learning task; an attribute classifier can be pre-learned independently from a source dataset unrelated to the current task. That means transfer learning via attribute representations allows for classifying new classes without the need for a new training phase.

Given an arbitrary domain $\mathcal{D}$, and two corresponding tasks $\mathcal{T}_{source}$ with $(\mathcal{Y}, \mathbf{f}(\cdot))_{source}$ and $\mathcal{T}_{target}$ with $(\mathcal{Y}, \mathbf{g}(\cdot))_{source}$, having $\mathcal{Y}_{source}$ and $\mathcal{Y}_{target}$ the source and target class spaces, and both tasks are disjoint $\mathcal{T}_{source} \neq \mathcal{T}_{target}$. The idea is to learn the objective predictive function $\mathbf{g}(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}_{target}$, which maps samples $\mathbf{X} \in \mathcal{X}$ to $\mathcal{Y}_{target}$ by using annotated samples of the source dataset $\{(\mathbf{X}, y)^n\}_{n=0}^N$ even if $\mathcal{T}_{source} \neq \mathcal{T}_{target}$. There is a coupling between classes in $\mathcal{Y}_{source}$ and $\mathcal{Y}_{target}$ to classify unseen classes, or classes for which no annotated data is available. This falls into our definition of transfer learning.

We assume that there is no-annotated data for the unseen dataset. The coupling cannot be learnt from samples, so it has to be inserted by humans. Ideally, the amount of human effort to specify new classes should be small; otherwise, collecting and labelling training samples might be a more straightforward solution.

The authors in [LNH14] proposed a solution of learning with disjoint training and testing classes for object recognition by introducing a small set of high-level semantic attributes that can be specified either on a per-class or on a per-image level. An attribute is a property of an object for which a human can decide whether the property is present or not for a certain object. The work from [LNH14] considered only binary-valued attributes, i.e., present or not. In contrast, image features are computable, but humans cannot interpret them.

Attributes will be denoted as $\mathbf{a} \in \mathcal{A}$ and binary attributes as $\mathcal{A} \subset \mathbb{B}$. They are distinguishable and nameable properties, e.g., the colour of an object. The authors in [LNH14] also considered properties that are not necessarily visible but are contextually related to visual information of the object or class, e.g., an animal's natural habitat.

Attributes can be assigned to every image sample or to every object class. Assigning the attributes directly to classes is particularly helpful for annotating the attributes with minimal effort. In addition, coupling data via only *common knowledge* is preferable over specialised expert knowledge, because the latter is often difficult and expensive to obtain. The authors [LNH14] proposed the *attribute-based object classification*.

Having the transfer learning scenario of learning with disjoint training and testing classes in Section 2.3, if for each class in $\mathcal{Y}_{source}$ and $\mathcal{Y}_{target}$ and attribute representation $\mathbf{a} \in \mathcal{A}^M$ is available, then a non-trivial classifier $\mathbf{g}(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}_{target}$ can be learned by transferring information between $\mathcal{Y}_{source}$ and $\mathcal{Y}_{target}$ through $\mathcal{A}$. The [LNH09, LNH14] proposed two attribute-based predictions approaches, namely, Direct Attribute Prediction (DAP), Indirect Attribute Prediction (IAP). They propose a probabilistic model that reflects the attribute representation classification, as Figure 2.4.1 shows. They assumed the attributes as being binary $\mathcal{A} \in \mathbb{B}$. The attribute representation is $\mathbf{a}^{(y)} = (a_m^{(y)}, ..., a_M^{(y)})$ for any training class y are M-fixed-length binary vectors.

2.4.1 *Direct Attribute Prediction (DAP)*

The authors in [LNH09, LNH14] proposed the Direct Attribute Prediction (DAP) for object classification. The DAP uses an intermediate layer of variables to decouple the image samples from the classes. These variables are related to semantic information or attributes from the classes, i.e., there is a deterministic relation from class labels to attributes. Having this relation, a supervised learning method can be used for learning per-attribute parameters, e.g., the parameters of a network for computing attribute representation. At testing, these allow the prediction of attribute values for each test sample, from which the test class label is inferred. Note that the classes during testing can differ from the classes

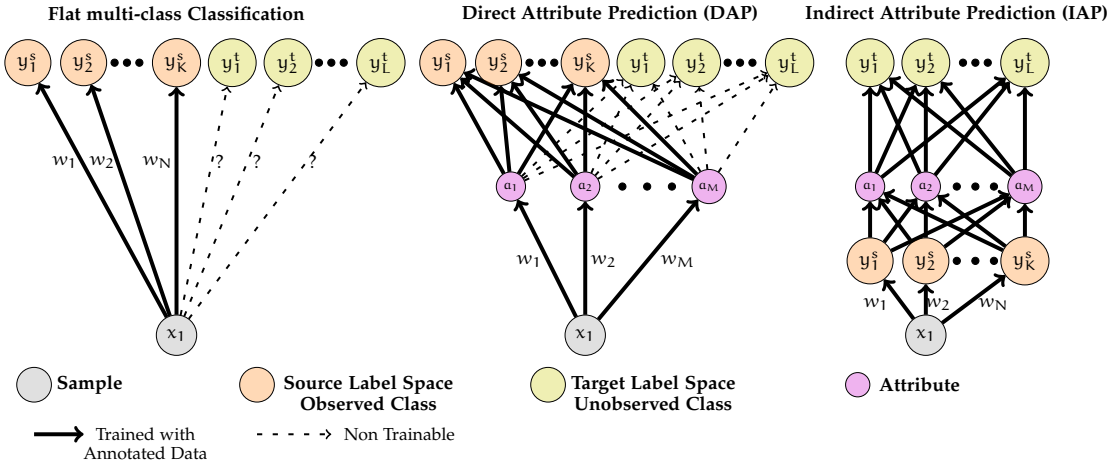


Figure 2.4.1: Graphical representation of the Direct Attribute Prediction (DAP) and Indirect Attribute Prediction (IAP) for object classification. The flat multi-class classification cannot be generalized for computing the target activities $y^t \in \mathcal{Y}_{target}$. For the DAP, the parameters of an attribute classifier $\boldsymbol{W}$ are learned using the annotated source dataset $\{(\mathbf{X}, y)^n\}_{n=0}^N$. The target activities $y^t \in \mathcal{Y}_{target}$ are inferred using the predicted attribute representation $\mathbf{a}$ using Equation 2.4.4. For the IAP, the parameters of an activity classifier $\boldsymbol{W}$ are learned using the annotated source dataset $\{(\mathbf{X}, y)^n\}_{n=0}^N$. The attributes $\mathbf{a}$ are determined deterministically, $p(\mathbf{a}|y^s) = [[\mathbf{a} = \mathbf{a}^{(y^s)}]]$. The target activities $y^t \in \mathcal{Y}_{target}$ are inferred using the predicted attribute representation $\mathbf{a}$ using Equation 2.4.5. Image adapted from [LNH09, LNH14].

used for training, as long as the coupling attribute layer is determined in a way that does not require a training phase.

For DAP, the authors learn probabilistic classifiers for each attribute a_m. They use all images from all training classes as training samples with their class determined by the entry in an attribute representation matrix. This matrix relates an attribute with a corresponding class, i.e., a sample of class y is assigned the binary label a_m^y. An attribute predictor model provides the estimates $p(a_m|\mathbf{X})$. A model for the complete image-attribute layer provides $\mathbf{p}(\mathbf{a}|\mathbf{X}) = \prod_{m=1}^M p(a_m|\mathbf{X})$.

On the testing scenario, every class $y^t \in \mathcal{Y}_{target}$ induces its attribute vector $\mathbf{a}^{(y^t)}$ in a deterministic way, i.e., $p(\mathbf{a}|y^t) = [[\mathbf{a} = \mathbf{a}^{(y^t)}]]$[7]. Applying Bayes's rule, the relation between

7 Iverson´s braket notation $[[P]] = 1$, if the condition P is true or $[[P]] = 0$ otherwise.

the y^t and the attribute vector $\mathbf{a}$ is obtained as $\mathbf{p}(y^t|\mathbf{a}) = \frac{p(y^t)}{p(\mathbf{a}^{(y^t)})}[[\mathbf{a} = \mathbf{a}^{(y^t)}]]$, known as the attribute-class layer.

$$p(y^t|\mathbf{X}) = \sum_{\mathbf{a}\in\mathbb{B}^M} p\left(y^t|\mathbf{a}\right) p(\mathbf{a}|\mathbf{X}) \tag{2.4.1}$$

$$p(y^t|\mathbf{X}) = \sum_{\mathbf{a}\in\mathbb{B}^M} p(z|\mathbf{a})p(\mathbf{a}|\mathbf{X}) = \frac{p(z)}{p(\mathbf{a}^{(y^t)})} \prod_{m=1}^{M} p\left(a_m^{(y^t)}|\mathbf{X}\right) \tag{2.4.2}$$

$$p(\mathbf{a}^{(y^t)}|\mathbf{X}) = \prod_{m=1}^{M} p(a_m^{(y^t)}|\mathbf{X}), \tag{2.4.3}$$

with $\mathbf{X}$ being an input image sample, and $\mathbf{a}^{(y^t)}$ the binary attribute vector of class y^t. Besides, the factor $p((y^t))$ is ignored assuming identical class priors. For the factor $p(\mathbf{a})$, [LNH14] assume a factorial distribution $p(\mathbf{a}) = \prod_{m=1}^{M} p(a_m)$, using the empirical means $p(a_m) = \frac{1}{K}\sum_{k=1}^{K=Y_{source}} a_m^{y_k}$ over the Y_{source} training source classes as attribute priors—the prior $p(\mathbf{a})$ is not crucial to the procedure in practice, and setting $p(a_m) = \frac{1}{2}$ yields comparable results. As decision rule $\mathbf{g} : \mathcal{X} \to \mathcal{Y}_{target}$ that assigns the best output class from all $L = Y_{target}$ target classes $z_1, ..., z_L$ to a test sample X, they use MAP prediction:

$$\mathbf{g}(x) = \underset{l=1,...,L}{\operatorname{argmax}} \prod_{m=1}^{M} \frac{p(a_m^{z_l}|\mathbf{X})}{p(a_m^{z_l})} \tag{2.4.4}$$

2.4.2 *Indirect Attribute Prediction (IAP)*

The Indirect Attribute Prediction (IAP) uses attributes to transfer knowledge between classes but uses an attribute layer from a connecting layer between two layers of activity classes, the source and the target tasks. The training phase of IAP is a standard multi-class classification. At testing, the predictions for all target classes induce an attribute representation, from which predictions over the target classes can be inferred.

In order to implement IAP, they only modify the image-attribute stage: as first step, they learn a probabilistic multi-class classifier estimating $p(y_s|\mathbf{x})$ for all training source

classes $y^s \in \mathcal{Y}_{source}$. Again assuming a deterministic dependence between attributes and classes, they set $p(a_m|y) = [[a_m = a_m^{(y)}]]$. The combination of both steps yields,

$$p(a_m|\mathbf{X}) = \sum_{k=1}^{K=Y_{source}} p(a_m|y_k^s)p(y_k|\mathbf{X}) \tag{2.4.5}$$

Thus, inferring the attribute posterior probabilities $p(a_m|\mathbf{X})$ requires only a matrix-vector multiplication with expert given $p(a_m|y_k^s)$. After computing $p(a_m|\mathbf{X})$, the activity class is inferred using Equation 2.4.4, in the same way as in for DAP.

The authors in [LNH09, LNH14] evaluated the DAP and IAP for classification of animal images, introducing the *Animals with Attributes (AwA)* dataset. They deployed non-linear SVM for each attribute a_m. Here, the attributes are given per class and not per sample.

This chapter presents the essential aspects of a M-HAR system. Besides, it covers the fundamentals of DL, starting from the perceptron, MLP, activation functions, and training procedure, and the relevant elements of DNNs. Finally, the chapter focuses on Transfer Learning for HAR, driving into the concept of attribute representations.

Following, Chapter 3 further looks into methods of HAR using DL approaches. Besides, it will present parameter transfer and relational transfer approaches using DL. Furthermore, Chapter 4 will regard related works combining attribute representations CNNs and applied on HAR.

DEEP LEARNING FOR HUMAN ACTIVITY RECOGNITION

3

DNN have been used successfully for solving M-HAR problems. In comparison to standard statistical M-HAR, presented in Subsection 2.2.3, DNNs conveniently combine the feature extraction and classification in an end-to-end approach. This chapter revises related works concerning M-HAR, video-based HAR and pose-based HAR using DNNs; specifically, CNNs and tCNNs, RNNs, and transformers.

The performance of M-HAR on OBDs using DNN, however, has not shown a significant increase in performance as in other HAR fields, such as video-based HAR. Considering M-HAR as a supervised problem, scarcity of annotated M-HAR data is the primary concern [KHC10, DPBR20]. Supervised DNN require a large amount of annotated data. Transfer learning, introduced in Section 2.3, can alleviate the problem of scarcity of annotated data. Nonetheless, it can be hindered by the enormous variation of recording settings, e.g., different recording rates, sensor resolutions, device position, or intrinsic device characteristics.

This chapter deepens into transfer learning specifically for M-HAR. For that, the chapter presents the concept of synthetic data generation, which was introduced in video-based HAR and pose-based HAR methods concerning human pose estimation in Section 3.2. This is relevant for discussing a transfer learning alternative for OBD. Further, it develops the parameter- and relational-knowledge transfer learning approaches for M-HAR, presented in Subsection 2.3.4 and Subsection 2.3.3, respectively.

3.1 DEEP NEURAL NETWORKS FOR M-HAR

DL methods are prevalent for HAR in the context of M-HAR for the applications of gesture recognition and ADLs. Compared to the statistical pattern recognition methods, DL methods combine the feature extraction, feature or dimensionality reduction, and classification in a holistic approach [ZNY+14, RC15, YNS+15, HHP16, OR16, MSR+17, YLSR18]. Their features are directly learned from data, being mode discriminative. Besides, they overcome some problems regarding the computation and adaptability of handcrafted features.

TCNNs are end-to-end architectures that combine learnable temporal convolutional filters along the time axis with non-linear operation functions, downsampling and classification. A temporal convolution is computed following Equation 2.1.38. These architectures map a sequence segment of multi-channel Time-Series recordings into a class representation. These architectures learn the non-linear and temporal relations of basic, complex and highly dynamic human movements. By stacking convolutional layers, and downsampling their outputs, tCNNs extract more complex and abstract features and are task-dependent, being invariant to distortions and time translations [LKF10]. tCNNs extract hierarchical human body movements, i.e., from basic and simple movements to complex ones. Moreover, they learn the temporal dependencies among different movements. They learn non-linear transformations directly from raw inertial data. These transformations are more discriminative with respect to the activity classes than the handcrafted features. These features are also invariant to distortions and temporal translations [HHP16]. Different configurations of such networks have been introduced, as Figure 3.1.1 shows.

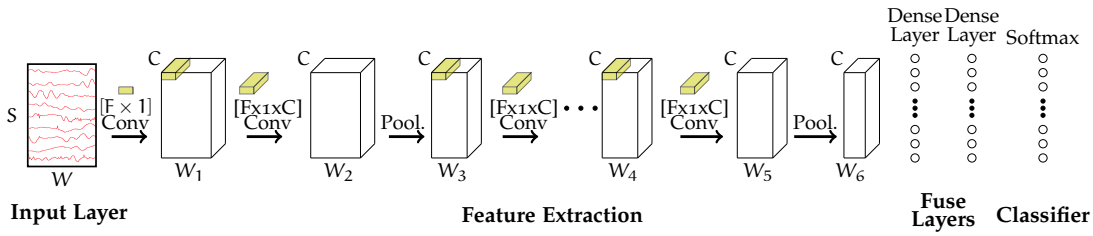


Figure 3.1.1: Different configurations of tCNNs in the literature are mainly divided into four parts: an input layer, a feature extractor block, fusion layers, and a classifier. Authors in the literature have investigated different numbers of convolutional and max-pooling layers, the number of filters per layer C, the size of the convolutional filter F, and fuse layers. The latter consists of either FC or LSTM layers.

For M-HAR, the inputs consist of a stack of N segmented sequences $\mathbf{X} \in \mathbb{R}^{[W,S]}$ from S sensors, for a certain temporal duration W. These windows are extracted, for example, using a sliding-window approach with window size W and stride Str, as Subsection 2.2.3 shows. These sequence inputs are of size $[W, S]$. Using a small Str, multiple windows representing the same activity are extracted. Although the information in these is highly redundant, the small stride allows to generate a large number of samples, which is important for training a tCNN [GLMR+17, YNS+15].

Networks for M-HAR are relatively small—in comparison to networks for image classification or document analysis—with a maximum of four convolutional layers and three pooling layers. Classification is usually performed using a softmax layer. The authors in [RC15, YNS+15, ZNY+14] introduced tCNNs containing convolution and pooling

operations, which are carried out along the time axis. As local temporal neighbourhoods are likely to be correlated independent of the sensors, these architectures share small convolutional filters among all the sensors. Convolutional filters extract temporal relations from their local neighbourhood at different temporal locations from the input sequences per channel. Temporal filters are shared among all the channels. These temporal relations are likely to be correlated independent of the type of sensor. This assumption is also valid if the measurements per channel are normalized. Ronao and Cho [RC15] investigated a tCNN for temporal filter sizes $\mathbf{F} = \left[\{F_t\}_{F_t=1}^{15} \times 1\right]$, number of filters $C = [10, 20, ..., 200]$ and up to three blocks of convolutional and max-pooling layers. In [YNS+15], Yang et al. evaluated a three-layered tCNN choosing the convolution filters such that the output of the last convolution is one, i.e., $[W_3 = 1, S]$. In [ZNY+14], the 3D components of the acceleration recordings $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ are separated and processed by a shallow 1 layer-tCNN independently. Then, the extracted features are fused by a FC layer.[1] The authors in [DBLG14] deployed a five-layered tCNN with alternating average and max-pooling operations.

Ordóñez et al. [OR16] introduced an architecture that combines temporal convolutions and RNNs. To be precise, they used LSTM units as fuse layers; see Figure 3.1.1. They combined $[\mathbf{F} = [5 \times 1], C = 64]$-temporal convolutions and $C = 128$-LSTMs layers, denoting the architecture as Deep Convolutional LSTM (DeepConvLSTM). This DeepConvLSTM consists of four convolutional layers and three LSTM layers. Convolution layers do not include a pooling operation. The authors observed that the DeepConvLSTM offers better performance when identifying the start and end of activities. Also, DeepConvLSTM improved the classification performance when compared to results reported by [YNS+15], using a four-layered tCNN. LSTM cells capture longer temporal dynamics within the data sequence compared to a tCNN. The authors in [HHP16] utilized a shallow RNN; namely, a three-layered LSTM network and one-layered Bidirectional-LSTM (B-LSTM). B-LSTMs process segmented sequences following their inputs in both forward and backward directions. Although the B-LSTM architecture performs better, the tCNNs are more robust against hyperparameter changes.

Authors in [XHF+18] proposed to use dilated temporal DeepConvLSTMs. The architecture consists of one initial convolutional layer followed by three dilated temporal convolutional layers with different dilated factors. The feature maps of the last dilated temporal convolutional layer is fed to a two-layered LSTMs followed by a softmax. Chen et al. [CLCG18] proposed a combination of LSTMs with raw-data input, an MLP with handcrafted-feature inputs and late fusion. In contrast, Zhu et al. [ZCS19] used a LSTM network with statistical features, Subsection 2.2.3, as input.

1 In general, these networks implement a late fusion of the sensors' features.

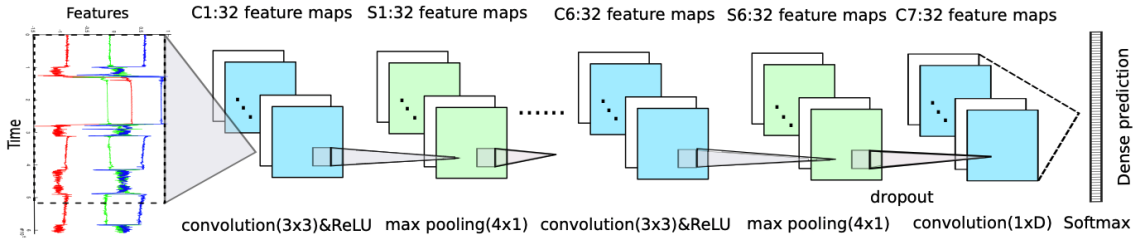


Figure 3.1.2: CNN with 2D convolutions, temporal max-pooling and a Fully Convolutional Layer (FCN) for dense predictions of M-HAR. Image taken from [YLSR18].

Differently, Yao et al. [YLSR18] proposed a CNN with 2D filters, temporal max-pooling and FCNs for predicting sample-wise M-HAR; hence, each sample is mapped to an activity class rather than a window of samples. FCNs perform convolutions considering only the depth of the input feature map, with filters of size $[C \times 1]$ and stride of $[1 \times 1]$ with C the dimension of the input feature. Their 2D convolutions and temporal max-pooling are carried out with padding to compensate for the kernel size and ensure the temporal size of the output as in the input. Their architecture contains six blocks of 2D convolutions with ReLU, and temporal pooling operations; a single FCN and a dense softmax classifier. The dense softmax imposes a softmax classification for the predictions of all the samples; this is different from the existing methods, which apply the conventional softmax loss layer to predict a single window.

The authors in [CVN+20] evaluated the tCNN for different number of convolutional layers, their depth, and filter sizes. Münzner et al. [MSR+17] investigated different sensor-fusion strategies. They utilized filters combining the data channels in the first-convolutional layer or among sensor types. Furthermore, they utilized temporal convolutions and late fusion as in [GLMR+17]. They also proposed a sort of tCNN with branches for each channel. Moreover, the authors evaluated the effect of three normalization strategies: zero-mean and unit standard deviation, batch normalization, and a pressure-mean subtraction. Using a normalization strategy improves the architecture's performance.

The authors of [TLLY18] used spectograms of inertial signals as image inputs for CNNs. They also concatenated recordings from Surface Electromyography Sensors (sEMG), corresponding to muscles activation levels, to the output of the first FC and input it to a Softmax layer.

Shavit and Klein [SK21] proposed a transformer architecture, merging OBD data and position data for M-HAR. The architecture is an attention-based encoder architecture. The authors argue that LSTMs do not perform a holistic temporal aggregation as tCNNs do—transformers aggregate information from the entire sequence. Nevertheless, the

authors only deployed the encoder and an MLP for M-HAR. Besides, it uses the convolutional layers of a pretrained tCNN on the same target dataset for feature extraction. Conformer architectures comprise two parallel branches of convolutional layers and attention layers [KCKL22]. By including bridge connections at different stages of the two branches, conformers seek to learn attention heads and feature extractors parallelly. They combine the feature extraction efficiency of the CNNs with the holistic attention power of transformers in an end-to-end approach. Attention mechanisms have also been explored in [Mah20, BV21, DLKP22] in the context of M-HAR.

3.2 VIDEO- AND POSE-BASED HAR

Deep methods for video-based HAR and pose-based HAR process video inputs in different stages to compute activity predictions. They generally consider the input as individual image frames, where intermediate representations of the frames are computed, e.g., visual, motion, or joint pose representations. Visual representation is a constant for all of the methods. Datasets intended for image and object classification are used for pre-training. The motion representation relies on pre-computing optical flow representation of the video frames. Pose-based HAR consider a representation of the joint poses. Interestingly, this representation is also an embedded result of the overall HAR method. The joint poses are somehow embedded in the methods. The different stages of these methods handle each input type in parallel, being trained individually, or using multi-task learning strategies. As different parallel architectures handle RGB and optical flow images and eventually pose estimations, fusion strategies are considered in different levels, e.g., early fusion either—at the input or in the early layers—, middle and late fusion—mainly combining activity predictions.

Simonyan and Zisserman [SZ14] proposed an architecture for video-based HAR on the UCF101 and HMDB51 [KJG+11] datasets. This architecture is comprised of two individual networks: an RGB CNN and an optical-flow CNN. The RGB CNN classifies single RGB frames from a video to an activity class, similar to a CNN for object recognition. The authors pre-trained the RGB CNN on a large static image dataset to human actions on static images, e.g., ImageNet dataset [RDS+15], transferring visual features. Simultaneously, the optical-flow CNN processes L consecutive flow images extracted from a video.[2] The authors did not pretrain this temporal network since large datasets are unavailable. The authors evaluated the length of consecutive optical flow frames L of an RGB frame to pass into the flow network; finally, setting $L = 10$. They found that subtracting the mean flow image from every flow frame significantly improved the overall accuracy. They argued

2 Optical flow images are computed between consequent frames.

that such subtraction reduces the camera motion, similar to the homography approach. The RGB CNN and an optical-flow CNN have almost the same architecture, consisting of five convolutional and two FC layers, followed by a softmax layer. Both classification outputs are fed to a multiclass linear SVM, which conclusively outputs the final action prediction.

To train their model, Simonyan and Zisserman [SZ14] utilized multi-task learning. First, they combined two datasets for training, the UCF101 and the HMDB51. The overall architecture has two softmax classifiers, one per dataset, as their tasks $\mathcal{T}$ differ, i.e., their label space differs. The authors found that training in such a way has a regularisation effect, reducing the risk of overfitting on any of the two datasets. They use the OpenCV library's algorithm for computing optical flow images. The authors in [FPZ16] proposed different fusion strategies for the RGB CNN and an optical-flow CNN feature maps; this is different to the late fusion or output fusion in [SZ14].

Video-based HAR can also be performed using human pose estimates. Choutas et al. [CWRS18] introduced a representation of videos for video-based HAR, combining joint pose estimation and a temporal aggregation of weighted poses to encode the motion of joints over a video. This representation is called *PoTion*. The PoTion representation aggregates joint movements along the video frames, encoding how much time a joint remains in a pose in pixel coordinates. Joint poses are represented by 2D heatmaps per frame, which are computed by a two-branch CNN trained on the MS COCO dataset. This CNN computes heatmaps of 19 joints, and their respective Part Affinity Field (PAF)[3] [CSWS17]. The authors extended the joints heatmaps encoding the joint movements along the time axis of the video. They assigned a colour value to the joint heat-maps of the first and end frames of the video, and interpolated the in-between joint heat-maps. The PoTion representation is the aggregated colourised joint heat-maps along time or frames—normalised with respect to the number of pixels and frames. Finally, a 6-layered CNN with average pooling and a softmax layer classifies a video processing its PoTion representation of a video.

Khalid and Yu [KY18] expanded the two-streams CNN in [SZ14] using pose information as a third stream. They merged human-pose estimations, RGB frames from videos, and their optical flow images for video-based HAR with deep architectures. They proposed a special fixed order of the human joints based on a tree-like structure. Besides, they deployed early, middle and late fusion of the streams. They estimated the poses for each frame using the joint heatmaps[4] proposed in [CWRS18]. The authors argue that missing

3 PAFs are vectors connecting joints to their two adjacent ones, which are found by a search of uniformly-spaced orientation vectors of all possible vectors connecting detected joints [CSWS17].

4 An argmax function is used as postprocessing step to compute the joint pixel coordinates of the highest peak of the heatmap.

joints need to be interpolated, leading to higher accuracy on the benchmark datasets. They propose two different interpolation strategies: temporal interpolation and spatial interpolation. Temporal interpolation is used whenever a joint is not visible for a few frames. Let $\mathbf{F}_l$ be the frame where the last joint was last visible. This frame is then followed by N frames, where the joint is not visible, followed by $\mathbf{F}_k$, which becomes visible again. Then, the authors linearly interpolate the position of the missing joint for each frame $\mathbf{F}_{l+1}, ..., \mathbf{F}_{l+N}$ using the positions in $\mathbf{F}_l$ and $\mathbf{F}_k$. However, they do not specify what constitutes a small number of frames, i.e., how small N has to be for this approach to lead to good results. Spatial interpolation is used for long-term occlusion, i.e., large N. Khalid and Yu argue that this is necessary since temporal interpolation leads to worse results the bigger N gets. They divide the joints into five groups for spatial interpolation, as Figure 3.2.1 shows. The groups are iterated, starting with the group where the missing joint belongs. Supposing the other joints in a group are present, in that case, they vote on the position of the missing joints based on a statistical model of relative joint positions computed on training data beforehand. The average vote assumes that the missing joints are highly correlated for certain joints. As an example, they mention the correlation between head and neck positions.

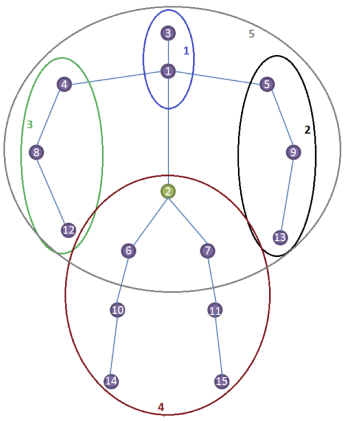


Figure 3.2.1: The authors divide the joints into five groups in order to estimate relative positions of joints to each other. Image taken from [KY18].

Once the joint position is estimated per frame, Khalid and Yu propose a pose tensor by ordering the joint positions, keeping the neighbourhood relationship between joints. They argue that this leads to higher accuracy since the relationship of the joints is a helpful feature, but they do not compare it to other approaches for joint ordering. By traversing the tree in such a way, some nodes are visited multiple times and are thus present multiple times in the pose tensor. That way, the authors argue, neighbourhood relationships are expressed more directly than simply starting at joint "1", see [KY18], and concatenating

the x and y positions. The pose tensor then contains the pixel coordinates, x and y, for each frame in a row. The authors process the video clips in chunks so that the pose tensor always has identical dimensions. Also, the tensor's second and third dimensions contain the joint position coordinates' first and second-order derivatives via finite differences. Finally, the joint positions are normalized with regards to the middle point between the neck and belly joint and the torso length.

Khalid et Yu [KY18] propose a shallow CNN for classification. The network comprises two convolutional layers with ReLU activation function, followed by a max-pooling layer and a FC layer. Finally, a layer with the softmax activation function is used for classification. Since the network is relatively small, the authors do not pre-train it. Instead, for the two-stream architecture, they fuse their pose network into the existing two-stream approach by equally weighting all three components.

In [LPT18], Luvizon et al. combined human pose estimations from images and sequences of RGB frames to create an end-to-end architecture for video-based HAR. To be precise, they merged an architecture for estimating human poses from videos and an architecture for HAR using the human poses to an end-to-end framework. Furthermore, they deployed a differentiable *Soft-argmax* function for computing human-joint pixel coordinates from the feature map activations. Then, these coordinates are fed to the video-based HAR architecture. The proposed differentiable *Soft-argmax* for estimating joint pixel coordinates, which makes the network completely differentiable, thus, allowing for using certain pretrained parts, like the pose estimator, and end-to-end fine-tuning the network. The authors claim that the combination of a pose estimator and activity recognition is novel as pose estimators were not fully differentiable—many pose estimators output a heatmap for each joint position. So, an argmax function is required compute the joint pixel coordinates, as in [CWRS18, KY18]. Hence, the authors propose the *Soft-argmax* function as a differentiable alternative to the regular argmax function. In addition, they propose methods to use both 3D and 2D pose datasets while training their network. Figure 3.2.2 presents the overall architecture.

Their architecture can be divided into several four stages: a *feature extraction*, a *pose estimation*, a pose-based HAR, an *appereance-based* HAR, and an over-all video-based HAR. The *feature extraction* extracts visual features $\mathbf{Z}$ from each input video frame separately. These features are used to predict the joint heatmaps in the pose estimation block. Besides, they serve as input to the *appereance-based* HAR. The authors based their feature extraction network on the Inception v4 network [SIVA17]. The last convolutional layer of the Inception v4 network was replaced by a *depthwise separable convolutional layer*.

The *depthwise separable convolutional layer* consists of two consecutive layers, namely, a *depthwise convolutional layer* and a *pointwise convolutional layer*. The *depthwise convolutional layer* processes a feature map input $\mathbf{Z} \in \mathbb{R}^{[W \times H \times C_{sep}]}$ with $C_{sep} = C$ number of filters

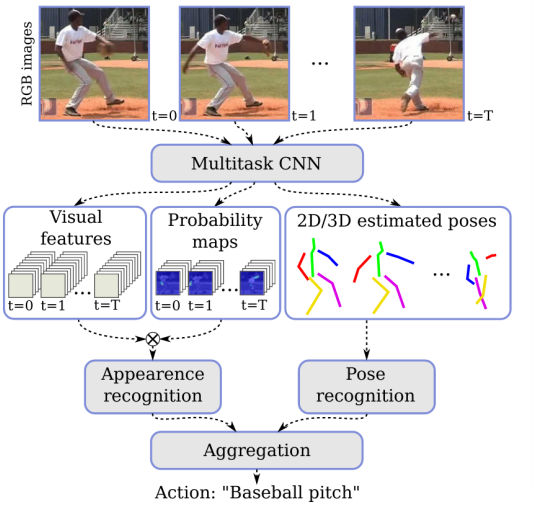


Figure 3.2.2: The multitask method for pose estimation and HAR on videos. High level visualization of the network. The images features and poses are computed frame-by-frame. The output is then passed to the appearance and pose recognition subnetworks, where they are jointly processed to predict the action of the video clip. Image taken from [LPT18].

$\mathbf{W} \in \mathbb{R}^{[F \times F \times 1]}$, one per channel of the feature map input. The resulting feature map is fed to the *pointwise convolutional layer*, which is a FCN with C_{point} number of filters $\mathbf{W} \in \mathbb{R}^{[1 \times 1 \times C_{sep}]}$. The *depthwise separable convolutional layer* reduces the number of parameters and matrix multiplications needed for convolutional layers with many channels.

The visual features $\mathbf{Z}$ are used to estimate joint poses and for action recognition. The pose estimator is constructed from K prediction blocks, which built upon the work by [NYD16]. Like the stacked hourglass architecture, each prediction block computes joint heatmaps on different input scales by utilizing max-pooling and upsampling layers. In particular, the authors use separable residual blocks instead of regular convolutions to extract these scale-dependent features. A separable residual block is defined as a *depthwise separable convolutional layer* whose input is added to the output using a residual connection. The Kth joint heatmaps $\mathbf{Z}^K$ are used to compute joint x and y coordinates using the *Soft-argmax* function.

The authors propose the *Soft-argmax* for computing the pixel coordinates from a joint heatmap. This *Soft-argmax* function computes the x and y image coordinates by

$$\boldsymbol{\Psi}_{[x,y]}\left(\mathbf{Z}^K\right) = \sum_{i=1}^{W}\sum_{j=1}^{H} \mathbf{W}_{i,j,[x,y]}\,\boldsymbol{\varphi}\left(\mathbf{Z}^K_{i,j}\right), \tag{3.2.1}$$

with $\mathbf{W}_{i,j,[x,y]} = \left[\frac{i}{W}, \frac{j}{H}\right]$ being a fixed weight matrix for a convolutional layer, $[W, H]$ refer to the width and height of the heatmap, $[i, j]$ are the pixel coordinates for each

element in the joint heatmaps $\mathbf{Z}^K$, $\boldsymbol{\varphi}(\mathbf{Z}^K_{i,j})$ the softmax on the heatmap $\mathbf{Z}_{i,j}$ for indices i, j, and $\boldsymbol{\Psi}_{[x,y]} \in [0, 1]$. The coordinates $\boldsymbol{\Psi}_{[x,y]}$ represent the highest peak in the probability map computed using the softmax. This is achieved by computing the expectation in x and y direction on the probability map.

The authors argue that such a method for regressing x and y coordinates is more accurate, and it requires fewer model weights than directly regressing them using, e.g., a FC layer. The authors observe that their approach consistently outperforms regression approaches.

Luvizon et al. [LPT18] propose encoding the visual features $\mathbf{Z}$, the joint heatmaps $\mathbf{Z}^K$ and the joint predictions $\boldsymbol{\Psi}_{[x,y]}$ into two tensors, *pose cube* and an *appearance cube*. These tensors are used for two HAR models, namely the pose-based HAR using the *pose cube* and appearance-based HAR using the *appearance cube*. The *pose cube* aggregates the predicted x and y coordinates of joints for each input video frame into a 3D matrix representation of size $[J \times F \times D]$, where F refers to the number of frames of the input clip, J is the number of joints, and D refers to the spatial dimensions, in this case, $D = 2$ for x and y coordinates. To ensure that the *pose cube* keeps a fixed size, regardless of the original input video length, the authors decide to subdivide the video into chunks of $F = 16$ frames, resulting in a total dimensionality of $[16 \times 16 \times 2]$.

The *appearance cube* consists of the concatenation the visual features $\mathbf{Z} \in \mathbb{R}^{[W_z \times H_z \times F_z]}$ weighted by each joint heatmap $\mathbf{Z}^K \in \mathbb{R}^{[W_z \times H_z \times J]}$, resulting in a tensor of size $\mathbb{R}^{[W_z \times H_z \times J \times F]}$ and collapsed or summed up along the spatial dimensions $[W_z \times H_z]$ per frame. A single *appearance cube* entry is a tensor of $[J \times F]$, resulting when summing the activations of a weighted visual features.

Next, two FC neural networks processes the *pose cube* and the *appearance cube* computing two activity predictions. Each FC neural network contains a feature extractor and K activity prediction blocks, each with a $[4 \times 4]$-MaxPlusMin pooling Equation 3.2.2 and a softmax layer. However, the authors do not explain the benefit of such a pooling operation over a regular max-pooling layer. Each activity prediction block computes an intermediate activity prediction. The Kth prediction is considered as the activity prediction.

$$\mathrm{MaxMinPooling}(x) = \mathrm{MaxPool}(x) - \mathrm{MaxPool}(-x) \tag{3.2.2}$$

A final MLP combines the the pose-based and the appearance-based activity predictions producing the overall-prediction.

In [JMA22], the authors deployed a tCNN processing temporal feature representations of optical flow images of a video. They deployed a pre-trained CNN to extract spatial feature maps from optical-flow images. These feature maps are then reduced by PCA.

The dimensions of the aggregated feature maps are time-wise concatenated. A tCNN further processes the temporal representation. A 1D-CNN is used to extract temporal information from these flow-feature maps. This tCNN is the only one to be trained. Their proposal seeks to reduce the number of trainable parts of a video-based HAR, combining well-known space feature maps computation, time encoded in optical flow and a tCNN. The authors suggest that tCNN contains significantly lesser parameters than CNNs, which helps in situations where one uses smaller datasets.

3.3 SYNTHETIC DATA FOR HUMAN POSE

Figure 3.3.1: Example of a triangulated mesh representing a human from recordings of marker-based MoCap. Image taken from [LMB14].

Synthetic data is considered for pose estimation from including measurements of inertial data using deep architectures. Computing human poses from inertial measurements is an under-constrained problem. Synthetic data helps for learning priors temporal features, which improve human pose estimation.

Loper et al. [LMB14] computed the 3D body-model and captured its non-rigid motion of soft tissue using the raw marker-based MoCap data with 67 markers. They estimated 3D body shape and pose in terms of a triangulated mesh representing the human body, using the recordings of the 67 makers and deploying a parametric model of the human body; see Figure 3.3.1. The authors showed that marker-based MoCap captures detailed sequential human motion data, even for producing lifelike, non-rigid animations. The authors in [LMR+15] adapted a triangulated mesh of the human body, using recordings of six OBDs and a kinematics chain with 24 joints. They used the Skinned Multi-Person Linear Model (SMPL) as a body model, having 6890 vertices. As they used sparse OBDs, the authors proposed a consistency loss among human body models for a set of frames. An extension of this work is presented in [MRBPM17]. The authors computed the 3D human pose from a small set of OBDs.

Figure 3.3.2: Example of the SMPL using a scarce set of OBDs. Image taken from [LMR+15].

The authors in [HKA+18] predict human poses, in terms of SMPL models, using sparse OBDs using a RNN, called Deep Inertial Poser (DIP). They trained a B-LSTM network processing sequences of OBD recordings and predicted the parameters of a SMPL model in the next time steps. As OBD data is scarce, especially with synchronised pose annotations, they proposed training their model using synthetic data. They created SOBDs, called by the authors virtual sensors, generated from a large raw marker-based MoCap dataset. They placed SOBDs on a SMPL model [LMR+15]. Lastly, they computed their synthetic poses, positions and orientations through forward kinematics, and their respective linear accelerations via finite differences, as Equation 3.3.1. The authors trained their B-LSTM in two steps: first, they trained solely on the SOBDs; second, they fine-tuned their model using recordings from six OBDs.

$$\mathbf{a}_t^{(J)} = \frac{\mathbf{p}_{t-1}^{(J)} + \mathbf{p}_{t+1}^{(J)} - 2\mathbf{p}_t^{(J)}}{dt^2}, \tag{3.3.1}$$

where $\mathbf{a}_t^{(J)}$ is the linear acceleration of joint J with 3D position $\mathbf{p}_t^{(J)}$ at time t [HKA+18].

Their RNN computes synthetic measurements from six SOBDs. The authors used one of the SOBD located on the subject's lower back as a root SOBD. They normalised the remaining sensors with respect to the root SOBD. An input is a sequence of duration W with the orientations and the linear accelerations from the five SOBDs, a $\mathbf{X} \in \mathbb{R}^{[W,60]}$. The

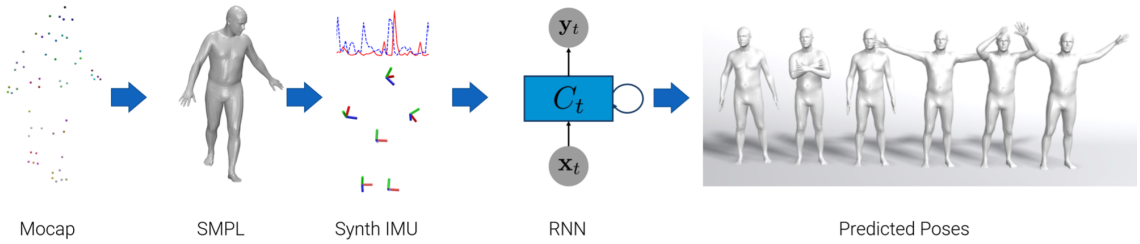


Figure 3.3.3: DIP that predicts parameters of a SMPL model using a B-LSTM processing recordings from six OBDs. The B-LSTM was trained on SOBDs located on a SMPL model, which is computed from marker-based MoCap. Image taken from [HKA+18].

orientations are given in terms of rotational matrices $\mathbf{R} \in \mathbb{R}^{[3,3]}$ in the SMPL model, and the linear accelerations as a vector in $[x, y, z]$, $\mathbf{a} \in \mathbb{R}^{[3]}$. The network consists of four layers: a dense layer with 512 neurons, two B-LSTM layers with 512 neurons, and two parallel regressor layers. Each layer comprises a dense layer with 512 neurons and an output layer. The two regressors output the mean, $\mu_{smpl} \in \mathbb{R}^{135}$, and standard deviation, $\sigma_{smpl} \in \mathbb{R}^{135}$ of the SMPL model, and the mean, $\mu_a \in \mathbb{R}^{15}$, and standard deviation, $\sigma_a \in \mathbb{R}^{15}$, of the linear accelerations of the SOBDs, respectively. They proposed reconstructing the acceleration input for propagating the acceleration information throughout the network, as networks only predicting the orientation of the SMPL model tended to discard the acceleration input.

For training, the authors normalised the input vector using zero-mean unit variance, as Subsection 2.2.2 describes for M-HAR. The network's output is a vector with the mean and standard deviation of the orientation of the SMPL model and linear accelerations of the OBDs. They trained their network using the following losses,

$$\log\left(p(\mathbf{y})\right) = \sum_{t=1}^{W} \log\left(\mathcal{N}\left(\mathbf{y}_t | \mu_{smpl}, \sigma_{smpl}\mathbf{I}\right)\right), \tag{3.3.2}$$

and

$$\log\left(p(\mathbf{a})\right) = \sum_{t=1}^{W} \log\left(\mathcal{N}\left(\mathbf{a}_t | \mu_a, \sigma_a\mathbf{I}\right)\right), \tag{3.3.3}$$

with $\mathbf{y}$ the target poses of the SMPL model, $\mathbf{a}$ the measurements of the OBDs, $\mathbf{I}$ the identity matrix, and $\mathcal{N}(\cdot)$ a Gaussian distribution, for a time t of a sequence length W. Noteworthy, they utilized a dropout layer after the input layer for regularisation.

Finally, the authors fine-tuned their model with six real OBDs measurements. For that, the orientation part of the input is computed from the real OBDs using forward kinematics, starting from an idle pose of the subject in a calibration phase.

The authors showed that human pose could be computed from sparse OBDs using a RNN model. Moreover, they carried out a baseline model using synchronised recordings of OBDs and marker-based MoCap, comparing against the models trained with SOBDs. They demonstrated the feasibility of training of a model with synthetic data generated from a large marker-based MoCap dataset.

3.4 TRANSFER LEARNING USING DNNS FOR HAR

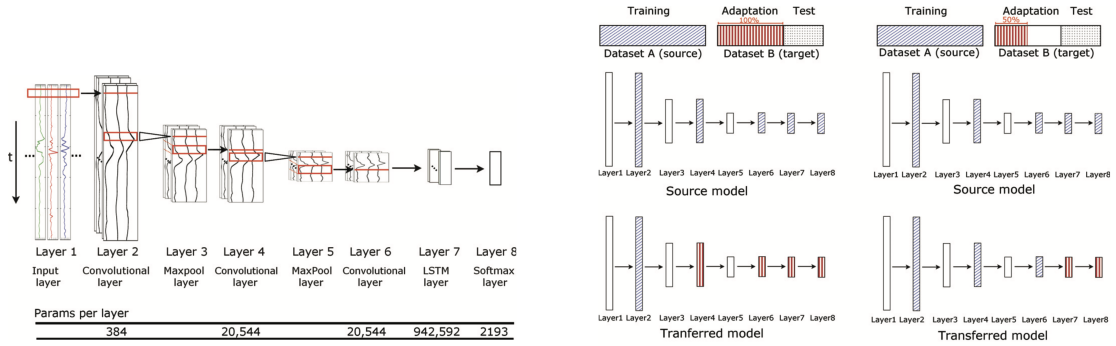
In transfer learning, learnt features from a particular domain, called source, are used on a related one, called target. In the context of deep networks, transfer learning has been used for coping with the limited amount of data in challenging problems, e.g., object detection [RHGS15], face recognition [PVZ15], and word spotting [GSF18]. Transfer Learning for HAR gives an alternative to the problem of scarcity of annotated data, considering that DL approaches demand a large amount of annotated data [OMR16]. Besides, it is of use when facing overfitting and the class-imbalance problems. As a result, Transfer Learning for HAR allows spending less time learning new tasks and requires lesser human experts' work, and more HAR scenarios can be handled effectively. Section 3.3 has already shown an example of transfer learning, where parameters of an RNN are pre-trained with SOBD for human pose estimation.

3.4.1 *Parameter Transfer*

In the case of deep learning, transfer learning uses a deep architecture trained on a source task to initialise a network on a related target one. In the context of DL for image classification and segmentation, so-called fine-tuning is a common and straightforward strategy for parameter transfer among similar tasks. Parameters from the convolutional layers trained on a large source dataset are taken for initialising networks to adapt to the target task. The FCs are the only ones that are trained from scratch.

Ordóñez Morales and Roggen in [OMR16] characterised the feasibility, benefits, and drawbacks of performing transfer learning in three M-HAR scenarios. For each scenario, they define the source $\mathcal{D}_{source}$ and target $\mathcal{D}_{target}$ domains. These scenarios are across subjects within a dataset $\mathbf{X} \in \mathcal{D}$ with $\mathcal{D}_{source} = \mathcal{D}_{target}$; across datasets from D different domains $\{\mathcal{D}^{(d)}\}_{d=0}^{D}$, i.e., $\mathcal{D}_{source} \neq \mathcal{D}_{target}$; and a more challenging one across datasets,

OBD locations on the human body, and sensor type, i.e., $\mathcal{D}_{source} \neq \mathcal{D}_{target}$. Moreover, they examined the effect of the number of transferred layers.



(a) TCNN-LSTM

(b) Feature-based transfer learning

Figure 3.4.1: The parameter-based transfer learning using tCNNs for M-HAR. The tCNN contains three convolutional layers, alternated with max-pooling layers, an LSTM layer and a final softmax layer. Besides, the number of parameters are given for each layer, with the example of a softmax layer for the Opportunity Challenge Dataset (Opp). Image taken from [OMR16].

The authors deployed a tCNN-LSTM with three convolutional layers with ReLU activation function, alternated with max-pooling layers, followed by an LSTM and a softmax layer, as Figure 3.4.1 shows.

They limit the scenarios such that the recordings of the source and target sets remain of the same size. Following this constraint, the tCNN-LSTM networks for the source and the target are structurally the same, except for the softmax layer. Hence, the parameters can be seamlessly copied. The transferred layers are fine-tuned with a proportion [%] of the target dataset and evaluated on a test subset of the target dataset. The non-transferred layers are initialised randomly. The transferred layers are frozen—transferred layer parameters do not change.

The authors evaluated Transfer Learning for HAR on two benchmark datasets, the Opportunity Challenge Dataset (Opp), see Subsection 6.1.1, and the Skoda Mini Checkpoint. Both datasets are composed of multi-channel Time-Series on multiple locations on the human body, and they contain activities of similar time scales, recording a sampling rate of 30Hz. In both cases, sensor data were pre-processed to fill in missing values using linear interpolation; an L2 normalisation per channel was performed. The authors used five OBDs for the Opp dataset. The Skoda Mini Checkpoint comprises recordings of one subject performing control gestures in a car production plant. Measurements of the OBDs are treated as individual channels.

The authors trained their models using SGD with Nesterov momentum, minimising the TCCE loss function. They trained their models for 300 epochs on the source datasets and fine-tuned them for 100 epochs on the target datasets. Training is repeated on average five times for all models to account for variations from run to run as the models are randomly initialised, reporting the mean and Standard Deviation (SD) of performance. The non-overlapping input segments are composed of 322 samples (9.7 seconds), which is, on average, long enough to learn temporal dependencies for the activities included in both datasets. This truncated input is a practical computational optimisation in the LSTM learning process. The authors always adapted their pre-trained model to the target dataset for deployment (fine-tuning). However, they considered segments of 9.7 sec. for training the long-dependencies of the LSTMs.

In the first case, transferring across subjects within the same domain, they consider the case when the target samples are drawn from the source domain $\mathcal{D}_{source}$ for different subjects; thus, the $\mathcal{X}_{source} = \mathcal{X}_{target}$ and $\mathcal{Y}_{source} = \mathcal{Y}_{target}$. They evaluated this scenario on the Opp dataset, which contains recordings from four subjects. The $\{(\mathcal{X}, \mathcal{Y})\}_{target}$ and $\{(\mathcal{X}, \mathcal{Y})\}_{source}$ are created using the leave-one-user-out approach, i.e., three subjects were used for the source dataset and the remaining one for the target. The authors concluded that the filters of the lower layers are more generic and user-independent and, thus, transferable—these filters learn, for example, mannerisms—compared to deeper layers. Performance decreases when transferring more than three layers. Moreover, training time on the target datasets reduces around 17% without performance loss. Besides, increasing the proportion of the target dataset for fine-tuning increases performance.

In the second case, transferring across tasks of different domains, Ordóñez Morales and Roggen considered the case when the domains are different $\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$. This case helps to know whether it is possible to have generic filters for human activities. They deployed only measurements from a single accelerometer located on the same human limb from the two domains, simplifying the transfer problem. Hence, the samples for the source and target datasets were assumed to be drawn from the same multi-channel time-series Space (multi-channel time-series Space) $\mathcal{D}_{source} \approx \mathcal{D}_{target}$, and the OBDs are located in the same position on the subject. However, the activities from the two domains differ $\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$. They deployed the Opp dataset as the source and the Skoda as the target dataset. They restricted the experiments to a single accelerometer located on the subject's right hand. Furthermore, they deployed 80% of the Skoda dataset for fine-tuning. The authors inferred that performance varies when the source and target datasets are different in the second case. Performance decreases as more layers are transferred. They speculate that deeper layers learn more scenario-specific features, yielding a worse transferability in case of Opp to Skoda and Skoda to Opp, regardless of the amount of data for fine-tuning. They conclude that even the generic layers are task-related; hence,

non-transferable. These findings suggest that the choice of source datasets is important. In the case of Opp, the filters are better than the ones from Skoda, i.e., the movements in Opp are more naturalistic. However, the type of activities in the source domain negatively affects the transfer learning performance.

Finally, transferring between domains, OBD locations on the body, and tasks, they considered the case when the domains are different $\mathcal{D}_{source} \neq \mathcal{D}_{target}$, and $\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$. They created a target dataset with a single sensor from all the four subjects, the accelerometer located on the right hand from the Opp dataset. The source dataset contains recordings from the Skoda dataset located on the right hand, from the Opp dataset on the left hand and right arm or different sensor types from gyroscopes located in the right hand from the Opp dataset. Here, 100% of the target dataset was used for fine-tuning. There was significant degradation in performance when transferring between modalities and locations irrespective of the number of transferred layers. The generic layers are domain- and task-specific for transferability. The best performance is obtained when transferring from a different location within the same domain. Source models from different domains and tasks show the worst performance.

In general, for the first two scenarios, they created subsets of the datasets such that the source and target datasets contained similar sensor settings and positions, $\mathcal{D}_{source} \approx \mathcal{D}_{target}$. In the third scenario, the target domain consisted only of recordings from a single sensor on the right hand; the source contained recordings of a similar sensor from the source dataset located on the right hand, of a similar sensor from the target domain on the left hand, of different sensor types from the target domain, worn on the right hand. The authors found that transfer learning was only possible within the same dataset, sensor type and position, independently of the number of transferred layers.

Chikhaoui et al. [CGS18] investigated the transfer learning performance on three scenarios focusing on: across subjects of different ages and biological sex using the same OBDs on the same location, i.e., on a similar domain $\mathcal{D}$; different positions of OBDs on the human body; and different sampling rates, sensor types, different domain $\mathcal{D}_{source} \neq \mathcal{D}_{target}$. and different tasks $\mathcal{Y}_{source} \neq \mathcal{Y}_{target}$. The authors used a tCNN with several blocks of two temporal convolutional layers, max-pooling operations, and a subsequent fusing part with a fully-connected layer and a softmax classifier. They do not fine-tune the transferred layers, but the remaining randomly initialised layers. Similar to [OMR16], they deployed very large non-overlapping windows, i.e., of 4 s, for segmenting the sequences. In case of transferring across subjects of different ages, among two datasets of different domains, the authors simplified the transfer task using only a single accelerometer on the forearm sensor placement, i.e., $\mathcal{D}_{source} \approx \mathcal{D}_{target}$. With this constraint, recordings from source and target datasets are equally shaped, so the architectures remain of same structure. This limitation is also seen in [OMR16]. They

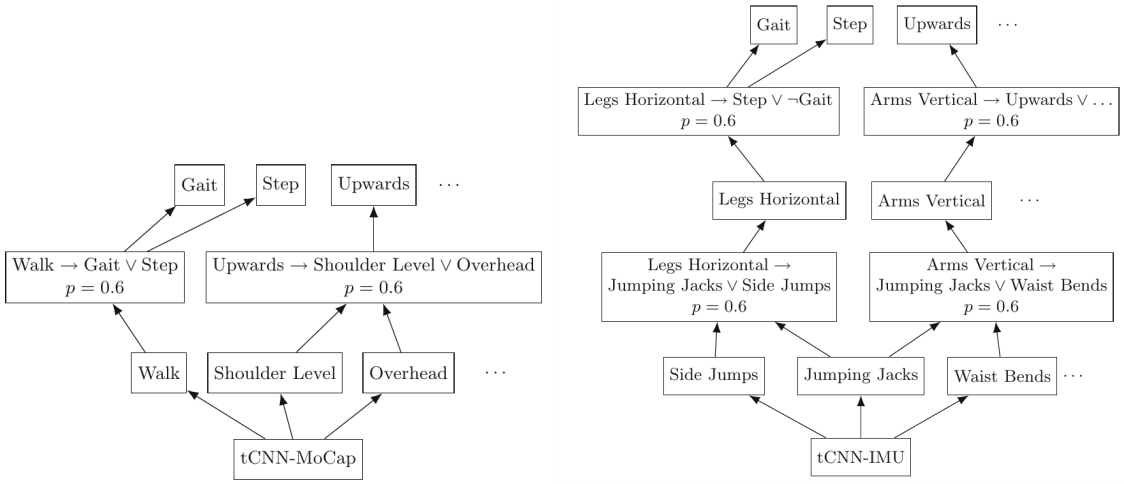
found that transferring from a source dataset with subjects of a wide range of ages is practical. In the case of transferring under different locations of the OBDs from two different datasets, similarly, they simplify the transfer among a single accelerometer between the source and the target. The authors inferred that transfer learning between any accelerometer location on the human body is possible. Besides, fine-tuning performed well across datasets with devices in generic and common locations on the human body. Lastly, they transfer among datasets with different sampling rates and sensor types, finding that lower layers capture generic features independent of the sampling rate. In general, they found that transferring on early layers enhances the architecture's performance on the target domain, e.g., in their experiments, up to four convolutional layers improve performance.

In brief, the work of [OMR16, CGS18] simplifies the transfer using only accelerometers. They found that transferability among datasets of different domains with different accelerometers and sampling rates is only possible with the first generic layers. However, they depend on fine-tuning a large proportion of the target domain.

Another approach for transferring the knowledge of a certain domain to other ones is zero-shot learning. Zero-shot learning is a classification task where the source and target tasks are disjoint, $\mathcal{T}_{source} \neq \mathcal{T}_{target}$, but having equal domain $\mathcal{D}_{source} = \mathcal{D}_{target}$ [LNH14, XLSA19]. Chapter 4 presents in more detail zero-shot learning for M-HAR.

3.4.2 *Relational-knowledge Transfer*

The authors in [KSR+21b] proposed an interpretable Transfer Learning for HAR method connecting the activity classes from source and target datasets through a set of manual or learnable rules from the semantic definition of the activities. They exploited the Probabilistic Rule Stacking Learner (pRSL) from [KSR+21a], finding logic rules in a probabilistic framework combining weak classifiers for HAR. The authors created a set of hierarchical logic rules for deriving activity classes of the target dataset using predictions of a tCNN, which is trained on source datasets, whose MoCap recordings and OBDs are similar to the target ones. Recordings of the source dataset of devices of different types and on different locations from the target dataset are discarded, $\mathcal{D}_{source} \rightarrow \mathcal{D}_{target}$. The authors provided manual interpretable rules that link the source datasets' activity classes to those of the target dataset for being used to a pRSL, as Figure 3.4.2 shows. They also trained a pRSL to find rules between the source and the target activity classes using a proportion of the target dataset, as Figure 3.4.2 shows. The interpretable Transfer Learning for HAR method requires the activity labels of the target dataset for deriving the semantic logic rules to the source dataset.



(a) Direct Transfer

(b) Indirect Transfer via Latent Labels

Figure 3.4.2: Connection of target activity labels to target labels via rules. a) shows the manual direct transfer, where domains are similar. b) shows the indirect transfer where domains differ, source activity labels are abstracted to limb movements labels and then connected to the target labels. Image taken from [KSR+21b].

3.5 DISCUSSION

Although Transfer Learning for HAR has progressed significantly in the last few years, there are still many open challenges. HAR remains a challenging task due to the large intra- and inter-class variability of human movements, i.e., humans carry out similar tasks differently. Additionally, there is a broad range of human activities or movements, and there is neither a standard definition nor structure for formulating an apparent problem of HAR [BBS14, ZNY+14, OR16]. There are more than 400 different activities that people do in their daily lives; without considering activities related to the diversity of people and cultures, the actual number of activities is likely even larger [CSG+13]. Transfer learning has been only carried out across the same domain, i.e., measurements from similar sensors from OBDs on specific locations of the human body [PY10, AR17, GY17]. Transfer learning across multi-channel time-series Space, e.g., sensor modalities, and across different tasks remain challenging. TL is limited by datasets with an equal number and type of sensor devices and localization on the human body. Besides, the source dataset should ideally be large, comprising recordings from several subjects and OBDs [OMR16].

Transfer learning has been carried out among datasets from OBDs. These datasets remain, however, scarce or limited in size. Moreover, annotated data is limited. Recording settings vary enormously. Authors use different recording rates, sensor resolutions, and device positioning, as summarised by the author of this thesis in [RNMR+19]. These variate settings might hinder the transfer learning among different scenarios for coping with the limited or lack of annotated datasets and for classifying unseen human actions [AR17]. Furthermore, datasets contain annotations of task-related activities.

Therefore, exploiting other data sources might be interesting for transferring purposes in M-HAR. Performing Transfer Learning for HAR for non-labelled source data has received little attention in current research. In other areas apart from HAR, researchers have leveraged the unlabeled source data to improve transfer in the target domain, but such techniques are yet to be applied to M-HAR. For example, most transfer learning works in video-based HAR have focused on parameter-based Informed Supervised Transfer Learning (IS Transfer Learning). Additionally, transferring across different activity classes is a much less studied problem in Transfer Learning for HAR. Finally, parameter-based transfer learning is also less studied for the ambient sensor modality. The current direction of most Transfer Learning for HAR is to push the limits on how different the source and target domains and tasks can be.

4 ATTRIBUTE REPRESENTATION FOR DEEP NEURAL NETWORKS

High-level attribute representations are semantic descriptions of categories, and they are relevant as a method for transfer learning in Zero-shot learning problems or content-based image retrieval. For example, attributes can be the shape, colour, texture, size of objects, or even geographic information [LNH09, LNH14] for object recognition in object and scene recognition [LNH14, ZJC17], and characters or strokes of words in visual document analysis [AGFV14, SF18]. For example, in HAR, a collection of verbs and objects represent human actions in images and videos [ZJC17]. Besides, attribute representations help in cases where data are highly imbalanced and datasets with a large number of classes, e.g., words of a language. Moreover, they become practical where annotations are hard to obtain [CGD+13]; see Chapter 7 for more details regarding annotation for M-HAR.

Having introduced DNNs for M-HAR in Chapter 3, this chapter firstly revises related works of attribute-based CNN in Section 4.1. An attribute-based CNN was first introduced for handwritten word spotting. Secondly, Section 4.2 presents an example of synthetic data generation for improving the performance of the CNN for word spotting, exploiting attribute representations. Furthermore, this chapter presents works of attribute-based classification for video-based HAR in Section 4.3 and for M-HAR with a standard statistical HAR pipeline in Section 4.4.

4.1 ATTRIBUTE-BASED CNN

Attribute representations in document analysis are related to the character-based appearance in word images. These attributes are advantageous as words are composed of a sequential combination of characters, i.e., from the alphabet. The alphabet becomes a common representation for words in a certain language. An attribute vector represents a string embedding of words. Attributes are, for example, word characters. A document analysis system uses attributes for learning a transformation of word images or snippets to the attribute representation space. The usual method is to consider these transformation as multi-label classification [LNH14].

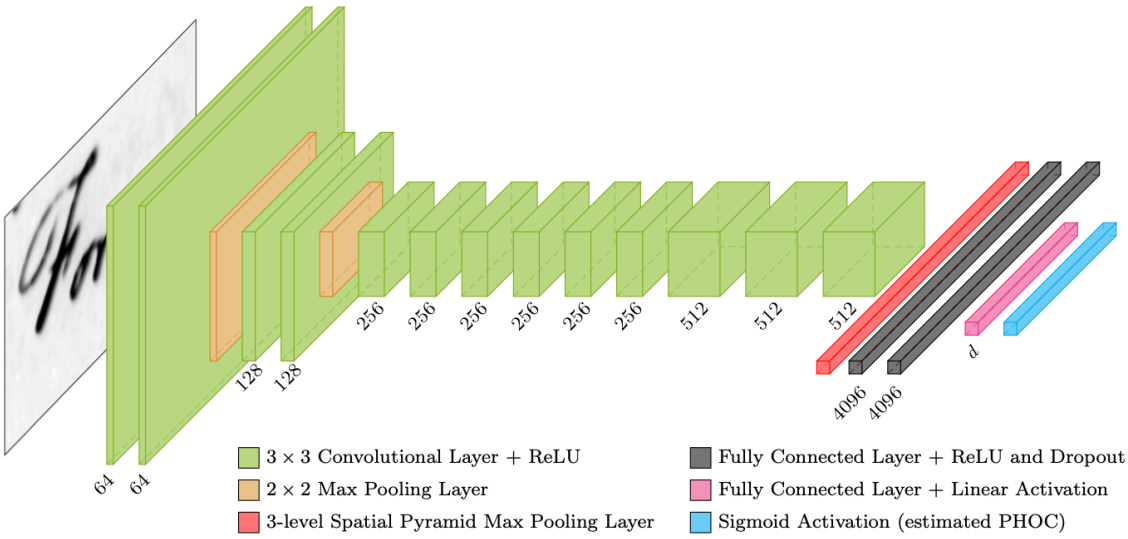


Figure 4.1.1: Pyramidal Histogram of Characters Network (PHOCNet) for handwritten word spotting, predicting a Pyramidal Histogram of Characters (PHOC) representation of a word image. Image taken from [SF16, SF18].

The authors in [SF16, SF18] presented a method for handwritten word spotting using attribute representations. They introduced the PHOCNet computing an attribute representation of word images. This representation is the PHOC, proposed by Almazán et al. [AGFV14], which encodes the presence of characters in a word. Besides, it considers spatial information of the character's presence in a pyramid fashion. The PHOC is a concatenation of the characters' splits per level of a word. The PHOCNet is based on the VGG16 network, which is CNN for object classification and face recognition; see Figure 4.1.1. The PHOCNet uses filters of $[3 \times 3]$ filters. As word images are of different sizes, the authors in [SF16] proposed a SPP layer and in [SF18] the TPP prior to the MLP. These pooling layers allow for computation of a fix-sized deep representation, independent of the input size; see Equation 2.1.3 for a revision of these two polling layers. A word spotting task seeks to retrieve samples of a dataset similar to a query. The authors ranked the samples based on the distance between the PHOC attribute representation of the word samples and the one from the query. The query can be selected as an image sample, query-by-example from the word string, or query-by-string. The cosine distance, Equation 4.1.1, has been deployed for ranking the samples. The cosine similarity results

from the angle between two vectors and evaluates to 0 for orthogonal vectors and 1 otherwise, considering only positive attribute values. The cosine similarity is defined as:

$$d_{cos}(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{a}^T \mathbf{b}}{\|\mathbf{a}\|_2 \cdot \|\mathbf{b}\|_2}, \tag{4.1.1}$$

having the vectors $\mathbf{a}$ and $\mathbf{b}$ denote the user-defined query and PHOCNet-predicted attributes.

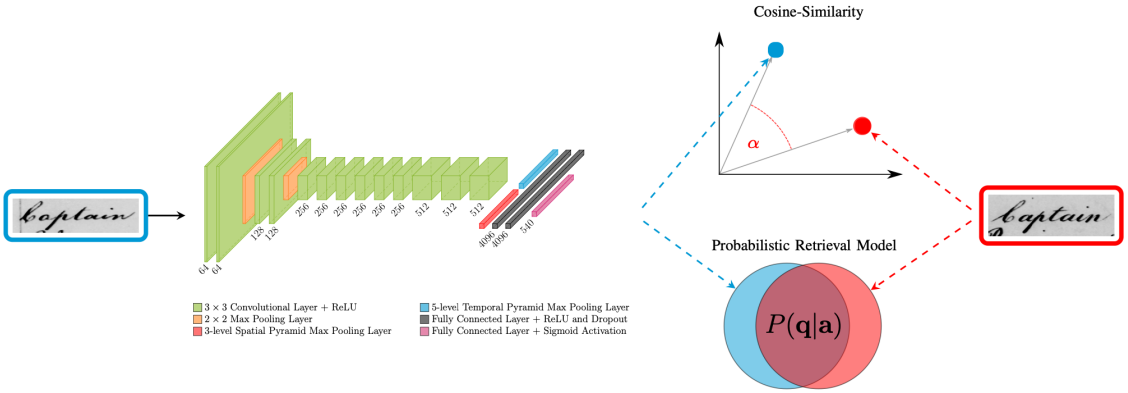


Figure 4.1.2: Probabilistic Retrieval Model (PRM) for handwritten word spotting using an attribute representation. The adapted PHOC predicts a PHOC representation of a word image. Image taken from [RRMF18, RSMF21].

Rusakov et al. [RRMF18] proposed the PRM for handwritten word-spotting and cuneiform spotting [RSMF21]. The PRM, Equation 4.1.2, interprets the attributes computed by the PHOCNet as probabilities. Considering each attribute as a binary random variable $a_m \in \mathbf{a}^M$, and conditional independence among attributes, their behaviour can be described by a Bernoulli distribution.

$$d_{PRM}(\mathbf{a}, \mathbf{b}) = \prod_{m=1}^{M} b_m^{a_m} \cdot (1 - b_m)^{(1-a_m)} \tag{4.1.2}$$

The model now computes an estimate $\hat{a}_m = p(a_m = 1|\mathbf{X})$ for the probability of the m-th attribute being present in the input $\mathbf{X}$. Due to the assumption of conditional independence, the PRM evaluates predicted attributes individually. In contrast, all the attributes holistically contribute to the cosine distance.

Furthermore, Rusakov et al. [RSMF21] concatenated parallel SPP and TPP outputs, Equation 2.1.3, fusing them into a deep representation of the input images, as Figure 4.1.2 shows.

4.2 SYNTHETIC DATA FOR LEARNING DEEP REPRESENTATION FOR WORD SPOTTING

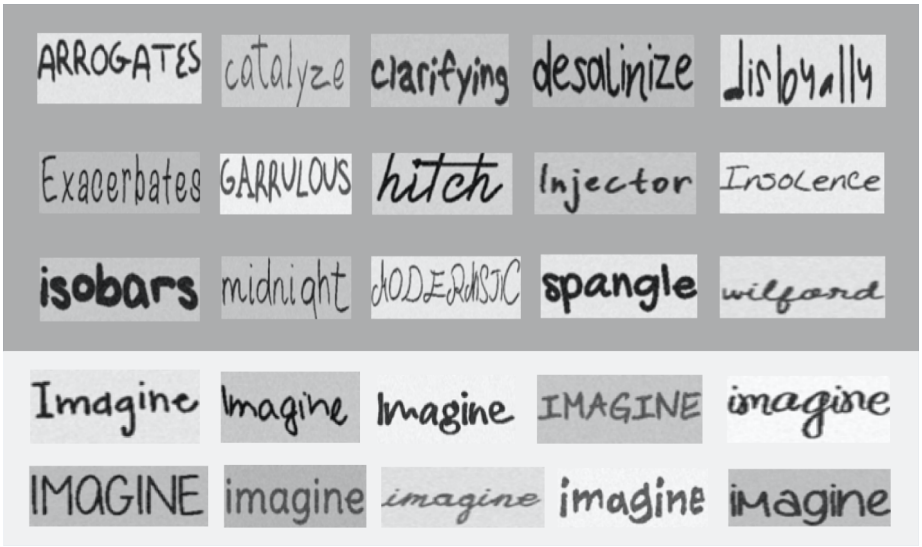


Figure 4.2.1: Random samples (top), along with samples of the same word with different fonts (bottom) of the machine-generated printed text, HW-SYNTH dataset. Image taken from [GSF18].

The authors [GSF18] improved the performance of the PHOCNet for handwritten word spotting under scarce annotated datasets, proposing the usage of synthetic generated data and exploiting the PHOC representation. They deployed machine-generated printed texts, using 100 different fonts. They addressed the variability of handwritten texts. The dataset is rendered by varying the inter-character space, the stroke width, and the main foreground and background pixel distributions.

The authors transfer the PHOCNet trained on the synthetic data to three real target datasets for performing word spotting. These datasets consist of word images from a historical collection from a single writer, a historical collection of different writers, and a contemporary collection from multiple writers, i.e., the input spaces from the generated source dataset and the target ones differ. Besides, they evaluated fine-tuning on the target dataset under different proportions. They showed that fine-tuning the pre-trained PHOCNet with only 100 samples of the target datasets performs similarly to

the unsupervised method; around 10% of the target datasets are needed for surpassing the performance of an attribute-based SVM; only 100 word images for achieving similar performance as a Triplet-CNN. For context, the target datasets, namely, historical single writer, historical different writers, and contemporary multiple writers, contain 4.860, 130.480, and 115.320 word images.

4.3 ATTRIBUTE REPRESENTATION FOR VIDEO-BASED HAR

In [ZJC17], human-annotated attributes and data-driven attributes are combined for solving video-based HAR in sports videos. They selected a subset from both attribute groups, maximising attributes' discrimination capability for distinguishing different sets of activity classes. The authors addressed visual classes through a collection of human-annotated attributes. For example, the activity class *long-jump* in *Olympic Sports Dataset* is associated with either motion attributes *jump forward*, *motion in the air*, or with scene attributes, e.g., outdoor and track.

Given a video sample $\mathbf{X} \in \mathcal{X}^{[W,H,3]}$, the classifier $\mathbf{f}_a : \mathbf{X} \rightarrow \mathbb{B}$ predicts the confidence score of the presence of an attribute a_m in an image or video. The classifier $\mathbf{f}_a$ is learned using the training samples of all classes, which have this attribute as positive and the rest as negative. Given a set of M attribute classifiers $\{\mathbf{f}_{a_m}(\mathbf{X})\}_{m=0}^{M}$, an instance $\mathbf{X}$ is mapped to the semantic attribute space $\mathcal{A} \in \mathbb{B}^M$:

To extract data-driven attributes, Zheng et al. [ZJC17] proposed to compute a larger set of data-driven attributes using a dictionary learning method. Low-level hand-crafted features of N training samples are denoted as $\mathbf{X} = \{\mathbf{x}_n\}_{n=0}^{N} \in \mathbb{R}^{q \times N}$, with q the dimensions of the extracted features. Having K classes, the features of training samples are $\mathbf{X}^{(k)} \in \mathbb{R}^{q \times N_k}$, where N_k denotes samples from class k. For each class k, they first learn a class-specific dictionary $\mathbf{A}^{M_k}$ using the K-SVD algorithm of [AEB06]. The K-SVD is an algorithm for finding a transformation with an iterative method, generalising the K-means algorithm. This transformation is a dictionary matrix $\mathbf{A}$ that leads to the best possible representations for each member in this set with strict sparsity constraints. Using an over-complete dictionary matrix, a sort of transformation, $\mathbf{A} \in \mathbb{R}^{q \times M}$, for M prototype attributes, or signal-atoms, a feature sample $\mathbf{x} \in \mathbb{R}^q$ can be represented as a sparse linear combination of these attributes. The representation of $\mathbf{x}$ might be $\mathbf{x} = \mathbf{A} \cdot \mathbf{z}$ or $\mathbf{x} \approx \mathbf{A} \cdot \mathbf{z}$, with $\mathbf{z} \in \mathbb{R}^M$ being the representation coefficients of the feature representation $\mathbf{x}$. Zheng [ZJC17] initialised a dictionary $\mathbf{A}$ using these class-specific dictionaries as $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K]$ and learned

it by minimising the reconstruction error of all the training samples. The class-specific dictionary $\mathbf{A}_k$ is learned by solving the following problem:

$$\underset{\mathbf{A}^{(k)},\mathbf{Z}^{(k)}}{\operatorname{argmin}} \left\| \mathbf{X}^{(k)} - \mathbf{A}^{(k)}\mathbf{Z}^{(k)} \right\|^2, \tag{4.3.1}$$

$\forall i$, $\|\mathbf{z}_{k_i}\|_0 \leqslant T$, where $\mathbf{A}^{(k)} = [\mathbf{a}_1^{(k)}, \mathbf{a}_2^{(k)}, ..., \mathbf{a}_{M_k}^{(k)}] \in \mathbb{R}^{q \times M_k}$, $\mathbf{Z}^{(k)} = [\mathbf{z}_1^{(k)}, \mathbf{z}_2^{(k)}, ..., \mathbf{z}_{N_k}^{(k)}] \in \mathbb{R}^{M_k \times N_k}$ are the sparse codes of $\mathbf{X}^{(k)}$. The sparsity constraint $\|z_i\|_0 \leqslant T$ specifies that the sample x_i has fewer than T dictionary atoms from $\mathbf{A}$ in its decomposition. It means that each vector $\mathbf{z}_i$ is sparse and has fewer than T non-zero entries.

After they have computed the class-specific dictionaries, they concatenated them to initialize a total dictionary $\mathbf{A} = [\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(k)}]^M$, with $M = \sum_{k=1}^{K} M_k$. The value of the jth entry z_{ij} in the coefficients z_i indicates whether the dictionary atom d_j is used for the decomposition of sample x_i. Thus, each dictionary atom d_j is treated as a data-driven attribute, and z_i is treated as M-dimensional attribute score vector.

The authors used the *Olympic Sports Dataset* containing 783 YouTube video samples of subjects practising different sports, containing $K = 16$ sports activity classes. They used human-annotated attributes. First, they computed Spatio-Temporal Interest Points (STIP) features. Then, they applied a 2D Gaussian smoothing filter to the video along the spatial dimension, followed by a pair of 1D temporal Gabor filters. Afterwards, they detect up to 200 interest points at the local maximum response from each action video. Next, they extract spatio-temporal volumes around the interest points and obtain 100-dimensional gradient-based descriptors via PCA. Finally, these descriptors based on interest points are quantized into 2000 visual words by K-means clustering, and a 2000 dimensional histogram represents each action video.

Three attribute-based representations are constructed, having Human-Labeled Attribute (HLA), Data-Driven Attribute (DDA) and *mixed set (HLA and DDA) attribute*. For classifying each HLA, they trained a binary SVM with a histogram intersection kernel. They concatenate the confidence scores from all these attribute classifiers into a 40-dimensional vector to represent a video. For the DDA, they learned a dictionary of size 457 from all video features using KSVD and each video is represented by a 457-dimensional sparse coefficient vector. For the mixed set, the attribute set is obtained by combining HLA and DDA sets.

They compared the performance of features based on selected attributes with those based on the initial attribute set. For all the attribute-based features, they used an SVM with a Gaussian kernel for classification. Compared with the initial attribute set, the selected attributes have greatly improved the classification accuracy, demonstrating the

proposed method's effectiveness for selecting a subset of discriminative attributes. They show that DDA are complementary to HLA, and together they offer a better description of actions. They showed the benefits of selecting discriminative attributes and removing noisy and redundant attributes.

4.4 ATTRIBUTE REPRESENTATION FOR MULTI-CHANNEL TIME-SERIES HAR

Attributes are human-readable terms that describe an activity's inherent characteristics. For example, collections of verbs and objects—similar to elements of a sentence—represent human actions from images and videos. In the context of M-HAR, datasets are highly imbalanced, especially towards the "NULL" class—this class covers all the human actions that are not relevant for the task, sometimes involving more than 75% of the recorded data. An attribute representation is beneficial for solving this case, where sequences of the "NULL" or "BACKGROUND" class provide good material for learning shared attributes contained in less frequent classes. Activity classes with more frequent samples can provide attributes to lesser frequent classes.

Datasets do not possess annotated attribute representations in the context of M-HAR, and having introduced the attribute representation and the attribute-based deep networks, let define the term attribute representation for HAR.

Postulate 1. ***Attribute Representation for HAR:*** *Consider for M-HAR, an annotated task from a domain $\mathcal{D}$ with N tuples $\{(\mathbf{X}, y)_n\}_{n=0}^{\mathsf{N}}$ of N segmented sequences $\mathbf{X} \in \mathcal{X}^{[W,S]}$ from an arbitrary multi-channel Time-Series space $\mathcal{X}$ (according to Subsection 2.2.1) and $y \in \mathcal{Y}$ their respective label space from a set of Y number of classes. The idea is to learn the objective predictive function $\mathbf{f}(\cdot) : \mathcal{X} \to \mathcal{Y}$., introducing an additional mapping, an attribute representation $\mathcal{A}$, i.e., $\mathbf{f}(\cdot) : \mathcal{X} \to \mathcal{A} \to \mathcal{Y}$.*

Cheng et al. [CGD+13] presented a Zero-Shot Human Activity Recognition (Zero-Shot HAR) method for multi-channel Time-Series data. Zero-shot learning for M-HAR can be thought of as having users provide a one-time description of an unseen activity using semantic attributes. They extended previous work by proposing a framework for multi-channel Time-Series data using a Conditional Random Field and comparing it to supervised learning with limited training data. Their method is the first based on attribute representations to recognise human activities, even in the case of non-existing training samples—addressing the problem when annotated data is hard to obtain. The authors proposed two categories of labels, high-level activities $\mathcal{Y}$, and mid-level semantic attributes $\mathbf{a} \in \mathbb{B}^{\mathsf{M}}$. Moreover, Cheng et al. provided a mapping among the activities and attributes through an attribute representation $\mathrm{A}^{[\mathsf{Y},\mathsf{M}]}$, what they called *Activity-Attribute*

Matrix. The matrix $\mathbf{A}^{[Y,M]}$ is an auxiliary input that is substituted for annotated sensor data by encoding the correlation between the attributes and the seen/unseen activities. This matrix is manually defined by common-sense knowledge and domain knowledge as an initial attempt toward zero-shot learning. They hypothesise that a user can create a new attribute representation for a new activity class by simply describing it using its semantic attributes, thus, just adding a new row into the matrix. Their validation scheme recognises unseen activity classes with no training samples.

The authors combined a probability distribution of an attribute **a** given an input sequence, the temporal dependency of neighbouring attributes predictions, and the correlation between an activity class and an attribute presentation. Thus, their model exploits the strong temporal dependency of sequential data and their semantic information, and it learns the relationship between input sequences and activities through a semantic attribute layer. Besides, this model enables the re-use and generalisation of learned attribute models for recognising unseen new activities. Furthermore, they addressed transfer learning among domains, reducing the need for annotated training material.

They proposed a Leave-Two-Class-Out Cross Validation (LTCOV) for evaluating their method. From Y activity classes, they train their system with data containing $[Y-2]$ activity classes and test on the remaining activity classes, which are "unseen" to their model. They also consider the case when a small fraction of the Y-remaining classes is used for training, denoted by Y-shot learning. This method assumes samples from the same multi-channel time-series Space, i.e., $\mathcal{X}_{target} = \mathcal{X}_{source}$.

The authors in [CSG+13] extended the work of [CGD+13] proposing a two-layered zero-shot learning method and an active learning algorithm for M-HAR, using attribute representations. As many human activities share the same underlying semantic attributes, the statistical model of an attribute can be transferred among classes. They hypothesise that it is better to use a nameable attribute that allows humans to describe a context type even without sensor-data recording and annotation. M-HAR has not seen advantages of attribute representations until then—only for video-based HAR. It has not been shown which representations or attributes are useful for recognizing human activities from sensor data. They designed a new representation of human activities by decomposing high-level activities into combinations of semantic attributes where each of them is a readable term that describes a basic element or an intrinsic characteristic of an activity. These attributes are computed based on low-level temporal features, which capture the dynamic relationships in data. They extended their previous work in active learning by designing an outlier aware, active learning algorithm and a hybrid stream- and pool-based sampling scheme suitable for M-HAR scenarios. Their method is called *NuActiv*. It is designed for scenarios independent of OBD types. They focused on activities in sports because their activities are well-defined, repeatable, and of lower variation among different

subjects. They computed statistical features: mean, Standard Deviation (SD) per channel, pairwise correlation among channels, local slope—using the first- and second-order finite derivative linear regression—, and the zero-crossing rate in all three axes; additionally, the time of the day. The authors proposed active learning, where users are asked to give feedback or annotate when highly uncertain predictions. New annotations are then used for re-training and updating the models for attribute detection and M-HAR. Their system is divided into three parts: feature extraction, Attribute-based Multi-channel Time-Series HAR (Attr-based M-HAR), and active learning. The Attr-based M-HAR is divided into two parts: attribute prediction—mapping the segmented sequence input $\mathbf{X}$ to an attribute vector $\mathbf{a}$— and HAR—mapping the attribute vector $\mathbf{a}$ to an activity class y, even to unseen classes. In the active learning part, they estimate the uncertainty of predictions. For uncertain predictions, annotators are asked to provide the activity class label. The labels are then used for re-training and updating models for attribute prediction and HAR.

The attribute representation $\mathbf{A}^{[K,M]}$ encodes the human knowledge of the relationship between an activity class K and its associated set of semantic attributes $\mathbf{a}$. For K activities and M attributes, the activity-attribute matrix A is of size $[K \times M]$, where the value of each element $\mathbf{a}_{k,m}$ represents the level of association between activity k and attribute m, with binary values indicating whether such an association exist or not.

The authors considered attributes not only as sub-activities themselves but also descriptions, characteristics, or consequences of activities. For a given feature vector $\mathbf{z}$—extracted features—, they infer the presence of an attribute $\mathbf{a}$; that is, $p(\mathbf{a}|\mathbf{z})$. They divided their training data into positive and negative samples for each attribute. To learn an attribute predictor, they re-use the existing training data by merging the annotated data of all activity classes that are associated with the attribute as the positive set. With this training data, a binary classifier is trained for each attribute. After the training phase, there is an attribute predictor per attribute a_m. They use a SVM classifier; see Equation 2.2.1. The authors have trained an SVM classifier per attribute specified in the activity-attribute matrix.

The authors deploy a Nearest Neighbour Approximation (NNA) to recognize the high-level activity class given an attribute vector generated from the attribute SVM-based predictors. Specifically, the activity recognizer takes an attribute vector $\mathbf{a}$ as input and returns the closest high-level activity $\hat{y}$, represented in the attribute representation $\mathbf{A}$—the attribute representation $\mathbf{A}$ essentially provides the information of $p(y|\mathbf{a})$, as specified in Subsection 2.4.1. Following this, their idea is to keep the advantages of both feature-based HAR; precisely, by classifying a sample belonging to the seen training dataset, one can directly apply a feature-based classifier to recognize the activity. On the other hand, for samples that belong to unseen classes, one applies Attr-based M-HAR reusing known attributes from seen classes. They consider this approach as an anomaly detection

problem. Samples from unseen activity classes are like an anomaly because they were not seen before by the system. They train an unseen class detector using the one-class SVM classifier, where only the positive samples, i.e., all samples that belong to the seen classes, are given to the classifier. This SVM estimates whether a segmented sequence is part of the seen activity class set. They deploy a feature-based HAR for a sample of the seen activity set. For a sample of the unseen activity, they deploy the attribute-based classifier, following the DAP, see Subsection 2.4.1.

The authors used their system to include user feedback, using the idea of uncertainty sampling in the field of active learning. The system asks a user for activity class annotations only when it is highly uncertain about its recognition results. The system selects a pool of uncertain windows according to: the least confident predictions, thresholding between distances to top-2 activity class predictions, maximum entropy, and borders of activity regions in feature space.

The authors in [AR17] created a hierarchical representation of human activities based on semantic descriptions for smart-home applications. However, there is no explicit definition of manual activities which is adaptable or transferable to represent the level of diversity that characterises human labour in warehousing. They investigated the challenges of improving the recognition of Zero-Shot HAR in a smart-home environment by better exploiting the hierarchical taxonomy of complex daily activities. They develop a hierarchy of classifiers that incorporates a cluster tree built on the domain knowledge from training samples.

They hypothesised that clustering activities into a hierarchical activity taxonomy could facilitate the analysis of activity patterns from multiple similar data sources, and such taxonomy can also be used to scale activity recognition. Therefore, they first determined the specific positions of the activity labels in the hierarchy using a few annotated samples, which were then combined with annotated samples of similar activities to initiate the taxonomy of learning for the new activity model.

They made the following contributions. First, they designed and represented human activities using semantic attributes. Second, they postulated a data-driven approach that calculates the similarity of predefined and unseen activities and helps generate complex activity taxonomy by creating a hierarchical cluster of activity labels generated from available datasets. Third, they employed this postulated hierarchical complex activity taxonomy, built a hierarchical classification tree and proposed an uncertainty metric to distinguish unseen activities from previously seen ones. Finally, employing the uncertainty metric, they augment transfer learning and activity learning to minimise the user intervention and maximise the activity recognition performance gain.

4.5 DISCUSSION

Semantic attribute representations have been exploited in different classification and spotting tasks. Different levels of attributes have been proposed. For example, the M-HAR scenarios addressed by Cheng et al. [CGD+13, CSG+13] consider problems where recordings belong to the same domain, i.e., multi-channel time-series Space. Besides, the mapping of attribute representations to the activity classes is fixed; that is, there is no attribute annotation per sample level. This drawback is a limitation as humans will hardly repeat the same short movements representing an activity. Thus, a different set of attributes might represent the same activity. Moreover, the authors claimed that an attribute representation could be manually given. However, this is valid under the assumption of having the same multi-channel time-series Space and a detailed description of the recorded activity. This limitation is addressed, for example, in the word spotting community, as words' characters and their occurrence in horizontal spatial regions represent all the images of specific words.

5 ATTRIBUTE-BASED TRANSFER LEARNING FOR MULTI-CHANNEL TIME-SERIES HAR

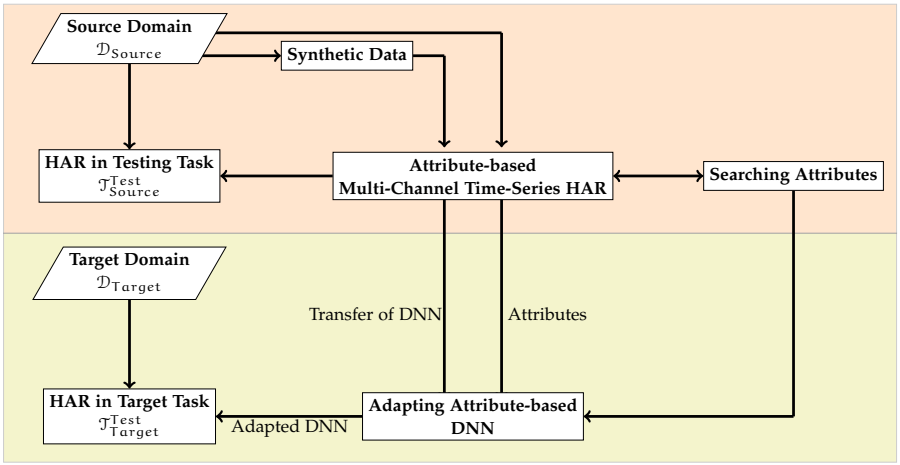


Figure 5.0.1: Attribute-based Transfer Learning for Multi-channel Time-Series HAR (Attr-based Transfer Learning for M-HAR) under different domains, $\mathcal{D}_{Source} \neq \mathcal{D}_{Target}$; it uses an attribute-based transfer learning and a DNN for parameter transfer. The DNN is adapted for processing human recordings from a variate amount of OBDs for full-transferability.

The multi-channel time-series Human Activity Recognition (M-HAR) task classifies human movements recorded by sequential data from multiple channels, for example, from sensors integrated in OBD.[1] M-HAR can be solved following the standard pattern recognition pipeline: data recording, preprocessing, feature extraction, feature-dimensionality reduction and classification, as Subsection 2.2.3 presents. Besides, M-HAR can be solved using DNNs that combines the last three steps in an end-to-end approach. DNNs are currently the state-of-the-art method for M-HAR, as Section 3.1 shows. However, the models

1 Here, M-HAR generalises Sensor-based Human Activity Recognition (Sensor-based HAR), as multi-channel Time-Series are recorded from different technologies, e.g., OBD or marker-based MoCap.

require a substantial amount of annotated data. Annotating data for M-HAR demands a huge amount of resources, i.e., human effort, time and economic expenses—Chapter 7 will discuss the annotation process of M-HAR. Transfer learning allows to re-use, squeeze and extend the usability of annotated data from source domains and trained models to related problems, i.e., target domains, as Section 2.3 shows. However, there is still an open problem of predicting activities under different domains $\mathcal{D}_{\mathrm{Source}} \neq \mathcal{D}_{\mathrm{Target}}$, i.e., different multi-channel time-series Spaces $\mathcal{X}_{\mathrm{Source}} \neq \mathcal{X}_{\mathrm{Target}}$ and different tasks, $\mathcal{T}_{\mathrm{Source}} \neq \mathcal{T}_{\mathrm{Target}}$, without the need for annotated data from the target domain, following Def. 3, in Section 2.3. This problem can be referred to as Zero-Shot M-HAR under different domains.

For example, consider the problem of performing M-HAR for different logistics scenarios. Using a trained method for M-HAR is not directly possible, as these scenarios might vary, with different subjects,[2] number of OBDs and their locations on the human body, and recording settings, i.e., $\mathcal{D}_{\mathrm{Scenario\,A}} \neq \mathcal{D}_{\mathrm{Scenario\,B}}$. For a new target scenario, the standard M-HAR system might be directly followed, as described in Section 2.2, starting from a dataset creation; that is, carrying out different recordings and subsequently a manual annotation and revision process. Semi-automated annotation methods might speed up the annotation process. However, a pre-trained method from the target domain $\mathcal{D}_{\mathrm{Target}}$ is necessary. Chapter 7 presents a practical example of a semi-automated annotation process.

This new M-HAR system might use a known M-HAR system adapting its annotated data material and its pre-trained M-HAR model. This adaptation is a type of Transfer Learning for M-HAR. It has to consider the differences in recording settings of the two or more scenarios, e.g., number and type of sensors, sampling rate, preprocessing, and M-HAR method. Following the definitions in Section 2.3, one defines a set of $\mathcal{T}_{\mathrm{Source}}$ source tasks from a source domain $\mathcal{D}_{\mathrm{Source}}$, a $\mathcal{T}_{\mathrm{Target}}$ target task from a target domain $\mathcal{D}_{\mathrm{Target}}$, where $\mathcal{D}_{\mathrm{Source}} \neq \mathcal{D}_{\mathrm{Target}}$ and $\mathcal{T}_{\mathrm{Source}} \neq \mathcal{T}_{\mathrm{Target}}$. One seeks to predict the activity classes of $\mathcal{T}_{\mathrm{Target}}$ by means of learning the objective predictive function $\mathbf{f}_{\mathrm{Target}}(\cdot)$ transferring experience from a source task $\mathcal{T}_{\mathrm{Source}}$ with a minimal portion of or without the usage of annotated data from $\mathcal{T}_{\mathrm{Target}}$.

This thesis proposes Attribute-based Transfer Learning for Multi-channel Time-Series HAR (Attr-based Transfer Learning for M-HAR) under different domains. This transfer learning method uses a semantic attribute representation $\mathcal{A}$ and a DNN suitable for M-HAR under different domains. To be precise, it develops a method for transfer learning for any target M-HAR domain, $\mathcal{D}_{\mathrm{Target}}$, independent of their activities and sensor settings. It proposes an attribute-based classification considering multiple attribute representations

2 Subjects have a wide range of physical characteristics.

per activity class. This Attr-based M-HAR uses a DNN architecture and an attribute representation for processing and classifying recordings of segmented multi-channel Time-Series of humans performing activities. The indicated architecture adapts to data of different multi-channel time-series Spaces, mapping input sequences to a semantic attribute representation. The DNN and the semantic attribute representations allow transferability to target domains. It copes with the different number of devices, duration variations and disjoint activity classes. Additionally, an algorithm is proposed for learning the attribute representation that is better suitable for solving M-HAR. Figure 5.0.1 presents the overall method.

Furthermore, the Attr-based Transfer Learning for M-HAR approach takes advantage of a large dataset composed of OBDs and human pose recordings and fine-grained annotations of semantic attributes. The DNN will process sequences of movements from the human limbs, either from poses or inertial measurements. In addition, synthetic data for initialising a deep model is proposed. Synthetic On-body Device (SOBD) data will be derived from sequences of human poses. This derivation takes advantage of large human-pose datasets as source domains. Specifically, these datasets consist of recordings from marker-based MoCap or datasets intended for video-based HAR and human pose estimation—the annotations of pixel coordinates as sequences of human joints. In the following, each of the parts of the Attr-based Transfer Learning for M-HAR is presented.

5.1 ATTRIBUTE-BASED MULTI-CHANNEL TIME-SERIES HAR

Considering the usage of a DNN for M-HAR, a question arises whether there is a DNN that can be directly applied to new segmented sequences data from different target domains to obtain some initial classification. This initial classification should not be that far from a classification using a supervised method, trained with annotated data from the target scenario. This classification method can be regarded as a sort-of generic transfer learning for M-HAR method. An initial classification is of use, for example, to a semi-automated annotation approach for M-HAR, reducing annotation effort without sacrificing consistency; see Chapter 7.

This transfer learning problem follows the Def. 3, in Section 2.3, and belongs to a Uninformed Supervised Transfer Learning (US Transfer Learning) problem. This generic method is divided into two parts of the transferable knowledge: first, the parameter-transfer via a DNN, following Subsection 2.3.4, and second, the feature-representation transfer by means of a semantic attribute representation, following Subsection 2.3.4. These two transferable parts are contained in a DNN. This DNN is capable of creating a representation from recordings of humans performing an activity, independent of the

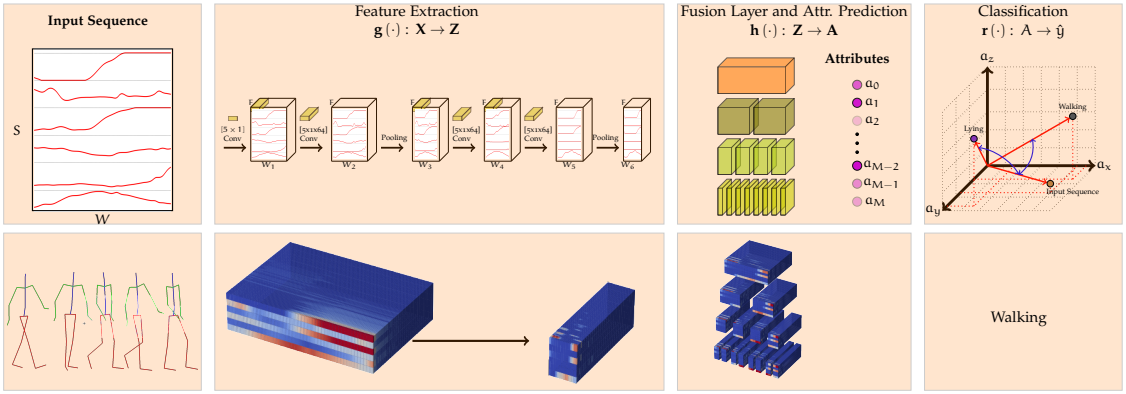


Figure 5.1.1: Attr-based M-HAR: a DNN processes a segmented multi-channel Time-Series sequence creating a fixed size temporal representation of the sequence that copes under the case of different domains of the input sequence and computing a semantic attribute vector representing the input sequence. Classification is carried out using Generalized Direct Attribute Prediction (GDAP) with the cosine or PRM similarity metrics in the attribute space.

recording settings. This representation is interpretable as a sort of semantics that describe the performed activity. The Attr-based M-HAR deploys the DNN and these semantics on a $\mathcal{T}_{Target}$. Classification is carried out by the Generalized Direct Attribute Prediction (GDAP), extending DAP from [LNH14].

Usually, parameter transfer of DNN means transferring the convolutional filters from the feature extractor, as Section 3.4 presents. One transfers a different number of layers, where the first layers contain general filters, and the deeper layers become more task-oriented. However, the fusion layers—usually a MLP and the classifier—are not transferable due to differences between the source and target domains. A source task consists of a set segmented sequences $\mathbf{X}_S \in \mathcal{X}_{Source} \subset \mathbb{R}^{[W_{Source}, N_{Source}]}$ of a source domain $\mathcal{D}_{Source}$, and $\mathbf{X}_T \in \mathcal{X}_{Target} \subset \mathbb{R}^{[W_{Target}, N_{Target}]}$ of a target domain $\mathcal{D}_{Target}$.

These layers are usually trained from scratch, and the convolutional layers are fine-tuned. The fusion layer and classifier change according to the M-HAR task, i.e., different human activities $\mathcal{Y}_{Source} \neq \mathcal{Y}_{Target}$, and to the different multi-channel time-series Spaces, as these vary due to number of OBDs, or the window length, $\mathcal{X}_{Source} \neq \mathcal{X}_{Target}$.

The proposed DNN architecture responds to these transfer limitations. This architecture is called Attr-IMU-tCNN. The Attr-IMU-tCNN adapts to the human body instead of the amount of OBDs. The architecture copes with the different number of OBDs, sequence

lengths, and disjoint activity classes. The architecture is composed of parallel branches that process segmented multi-channel Time-Series sequences from OBDs per human limb. The fusion layer creates a fixed-size representation, which contains temporal relevant information from the activity. These branches compute a representation per limb, independently of the amount of OBDs, creating a temporal fixed-size representation. Finally, the DNN predicts a semantic attribute vector. The classification is based on a similarity between attribute vector predictions and a given attribute representation $\mathcal{A}$ from the dataset. The Attr-based M-HAR is then divided into four parts: the feature extractor, the fusion layer, the attribute prediction and the classification, as Figure 5.1.1 illustrates. In the following, the Attr-IMU-tCNN and the GDAP are presented in detail.

5.1.1 *Attribute-based IMU-Temporal Convolutional Neural Network (Attr-based IMU-tCNN)*

The Attr-IMU-tCNN architecture processes multi-channel Time-Series sequences of human movements, and maps them into an attribute representation. The architecture is built based on a sensor set-up where an individual wears multiple devices. Figure 5.1.1 presents the Attr-IMU-tCNN as the function $\mathbf{g}(\cdot)$. The Attr-IMU-tCNN is based on the tCNN [HHP16, OR16, YNS+15], and it has been partially presented by the author of this thesis in [GLMR+17, MRGF+18, MRF18, MRF21]. It uses temporal-convolution layers for finding temporal-local features in the input data and fully-connected layers for connecting all of these local features, creating a label representation of the data. However, this architecture contains multiple parallel processing branches, following the idea of wider rather than deeper networks. This architecture is called Attr-IMU-tCNN. It processes segments of multi-channel Time-Series from different OBDs located on the human limbs separately. The architecture proposes parallel computing blocks for processing the sequences and for deriving an intermediate temporal feature-representation of OBDs per human limb. Dividing the OBDs recordings according to human limbs has been explored in [DLGY12, KY18]. The architecture concentrates on five human limbs: the arms, legs, and torso. The temporal feature representations from the human limbs are fused to compute a global representation of a segmented sequence $\mathbf{X}$. In M-HAR, the input of tCNNs consists of segmented sequences $\mathbf{X} \in \mathcal{X}^{[W,S]}$ from S different sensors, or channels, for a certain temporal duration W. The parallel blocks introduce an invariance relative to human limbs, as human movements vary independently from them.

The Attr-IMU-tCNN is divided into the feature extractors, the fusion layers, and the attribute predictor. The feature extractor contains temporal convolutional and spectral pooling operations, in contrast to [GLMR+17, OR16, YNS+15] where max-pooling operations were used. The spectral pooling follows the conclusions in [MRF18] showing

that stride-based pooling operations negatively affect M-HAR performance. Figure 5.1.1 explains the spectral pooling. Temporal convolutional filters extract relations from their local neighbourhood from the segmented sequences per channel at different temporal locations. Temporal filters are shared among all the channels, or sensors per OBD. These temporal relations are likely to be correlated independent to the type of sensor. This assumption is valid if the measurements per channel are normalized. By stacking convolutional layers, and downsampling their outputs, tCNNs extract more complex and abstract features and are task-dependent, being invariant to distortions and time translations [LKF10, MRGF+18, YNS+15]. tCNNs extract hierarchical human body movements, i.e., from basic and simple movements to complex ones. Besides, they learn the temporal dependencies among different movements. Figure 5.1.2 shows the Attr-IMU-tCNN.

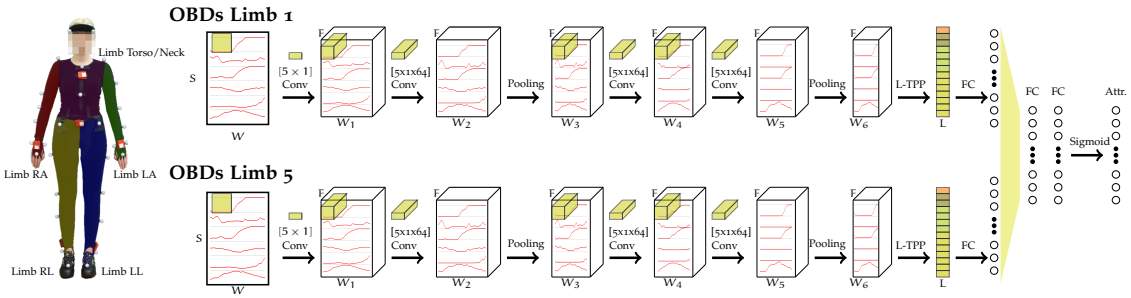


Figure 5.1.2: The Attr-IMU-tCNN comprises five parallel branches, corresponding to recordings of all of the OBDs located on the LA, LL, RA, RL, and NT. Branches are composed of two blocks, each with two $[5 \times 1]$ temporal convolutions followed by a spectral pooling and a final L-TPP layer, computing an intermediate temporal feature-representation of the OBDs per human limb. The outputs of the blocks are concatenated and forwarded to a FC layer. The output architecture is the sigmoid function computing an attribute representation $\mathcal{A}$ from the input sequence.

The Attr-IMU-tCNN allows to transfer the experience of a source domain $\mathcal{D}_{\mathtt{Source}}$ to recordings of a different domain, $\mathcal{D}_{\mathtt{Target}}$. The architecture is meant to be used for solving a task $\mathcal{T}_{\mathtt{Target}}$ from $\mathcal{D}_{\mathtt{Target}}$ with no fine-tuning—or fine-tuned with a proportion of the target dataset. This architecture should cope with different multi-channel time-series Spaces of the source and target datasets, $\mathcal{X}_{\mathtt{Source}} \neq \mathcal{X}_{\mathtt{Target}}$, and different data label spaces $\mathcal{Y}_{\mathtt{Source}} \neq \mathcal{Y}_{\mathtt{Target}}$. For that, the architecture computes a feature representation of the sequence segments, and maps them to a fixed-size descriptor.

Feature Extractor

The Attr-IMU-tCNN is composed of parallel temporal convolutional blocks, a fusion layer, and an attribute predictor layer. The temporal convolutional blocks comprise temporal convolutional layers, spectral pooling operations, and a TPP layer. These blocks process sequences from a set of OBDs per human limb, creating an intermediate and temporal fixed-size descriptor per human limb—the fixed-size descriptor is computed independently of the number of channels or sensors of the input segmented sequence, with $\mathbf{X} \in \mathcal{X}^{[W,S]}$. Each parallel branch has a logical meaning as it represents the data of a set of OBDs from a human limb. This structure should allow for more robustness against the OBDs being sightly asynchronous or having different characteristics. Besides, the structure allows for transferability to datasets of other domains. As these OBDs are located in different parts of the body, temporal blocks process only signals that come from individual parts, increasing the descriptiveness. Furthermore, the fixed-size temporal representation is vital for subsequent transferability. The Attr-IMU-tCNN creates a representation per human limb. Specifically, each temporal block of the Attr-IMU-tCNN architecture contains four $[5 \times 1]$ temporal convolution layers with ReLU activation functions and two spectral pooling layers, organised in two blocks. Each block contains two temporal convolutional layers followed by spectral pooling operations.

In general, the Attr-IMU-tCNN has five temporal-convolutional branches for specifically Left Arm (LA), Left Leg (LL), Right Arm (RA), Right Leg (RL), and Neck or Torso (NT). Each branch has two blocks of a pair of convolutional layers followed by a spectral pooling, and a TPP layer. Temporal convolutional layers contain $C = 64$ filters of size $[5 \times 1]$. In the following, the spectral pooling and the TPP will be introduced for Transfer Learning for HAR.

Temporal Spectral Pooling

As discussed in Subsection 2.1.3, stride-based pooling functions like max-pooling encourage a spatial invariance and a capacity bottleneck. They have shown a relatively good empirical performance for object recognition using CNNs, especially, for inputs with spatial structure. Nevertheless, they imply a very sharp dimensionality reduction, e.g., by at least a factor of four, when applied to two-dimensional inputs. Moreover, the maximum value of each window only reflects very local information and does not often represent the window's contents well [RSA15]. Following the empirical findings from the author of this thesis in [MRGF+18], stride-based pooling operations affect negatively M-HAR performance. They retain information near zero frequencies in local neighbourhoods of the input, which are related to the gravity component of the inertial measurements of the human recordings [AI15, BPT14, KLLK10, LYA09, NTJ18, SSH13, SR12, WGWH18].

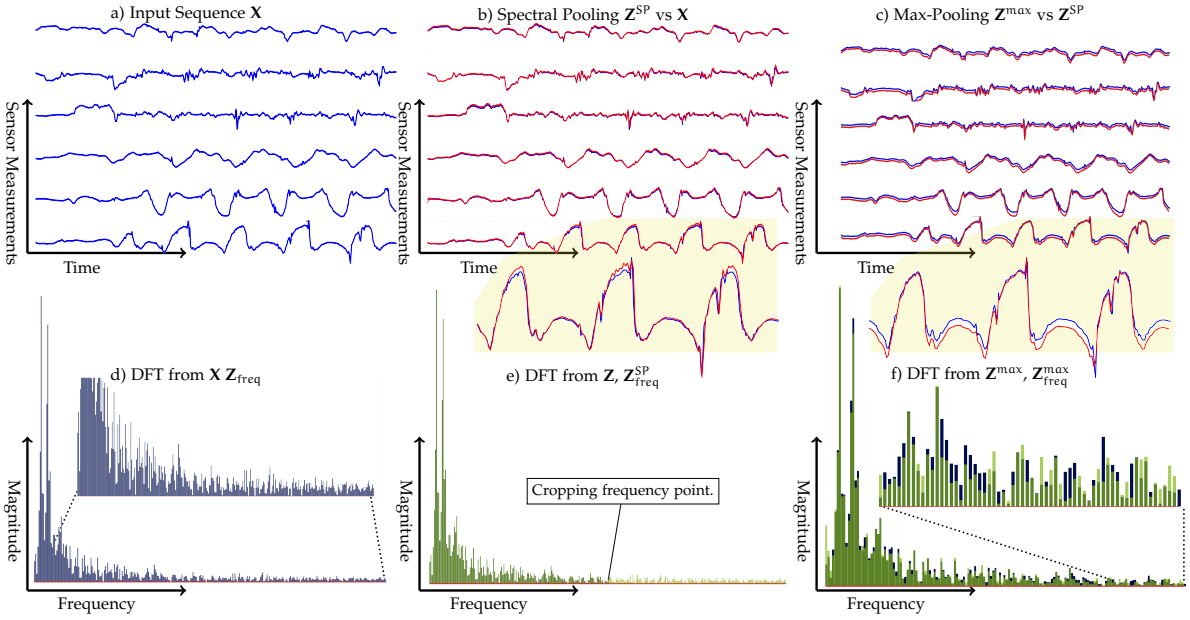


Figure 5.1.3: DFT from a input sequence $\mathbf{X}$, after a spectral pooling $\mathbf{Z}^{SP}$ and max-pooling $\mathbf{Z}^{max}$ operations. Spectral pooling crops the DFT representation of the input $\mathbf{X}_{freq}$ at H_x, e.g., at half of the frequencies as e) shows; b) compares the time reconstruction after spectral pooling $\mathbf{Z}^{SP}$, shown in ■, vs. the original input sequence, shown in $\mathbf{X}$, ■, where the amplitude of the high frequencies after H_x are zeroed out. Max pooling affects the DFT representation of the input, adding high frequency components into the lower ones, as presented in f) with $\mathbf{Z}^{SP}_{freq}$, represented in ■, and $\mathbf{Z}^{max}_{freq}$, shown in ■; c) compares the time reconstruction after max-pooling $\mathbf{Z}^{max}$, shown in ■, vs. $\mathbf{Z}^{SP}$, given in ■.

Moreover, they increase the influence of the high frequency components on the low frequencies, i.e., an aliasing effect.

A temporal spectral pooling carries out the DFT, cropping and inverse DFT per channel, i.e., along the temporal dimension, see Algorithm 1. This temporal spectral pooling is proposed for M-HAR. This pooling concentrates in the low frequencies of the feature maps, following that human motions lie in the a range of low frequencies. Furthermore, this layer removes high frequency components. This pooling follows the body movements and noise components separation based on frequency from [LYA09, KLLK10, SR12, SSH13, BPT14, AI15, NTJ18, WGWH18], but without falling in a sharp separation by stride-based poolings, and avoiding alising. The temporal spectral pooling operation allows for dimensionality reduction without sacrificing performance.

Algorithm 1 Temporal Spectral Pooling

// Feature map Input $\mathbf{Z}$: Output width W_p
procedure SPECTRALPOOLING($\mathbf{Z}_{int}^{[W_x,H_x]}, W_p$)
  // Channel-wise DFT, i.e., along the time axis
  for $h = 1 : H_x$ **do**
    $\mathbf{Z}_{freq}^{[W_x,h]} \leftarrow \mathcal{F}\left(\mathbf{Z}_{int}^{[W_x,h]}\right)$
  // Cropping along the time axis
  $\mathbf{Z}_{freq}^{[W_p,H_x]} \leftarrow crop\left(\mathbf{Z}_{freq}^{[W_x,H_x]}, W_p\right)$
  // Channel-wise inverse DFT, i.e., along the time axis
  for $h = 1 : H_x$ **do**
    $\mathbf{Z}_{int}^{[W_p,h]} \leftarrow \mathcal{F}^{[-1]}\left(\mathbf{Z}_{freq}^{[W_x ph]}\right)$
  return $\mathbf{Z}_{int}^{[W_p,W_p]}$

Temporal Pyramid Pooling (TPP)

Finally, each branch merges the extracted temporal features using an additional TPP layer. This TPP layer creates a fixed-size representation of the limb, keeping temporal relations. He et al. [HZRS15] pointed out that convolutional layers do not need fixed-size feature-map inputs since they perform a convolution operation and their filters are not fully-connected to their inputs. However, in the usual tCNN, the FC layers necessarily need fixed-size feature-map inputs. The last convolutional layer is the only one that should generate a fixed-size feature map because the first FC layer is connected to it. For that reason, He et al. [HZRS15] replaced the last pooling with a SPP layer to eliminate the need for the fixed-size input images for object classification. In the case of these sequential models, Sudholt and Fink [SF18] introduced the TPP for improving retrieval results in word-spotting with word images. In this specific example, the feature maps of *word* images are partitioned along the horizontal axis, being important following the sequential writing. Therefore, the vertical axis is not partitioned. See Section 4.1 for more details.

For M-HAR, the TPP divides an input from finer to coarser levels; it aggregates their information generating local outputs, and it concatenates these outputs into an overall feature output, see Figure 5.1.1. The TPP splits the input along the horizontal axis, for M-HAR, along the time axis. An L-level TPP splits its feature representation input into 2^l non-overlapping horizontal cells per level l. For a feature map input, $\mathbf{Z}^{[W,S]}$, a cell of level l is of size $\left[\frac{W}{2^l}, S\right]$. Each cell covers the entire vertical axis, channel, or sensor axis of the feature-map sequence. The layer TPP pools the maximum value from all the cells. The pooling is thus along the time axis, and each cell roughly represents features from consecutive intervals of the activity sequence. A pyramid representation results when stacking multiple of these poolings per cell along the time axis. The Attr-

IMU-tCNN uses a $L = 5$-level TPP layer with max-pooling, where each level indicates the number of pooling bins along the horizontal axis. The size of the TPP output is $C_{tpp} = C_{i-l} \cdot \sum_{l=0,1,\dots,L} 2^l$, where C_{i-l} is the number of filters, or depth, of the previous $i-th$ feature map input. For example, with the last convolution layer having $C = 64$ filters, the output size of the TPP layer size evaluates $C_{tpp} = 1984$. This layer differs from the IMU-tCNN from [GLMR+17, MRGF+18, MRF21], whose FC layer sequence changes according to the dataset.

Instead of scaling the network deeper, these layers are processed in parallel for all of the OBDs per human limb, increasing the network's descriptiveness. Despite the TPP layer being a very destructive layer due to the max-pooling function, the parallel branches allow learning TPP-representations per limb. This layer gives a sort of activation of individual limbs, taking care of temporal changes of a limb. The more prominent activations of the whole body do not delete the prominent activations per limb. This effect appears when using a tCNN [OR16, YNS+15], which is a IMU-tCNN with only a single branch for all of the sensors, as reported by the author of this thesis in [GLMR+17, MRGF+18].

Multi-Layer Perceptron (MLP)

The TPP-representation of the limbs are concatenated, creating a global representation of the body. This representation is of size C_{tpp} times the number of limbs, e.g., considering five limbs and 5-level TPP layer, $C_{concat.} = 5 \cdot C_{5L-TPP} = 9920$. The Attr-IMU-tCNN then combines this global descriptor into an attribute representation. The Attr-IMU-tCNN uses an MLP with a sigmoid layer for predicting an attribute representation $\mathcal{A}$. The proposed network will compute an attribute representation of an input sequence rather than classifying it using a softmax function [GLMR+17, OR16, YNS+15]. The sigmoid activation function is applied to each element of the output layer. Its output corresponds to pseudo-probabilities for each attribute a_m present in the segmented sequence, following the architecture described in the context of word spotting [SF18]; see Section 4.1.

5.1.2 *Generalized Direct Attribute Prediction (GDAP)*

For M-HAR, it might be helpful to create a common representation of actions in different scenarios with different domains; this representation will be the semantic attributes. Transfer learning boosts M-HAR performance from different scenarios using attributes, as the classifier does not need to be trained from scratch using a small target dataset but instead profits from a trained model on a large dataset. This additional mapping serves as an intermediate layer that allows sharing high-level concepts among activity classes in

$\mathcal{Y}_{Source}$ and $\mathcal{Y}_{Target}$; here, the training and testing tasks are disjointed, and the source and target domains might differ.

Following the Postulate 1, in Section 4.4, assuming for Attr-based M-HAR an annotated dataset or task $\mathcal{T}$ from an arbitrary domain $\mathcal{D}$ with N tuples $\{(\mathbf{X}, y)\}_n^N$ with $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_N]$ segmented sequences from an arbitrary multi-channel time-series Space $\mathcal{X}^{[W,S]}$ with W the length of the sequence and S the number of sensors, and $\mathbf{Y} = [y_1, y_2, ..., y_N]$ their corresponding classes from label space $\mathcal{Y}$. M-HAR seeks to learn the objective prediction function $\mathbf{f}(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$, introducing an additional mapping, specifically an attribute representation $\mathcal{A}$, i.e., $\mathbf{f}(\cdot) : \mathcal{X} \rightarrow \mathcal{A} \rightarrow \mathcal{Y}$.

Postulate 2. ***Attribute Representation for Transfer Learning for HAR****: Consider a set of L segmented sequences $\mathbf{X}_{Target}$ from a target dataset or task $\mathcal{T}_{Target}$, being L the number of samples from arbitrary target domain $\mathcal{D}_{Target}$, and a set of corresponding labels $\mathbf{Y}_{Target} \in \mathcal{Y}_{Target}$ number of classes. The task of Transfer Learning for HAR is to construct a classifier $\mathbf{f}(\cdot) : \mathcal{X}_{Target} \rightarrow \mathcal{Y}_{Target}$ by using the attribute representation $\mathcal{A}$ and the objective prediction function $\mathbf{f}(\cdot)$ from a source task $\mathcal{T}_{Source}$, even when the labels spaces $\mathcal{Y}_{Source} \cap \mathcal{Y}_{Target} = \emptyset$ and the multi-channel time-series Spaces $\mathcal{X}_{Source} \neq \mathcal{X}_{Target}$ differ. For classifying the set $\mathbf{X}_{Target}$, as classes do not have annotated data, a function $\mathbf{r}(\cdot)$ that maps the attribute representation $\mathcal{A}$ to the activity classes $\mathcal{Y}_{Target}$ is needed.*

Assuming an attribute predictor $\mathbf{g}(\cdot)$, e.g., the Attr-IMU-tCNN as the function, the output can be interpreted as a probability for each attribute being present or not in the input sequence. More formally, the attribute predictor represents a function $\mathbf{g}(\cdot) : \mathcal{X} \rightarrow \mathcal{A}$, where $\mathbf{X} \in \mathcal{X}^{[W,S]}$ is a sequence segment sample, and $\mathbf{g}(\mathbf{X})$ are the attribute estimates. The probability distribution, over the binary attribute vector $\mathbf{a} = (a_1, ..., a_M) \in \mathcal{A} \subset \mathbb{B}^M$, $\mathbf{p}_g(\mathbf{a}|\mathbf{X})$, is given by a product of Bernoulli distributions:

$$\mathbf{p}_g(\mathbf{a}|\mathbf{X}) = \prod_{m=1}^{M} p(a_m|\mathbf{X}) = \prod_{m=1}^{M} \mathbf{g}(\mathbf{X})_m^{a_m} (1 - g(\mathbf{X})_m)^{1-a_m} \tag{5.1.1}$$

Following the Postulate 1, in Section 4.4, the attribute representation becomes a layer for computing a human activity $y \in \mathcal{Y}$. In general, the maximum-likelihood estimate $\hat{y}$ of the activity class is given by,

$$\mathbf{p}(\hat{y}|\mathbf{X}) = \underset{y=1,...,\mathcal{Y}}{\operatorname{argmax}} \mathbf{p}(y|\mathbf{X}) \tag{5.1.2}$$

The activity class $y \in \mathcal{Y}$ and the sensor segment sequence $\mathbf{X}$ are conditionally independent, given the attribute vector $\mathbf{a}$, the activity class posterior can be written as,

$$\mathbf{p}(y|\mathbf{X}) = \sum_{\mathbf{a}\in\mathcal{A}} \mathbf{p}(y|\mathbf{a})\mathbf{p}_g(\mathbf{a}|\mathbf{X}) \tag{5.1.3}$$

The DNN model $\mathbf{g}(\cdot)$ computes directly $\mathbf{p}_g(\mathbf{a}|\mathbf{X})$. An estimate of the activity class $\hat{y}$ for a given data segment $\mathbf{X}$ is obtained by, first, computing $\mathbf{a}$ via an attribute predictor $\mathbf{g} : \mathbf{a} = \mathbf{g}(\mathbf{X})$, e.g., a DNN for attributes, and second, computing $\hat{y}$.

In the following, two options for modelling $\mathbf{p}(y|\mathbf{a})$ are presented; specifically, the DAP, presented in Subsection 2.4.1, for HAR, and the GDAP. Finally, a novel idea is described that does not explicitly use the probabilistic model above, but uses a classifier to model the dependency between the attribute predictor outputs $\mathbf{a}$ and activity classes y.

Recapping from Subsection 2.4.1, the DAP model assumes that each class $y \in \mathcal{Y}$ has an associated unique attribute representation $\mathbf{a}^{(y)}$. There is a deterministic relationship $h(y) = \mathbf{a}^{(y)}$ so that $\mathbf{p}(\mathbf{a}|y) = [[\mathbf{a} = \mathbf{a}^{(y)}]]$. From Bayes' theorem, one can rewrite $\mathbf{p}(y|\mathbf{a}) = \frac{p(y)}{\mathbf{p}(\mathbf{a})}\mathbf{p}(\mathbf{a}|y)$. So, for a segmented sequence $\mathbf{X}$, Equation 5.1.3 becomes

$$p(y|\mathbf{X}) = \sum_{\mathbf{a}\in\mathcal{A}} \frac{p(y)}{p(\mathbf{a})} p(\mathbf{a}|y)\mathbf{p}_g(\mathbf{a}|\mathbf{X}). \tag{5.1.4}$$

The normalization factor $\mathbf{p}(\mathbf{a})$ is constant w.r.t. the activity class. Furthermore, Lampert et al. [LNH09, LNH14] propose to use a uniform class prior $\mathbf{p}(y)$. Thus, the factor $\frac{p(y)}{p(a)}$ can be ignored for classification, hence the expression simplifies to:

$$\mathbf{p}(y|\mathbf{a}) \propto \sum_{\mathbf{a}\in\mathcal{A}} [[\mathbf{a} = \mathbf{a}^{(y)}]]\mathbf{p}_g(\mathbf{a}|\mathbf{X}) \tag{5.1.5}$$

$$= \mathbf{p}_g(\mathbf{a}^{(y)}|\mathbf{X}), \tag{5.1.6}$$

where $\mathbf{a}^{(y)} = h(y)$ is the unique attribute representation of class y.

Unfortunately, in M-HAR, activity classes do not typically have a unique attribute representation. For example, the activity class *take* can involve the attribute *left hand* and/or *right hand*. There exists, however, a deterministic relationship in the opposite direction in M-HAR, i.e., each attribute vector $\mathbf{a}$ corresponds to exactly one activity class y; for example, in case of the LARa dataset, see Subsection 6.1.1. This mapping $\mathbf{r}(\mathbf{a}) = y$

can either be defined in advance from prior domain knowledge, or estimated from training data. The distribution becomes $\mathbf{p}(y|\mathbf{a}) = [[y = \mathbf{r}(\mathbf{a})]]$, which can be inserted into Equation 5.1.2 and Equation 5.1.3. This case is referred to as GDAP, and the maximum-likelihood estimate is computed as

$$\hat{y} = \underset{y=1,\dots,Y}{\operatorname{argmax}} \sum_{\mathbf{a}\in\mathcal{A}|\mathbf{r}(\mathbf{a})=y} \mathbf{p}_g(\mathbf{a}|\mathbf{X}). \tag{5.1.7}$$

The maximum-likelihood in Equation 5.1.7 can be further approximated to

$$\hat{y} = \mathbf{r}(\hat{\mathbf{a}}), \tag{5.1.8}$$

here, $\hat{\mathbf{a}}$ is computed as

$$\hat{\mathbf{a}} = \underset{\mathbf{a}\in\mathcal{A}}{\operatorname{argmax}}\, \mathbf{p}_g(\mathbf{a}|\mathbf{X}). \tag{5.1.9}$$

Besides, $\hat{\mathbf{a}}$ can be computed by means of a Nearest Neighbour Approximation (NNA). More precisely, the authors in [MRF18] do not compute Equation 5.1.9 but,

$$\hat{\mathbf{a}} = \underset{\mathbf{a}\in\mathcal{A}}{\operatorname{argmin}}\, d(\mathbf{a} - \mathbf{g}(\mathbf{X})), \tag{5.1.10}$$

with $d(\cdot)$ $d_{Euc} = \{\|\cdot\|_2\}$,—after normalizing $\mathbf{a}$ and $\mathbf{g}(\mathbf{X})$— $d(\cdot) = d_{cos}$, see Equation 4.1.1, and $d(\cdot) = d_{PRM}$, see Equation 4.1.2. The d_{Euc} and d_{cos} assume that the sum in Equation 5.1.9 is dominated by its largest terms.

5.2 LEARNING ATTRIBUTE REPRESENTATIONS FOR M-HAR

Only annotations exist concerning coarse human activities, e.g., walking, jumping, and running, commonly for long-duration activities. However, there do not exist annotations of small and granulate activities—lesser than a second—that could describe a coarse activity. Thus, attribute representations do not exist for common M-HAR datasets—as an exception, the LARa dataset, Subsection 6.1.1, with sample-based attribute annotations. Furthermore, humans can not provide annotations easily by only observing the data; for example, when they observe activities on images or videos—see Chapter 7 for details of the annotations process. A search for suitable attributes that represents signal segments is presented.

Considering the literature on random subspace projection and random indexing for representing words, phrases and documents, a set of different sequence segments could be projected to or described by a random representation. This random representation should be adequate, such that one can differentiate the activities. However, as unknown attributes span this random space, it might not be suitable. Therefore, one must search for the best possible representation that adequately describes sequence segments. An evolutionary algorithm can perform this search, where attribute representations are seen as genotypes of the classes. A genotype is a code or vector that represents an individual, e.g., the human genome. Here, a binary vector represents an activity class. A proper representation of the classes can be learned by evaluating, selecting, and mutating those genotypes.

Algorithm 2 Evolutionary algorithm for finding $\mathbf{r}(\cdot)$

// Input niter: number of iterations, noffsprings: number of offsprings, K: number of classes, M: number of attributes, TrainSet: training set, ValSet: validation set
procedure EVOLUTION($niter, noffsprings, K, M, TrainSet, ValSet$)
// Draw initial generation of attribute representations $A_{parent1}, A_{parent2} \subset \mathbb{B}^{[K,M]}$
$A_{1-gen}^{parent1} \leftarrow random(K)$
$A_{1-gen}^{parent2} \leftarrow random(K)$
$F_1^{best} \leftarrow 0.0$
// Evaluate attribute representation generations
for $n = 1 : niter$ **do**
$\{A_{n-gen}^{(os)}\}_{os=0}^{noffsprings} \leftarrow mutate\left(breed\left(A_{n-gen}^{parent1}, A_{n-gen}^{parent2}\right)\right)$
// Evaluate each offspring
for $i = 1$:noffsprings **do**
$network \leftarrow trainCNN\left(TrainSet, A_{gen}^{(i)}\right)$
// Compute the $A_{gen}^{(i)}$'s fitness
$\{F_1^i\} \leftarrow testCNN\left(network, ValSet, A_{n-gen}^{(i)}\right)$
$Sorted\_Offpsrings \leftarrow \underset{i=1,\ldots,noffsprings}{argsort} \{F_1^i\}$
// Selecting the top 2 offsprings for next generation
if $Top2\ (Sorted\_Offpsrings) > F_1^{best}$ **then**
$A_{n-gen}^{parent1} \leftarrow A_{n-gen}^{(Top_1)}$
$A_{n-gen}^{parent2} \leftarrow A_{n-gen}^{(Top_2)}$
$F_1^{best} \leftarrow F_1^{(Top_2)}$
// Best attribute parent
return $A_{n-gen}^{parent1}$

Algorithm 2 shows the Evolutionary Algorithm (EA) that finds the best attribute representation of a certain dataset. The EA seeks a representation that performs well for M-HAR. The EA initially assigns a representation to human activities, evaluates the representation using the performance on the validation set as a fitness metric, and mutates

the representations with the best fitness values. The algorithm evaluates the fitness of an attribute representation by training and validating a DNN. The validation $F1_w$ serves as the performance metric; Section 6.2 presents the metric in detail. First, the NNA classifies all the segmented windows comparing predictions of the Attr-IMU-tCNN and the attribute representation $\mathbf{A}_{gen}$. The NNA uses the cosine distance, Equation 4.1.1, between the computed attributes $g(x)$—with $g(\cdot)$ being the DNN and $\mathbf{X}$ an input sequence—and the target attributes $\mathbf{a} \in \mathbf{A}_{gen}$, finding class predictions. Second, the EA computes the precision and recall of the validation predictions. Finally, the EA mutates $\mathbf{A}_{gen}$. The algorithm mutates and evaluates $\mathbf{A}_{gen}$ until selecting the best attribute representation for a certain number of iterations starting from a random representation. A global mutation in $\mathbf{B}^M$ is selected to change the attribute representation, i.e., the attribute a_i for $i = 1, \cdots, N$ of a single target attribute representation $\mathbf{A}$ flips with probability $p_i \in (0, 1)$.

5.3 SYNTHETIC DATA FOR M-HAR

Following [GSF18] for word-spotting and [HKA+18] for human pose estimation, using synthetic data for enlarging limited datasets or for pretraining DNN is beneficial for classification and regression. Inertial measurements are computed by using the derivative of sequences of joint poses of a human, e.g., from marker-based MoCap. These derivatives will act as a sort of Synthetic On-body Device (SOBD), which are located on the human. The SOBDs will provide linear acceleration $\mathbf{a}$ and angular velocity $\boldsymbol{\omega}$ of the human joints.

From Equation A.2.6, the linear acceleration is computed given by a combination of the linear acceleration and the angular rotation of the joint with respect to a global or given origin frame {{O}}[3] [Crao5].

$$\begin{Bmatrix} O \\ J \end{Bmatrix} \mathbf{a}_j(t) = \begin{Bmatrix} O \\ J \end{Bmatrix} \mathbf{a}(t) + \begin{Bmatrix} O \\ J \end{Bmatrix} \boldsymbol{\omega}(t) \times \left(\begin{Bmatrix} O \\ J \end{Bmatrix} \boldsymbol{\omega}(t) \times^{\{O\}} \mathbf{p}_j(t) \right) + \begin{Bmatrix} O \\ J \end{Bmatrix} \dot{\boldsymbol{\omega}}(t) \times^{\{O\}} \mathbf{p}_j(t), \tag{5.3.1}$$

with $\begin{Bmatrix} O \\ J \end{Bmatrix} \mathbf{a}(t)$ the linear acceleration of frame attached to the joint $\{J\}$ with respect to a given origin frame; $\begin{Bmatrix} O \\ J \end{Bmatrix} \boldsymbol{\omega}(t)$ and $\begin{Bmatrix} O \\ J \end{Bmatrix} \dot{\boldsymbol{\omega}}(t)$ the angular velocity and acceleration of the frame attached to to the joint $\{J\}$ with respect to a given origin frame; and $\mathbf{p}_j$ the vector from joint $\{J\}$ to the given origin frame [Crao5]. Equation 5.3.1 is obtained from Equation A.2.6 with ${}^{\{B\}}\dot{\mathbf{v}}_Q = 0$ Section A.2 presents, in more detail, the linear acceleration and angular velocity of a point with respect to different frames.

The second derivative of a smooth spline approximation of degree three on a small time-interval from a sequence of joint-pose estimations is deployed for computing $\begin{Bmatrix} O \\ J \end{Bmatrix} \mathbf{a}(t)$.

3 The origin frame $\{O\}$ depends on the specifications of a marker-based MoCap system.

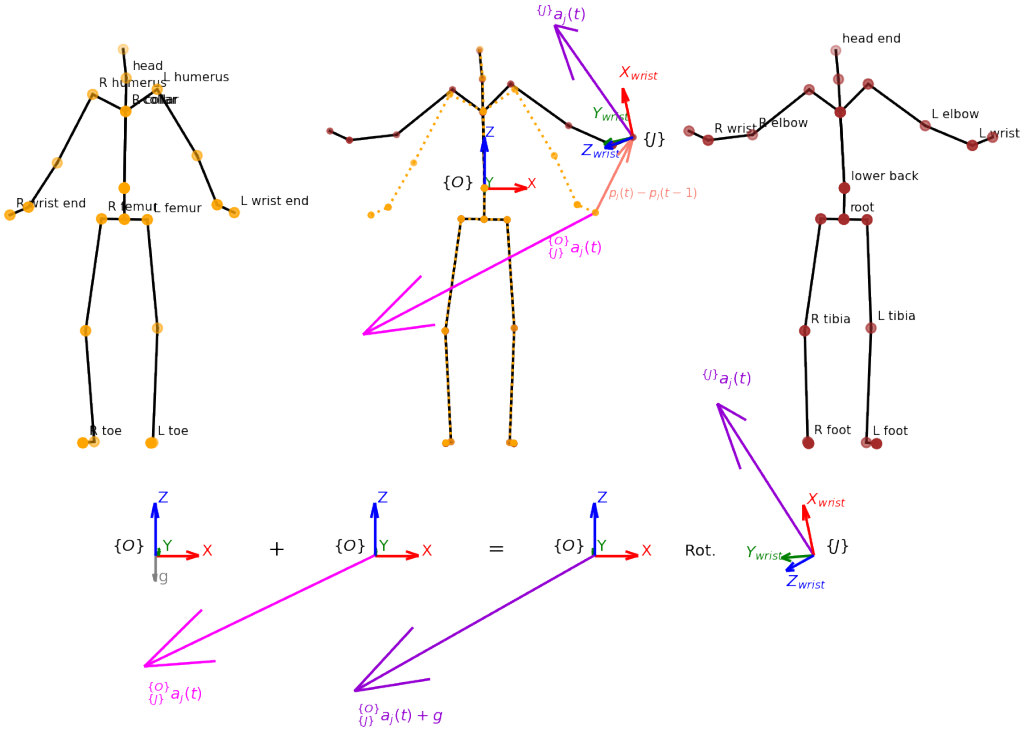


Figure 5.3.1: Example of the linear acceleration of a Synthetic On-body Device (SOBD) from marker-based MoCap from two consequetive poses of the joint {J} located on left wrist of the LARa-MoCap (LARa-M) dataset. A frame attached to joint {J} is represented by the unitarian vectors towards (X, Y, Z), specified by ■, ■, ■. The linear acceleration ${}^{O}_{\{J\}}\mathbf{a}_j\,(t)$ on origin frame is presented in ■. The vector from pose $\mathbf{p}_{\{J\}}\,(t-1)$ to pose $\mathbf{p}_{\{J\}}\,(t)$ is given in ■. The linear acceleration of the joint {J} given in origin frame O and frame {J}, including the gravity ${}^{\{J\}}\mathbf{a}_j\,(t)$ is shown in ■. The gravity vector $\mathbf{g}$ is given in ■.

Similarly, the first and second derivative of a cubic spline interpolation from a sequence of joint rotations is used for computing ${}^{\{O\}}_{\{J\}}\boldsymbol{\omega}\,(t)$ and ${}^{\{O\}}_{\{J\}}\dot{\boldsymbol{\omega}}\,(t)$. This approach differs from the SOBDs using finite differences for human pose estimation; see Section 3.3. This assumes that local temporal-neighborhoods are likely to be correlated. Specifically, a SOBDs dataset is created from the derivatives of overlapping sequences of human-joint recordings.

Finally, the sum of ${}^{\{O\}}_{\{J\}}\mathbf{a}_j\,(t)$ and gravity in the direction of Z is rotated to its corresponding joint frame. Gravity is considered as the OBDs will be used as source for real and non-preprocessed OBD. Figure 5.3.1 shows an example of a SOBD located on left wrist of the LARa-M dataset.

$$ {}^{\{J\}}\mathbf{a}_j\,(t) = {}^{\{J\}}_{\{O\}}\mathbf{R}\,(t)\cdot\left({}^{\{O\}}_{\{J\}}\mathbf{a}_j\,(t)+\mathbf{g}\right). \tag{5.3.2} $$

with ${}^{\{J\}}_{\{O\}}\mathbf{R}(t) = \mathbf{R}_z(t)\,\mathbf{R}_y(t)\,\mathbf{R}_x(t)$ the rotation to the joint frame $\{J\}$, and $\mathbf{g} = \begin{pmatrix} 0 \\ 0 \\ -9.8\mathrm{m/s} \end{pmatrix}$.

Additionally, sequences of human poses from data of different purposes are considered as source for SOBDs. The *Annotations* of pixel-coordinate sequences of human joints from video data intended for video pose estimation are used. These datasets contain annotated recordings from different scenarios with a broad range of human activities in the wild. Thus, this approach seeks to squeeze the usability of these datasets for M-HAR. These human-pose annotations from videos can be considered multi-channel Time-Series of human movements. This proposal relates to hybrid approaches of human-pose estimation and video-based HAR [LPT18, KY18], where different input types are considered for the classification task. Figure 6.3.3 shows an example of SOBDs from a video-based HAR dataset.

Algorithm 3 Online Augmentation algorithm

// Input $\mathbf{X} \in \mathbb{R}^{[W_{seg},S]}$: input sequence, β_a: variance for random noise, β_j: variance jitter, W: wanted window length
procedure AUGMENTATION ($\mathbf{X}$, β_a, β_j, W)
  // First, randomly cropping and warping it to the desired window length.
  $\mathbf{X} \leftarrow$ RANDOMCROPPINGTIMEWARPING($\mathbf{X}$, W)
  // Second, adding random jitter to each channel.
  $\mathbf{X} \leftarrow$ JITTER($\mathbf{X}$, β_j)
  // Third, adding Gaussian noise to the sequence before feed-forward to the network.
  $\mathbf{X} \leftarrow$ RANDOMNOISE($\mathbf{X}$, β_a)

  function RANDOMNOISE($\mathbf{X}$, β_a)
    // Drawing random noise from a Gaussian Distribution.
    $\mathbf{R} \leftarrow\sim \mathcal{N}(0, \beta_a)$
    $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{R}$
    return $\mathbf{X}$
  function JITTER($\mathbf{X}(t)$, β_j)
    // Drawing random noise from a Gaussian Distribution
    $t_{rnd} \leftarrow\sim \mathcal{N}(0, \beta_j)$
    $t \leftarrow t + t_{rnd}$
    // Evaluating the piece-wise approximation of $\mathbf{X}$ per sensor at time-shifted points
    $\mathbf{X} \leftarrow \mathbf{X}(t)$
    return $\mathbf{X}$
  function RANDOMCROPPINGTIMEWARPING($\mathbf{X}$, W)
    // Drawing from uniform distribution
    $W_{rnd} \leftarrow\sim \{W_{min}, \cdots, W_{max}\}$
    $\mathbf{X}^{[W_{rnd},S]} \leftarrow \mathtt{crop}\left(\mathbf{X}^{[W_{seg},S]}, W_{rnd}\right)$
    // 1D interp. for W monotonically increasing points per sensor
    $\mathbf{X}^{[W,S]} \leftarrow \mathtt{interpolate}\left(\mathbf{X}^{[W_{rnd},S]}, W\right)$
    return $\mathbf{X}$
  return $\mathbf{X}$

Furthermore, a data augmentation strategy for M-HAR is proposed. A random cropping plus time warping of segmented sequences of different lengths is deployed. Random cropping has been useful as online data augmentation for object and face recognition using CNNs [PVZ15]. Here, random cropping is proposed along the time dimension. Window-based segmentation is carried out with a slightly larger window size W_{seg} than the expected at deploying W,[4] generating $\mathbf{X} \in \mathbb{R}^{[\mathsf{W}_{\text{seg}},\mathsf{S}]}$. Windows of random sizes W_{rnd} drawn from a discrete random distribution from set $\{\mathsf{W}_{min}, \cdots, \mathsf{W}_{max}\}$ are cropped from $\mathbf{X}$, where $\mathsf{W}_{min} < \mathsf{W} < \mathsf{W}_{max}$ and $\mathsf{W}_{max} < \mathsf{W}_{seg}$. Random cropping generates sequences that are shorter or larger than the expected W, i.e., $\mathbf{X} \in \mathbb{R}^{[W_{\text{rnd}},\mathsf{S}]}$. Then, a 1D piece-wise interpolation per sensor is carried out, evaluated at W monotonically increasing discrete points. This interpolation warps the sequence $\mathbf{X} \in \mathbb{R}^{[W_{\text{rnd}},\mathsf{S}]}$ into a sequence $\mathbf{X} \in \mathbb{R}^{[\mathsf{W},\mathsf{S}]}$. This cropping plus warping simulates activities of slightly different durations.

Furthermore, jitter noise are added to the warped sequences. A piece-wise approximation of the input sequence $\mathbf{X} \in \mathbb{R}^{[\mathsf{W}]}$ per sensor is evaluated at time-shifted points. Finally, random noise is added to the sequence. These augmentations take place at training. Algorithm 3 summarises these data augmentations.

5.4 TRANSFER LEARNING FOR M-HAR

Figure 5.4.1 summarises in detail the Attr-based Transfer Learning for M-HAR, given in the introduction of this chapter, Figure 5.0.1. Sequences from different source domains will be used for Transfer Learning for M-HAR. Specifically, a fine-grained class-wise and attribute-wise annotated dataset will be deployed. This dataset contains recordings from OBDs and human joint-pose measurements from a marker-based MoCap. Human poses are represented by the position and angular rotation of human joints when carrying out an activity. In addition, SOBDs are proposed as alternative source datasets, extending the usability of the human-joint pose datasets. These synthetic devices are computed from recordings of human poses or pixel coordinates annotations of video datasets for video-based HAR. These datasets allows creating a synthetic dataset for initialising DL models for M-HAR.

Furthermore, a transferable tCNN is proposed, which handles input sequences from different domains, either different multi-channel time-series Spaces or label spaces, exploiting attribute representations. This tCNN creates a representation of recordings from the human limbs. Following, the attribute-based tCNN is evaluated for Transfer Learn-

4 The window size influences the speed of M-HAR and memory consumption at deployment and is usually a hyperparameter that remains free for tuning when deployment. For example, at MotionMiners Dataset (MM), predictions of 1s are used as a compromise of acceptable classification performance and prediction time.

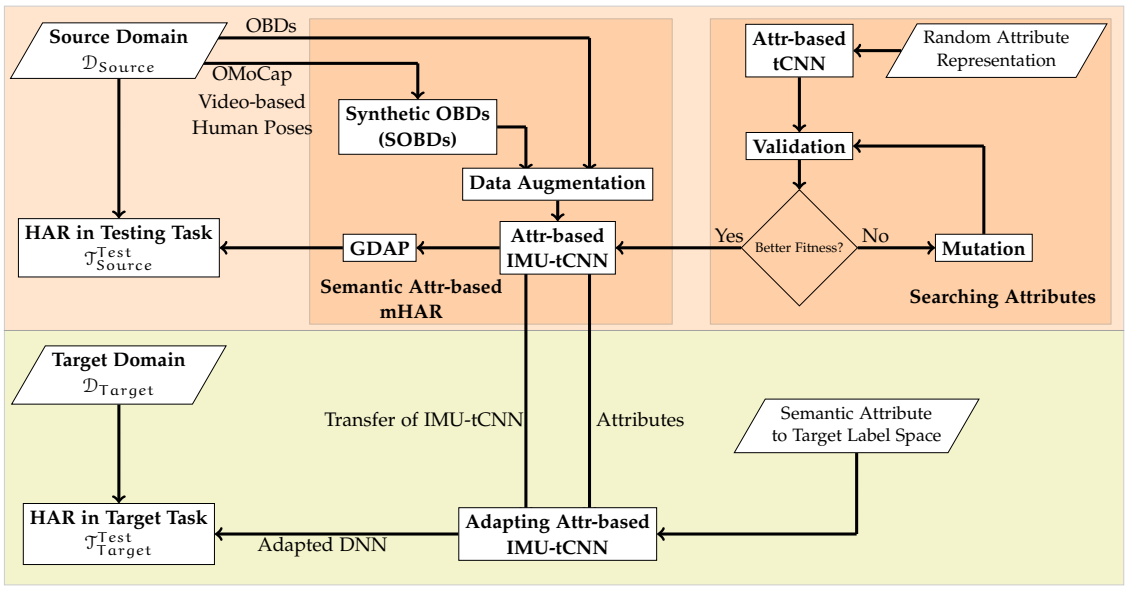


Figure 5.4.1: Attribute-based Transfer Learning for Multi-channel Time-Series HAR (Attr-based Transfer Learning for M-HAR) under different domains, $\mathcal{D}_{Source} \neq \mathcal{D}_{Target}$; it uses an attribute-based transfer and a Attr-IMU-tCNN for parameter transfer. The Attr-IMU-tCNN is adapted for processing human recordings and for full-transferability; An evolutionary algorithm searches for a suitable attribute representation.

ing for M-HAR. Moreover, the impact of enlarging a dataset or transferring learning for M-HAR is evaluated; that is, transferring from a large dataset or from a synthetic dataset using sequences from human poses and inertial measurements to target datasets. The attribute-based tCNN is also evaluated for boosting dataset annotation for M-HAR. Human Activity Retrieval (HARetr) is proposed as part of a semi-automated annotation approach. This approach uses attribute representation of windows for providing an ordered list of window candidates given an activity query.

EVALUATION 6

Attr-based Transfer Learning for M-HAR is a feature-representation and parameter transfer method. It seeks to address different transfer learning scenarios for performing M-HAR. These transfer learning scenarios handle situations where the source and target multi-channel time-series Space, tasks, and domains differ, as Section 2.3 discusses. Attr-based Transfer Learning for M-HAR considers different levels of transferability to solve M-HAR on benchmark target datasets: first, an architecture that handles different dataset configurations; it creates fixed-size representations of OBD recordings that are representative of the human limbs; second, semantic activity representations of activities that are found in target datasets $\mathcal{D}_{\mathtt{Target}}$; and third, data from a variety of source datasets.

The Attr-based Transfer Learning for M-HAR is built upon a composite of functions, namely $\mathbf{r}(\mathbf{h}(\mathbf{g}(\cdot)))$ as proposed in Section 5.1. The evaluation of the Attr-based Transfer Learning for M-HAR is carried out addressing each of the functions: the feature extractor $\mathbf{g}(\cdot)$, the fusion and attribute predictor $\mathbf{h}(\cdot)$, and classification $\mathbf{r}(\cdot)$, following these four steps.

1. Different configurations of deep architectures are evaluated on the source and benchmarking target datasets. This evaluation sets the baseline for this thesis, setting a starting point for comparison. Besides, this evaluation seeks to collect the M-HAR performance for different network configurations explored in the related work and by the author of this thesis. These configurations are discussed in the related work and presented in Subsection 5.1.1. Experiments consider two major network architectures, the *TCNN* and *IMU-TCNN*, and four fusion strategies (FC, LSTM, FCN, TPP), two MLPs using softmax and *Sigmoid*, and three pooling strategies (no pooling, max-pooling and spectral pooling). The composite function $\mathbf{h}(\mathbf{g}(\cdot))$ is given by the *IMU-TCNN* with a TPP layer per parallel branch and an attribute predictor, as Figure 5.1.1 shows. Pooling layers are added after every two convolutional layers, also noted by the number of pooling layers.

2. The Attr-IMU-tCNN is subsequently evaluated on attribute-wise annotated LARa. Besides, M-HAR is carried out on Synthetic On-body Device (SOBD) datasets. These datasets are created following Section 5.3 exploiting human poses of a marker-based

MoCap and pixel coordinates from datasets intended for video-based HAR. The idea is to evaluate the potential of SOBD datasets from a kinematic perspective for improving real-world M-HAR problems.

3. The attribute representation function $\mathbf{r}(\cdot)$, Equation 5.1.8, is derived on target datasets $\mathcal{D}_{\mathtt{Target}}$ using the EA presented in Section 5.2. The Attr-IMU-tCNN from step two trained from an attribute-wise annotated dataset allows for such search, allowing to find interpretable attribute representations in benchmarking datasets. Learnt attributes are then deployed for M-HAR.

4. Attr-based Transfer Learning for M-HAR is experimented upon considering the proposed composite function $\mathbf{r}(\mathbf{h}(\mathbf{g}(\cdot)))$, Adaptable Attr-IMU-tCNN, proposed in Subsection 5.1.1 and the learnt attribute representation. The Attr-based Transfer Learning for M-HAR proposed in this thesis seeks to transfer the attribute representation and the parameters or weights of a deep architecture. The Attr-IMU-tCNN trained on source datasets, including the SOBDs, along with the learnt attributes, are transferred to benchmark datasets. Attr-based Transfer Learning for M-HAR seeks to combine the Attr-IMU-tCNN architecture and the semantic attributes from the source datasets, LARa, to the target datasets. Here, the impact of this transfer learning under cases where the target data is limited is evaluated.

   Architectures are pretrained using source datasets $\mathcal{D}_{source}$. The convolutional layers of the pre-trained networks $TCNN_{\mathcal{D}_{\text{Source}}}$ and $IMU\text{-}tCNN_{\mathcal{D}_{\text{Source}}}$ are transferred to a *TCNN* to be trained on the $\mathcal{D}_{\mathtt{Target}}$, denoted by for example $IMU\text{-}TCNN^{attribute}_{fuse} \uparrow^{\mathcal{D}_{\text{Target}}}_{\mathcal{D}_{\text{Source}}}$. The number of transferred convolutional layers (N_{conv}) and the [%] of $\mathcal{D}_{\mathtt{Target}}$ for fine-tuning is investigated. The non-transferred layers, e.g., FC, are trained from scratch. Depending on the architecture, layers, to a certain extent, are transferred models deployed on the target datasets $\mathcal{D}_{\mathtt{Target}}$, presented in Table 6.1.1. The Attr-IMU-tCNN with the learnt attributes are completely transferable. Here, the LARa becomes important as it provides annotation of attribute representation sample-wise.

The target datasets consist of OBD recordings of limited size, which are common in the literature. Two types of source datasets are used: a large multi-channel Time-Series dataset composed of recordings from marker-based MoCap and OBDs; and the here proposed SOBD. Section 6.1 presents in details the datasets. This chapter presents the datasets, considering their recording, annotations, and properties that made them suitable for evaluating the method presented in this thesis.

A summary of the most relevant evaluations is shown below. Appendix A shows a detailed evaluation of the datasets, presenting the evaluation per dataset, considering

the baseline networks, the fusion strategies, the MLPs, and TL varying the number of transferred layers and training set proportions—a set of hyperparameters that are usual when training deep architectures are investigated.

6.1 DATASETS

Table 6.1.1: Source $\mathcal{D}_{Source}$ and target $\mathcal{D}_{Target}$ datasets for experimentation.

	Type	Datasets					
$\mathcal{D}_{Source}$	OBD	LARa-Mb					
	marker-based MoCap	LARa-M					
	SOBD from pose-based HAR	J-HMDB	CAD-60	NTU RGB+D			
$\mathcal{D}_{Target}$	OBD	LARa-Mb	LARa-MM	Opp	Pamap2	OPD	MM

For the validation of the Attr-based Transfer Learning for M-HAR, five multi-channel Time-Series datasets are selected, namely, the LARa [NRMR+20], Opp [RCR+10, CSC+13b], the Physical Activity Monitoring Data Set (Pamap2) [RS12a, RS12b], the Order-Picking Dataset (OPD) [FMHF16], and the MM dataset. Besides, SOBD are computed from marker-based MoCap dataset, concretely a subset of the LARa dataset, LARa-M; human poses from pixel-based annotations of video-based HAR datasets. The LARa dataset is considered as it contains synchronised recordings from marker-based MoCap, two sets of OBDs, and sample-wise attribute annotations. The Joints Human Motion DataBase (J-HMDB), CAD-60, NTU RGB+D video-based HAR datasets are selected as they contain pixel coordinates annotations from the human joints. The OPD and MM are dataset recordings from real M-HAR scenarios, with no restrictions on the subjects performing the activities. These datasets are very challenging as they are highly unbalanced, making them suitable for validating the method, the TL for M-HAR. The LARa-Mbientlab (LARa-Mb) and LARa-MotionMiners (LARa-MM) will also be used as a target domain, i.e., from human pose to inertial measurements.

These datasets are explained below.

6.1.1 *Multi-Channel Time Series Datasets*

LARa Dataset

The LARa Dataset is chosen as a source of marker-based MoCap from joint poses and OBDs [NRMR+20]. The dataset is made by recreating three warehousing scenarios in a

(a) (b)

Figure 6.1.1: The LARa dataset contains measurements from a marker-based MoCap System and two sets of OBDs. Subjects perform activities in three intralogistics scenatios. Image taken from [NRMR+20].

constrained environment, ensuring natural motion and resemblance to reality [Rei21]. The constrained scenario is the *Innovationlab Hybrid Services in Logistics* at the TU Dortmund University, and the Fraunhofer IML [RVBZH17].

The LARa dataset consists of an extensive collection of synchronized recordings from a marker-based MoCap system, called LARa-M, and two sets of OBDs, called LARa-Mb and LARa-MM respectively [NRMR+20]. It contains recordings of 14 subjects performing activities in the intralogistics. Each subject performs activities in $R = 30$ independent recordings of 120s. Data is recorded in three distinct warehousing scenarios. A Business Process Model (BPM) describes the process of each scenario. This BPM gives the relevant information to the subjects to carry out a process, but it does not script the subject's movement. Furthermore, LARa provides sample-wise annotations of $Y = 8$ activity classes and $M = 19$ semantic attributes. These dense annotations make LARa dataset suitable for evaluating Attr-based Transfer Learning for M-HAR.

The LARa dataset considers activities that are common in intralogistics, as Table 6.1.2 lists. *Synchronization* is employed for synchronizing the joint poses and the inertial measurements, as Figure 2.2.1 shows. Besides, LARa provides attribute annotations of the activities [NRMR+20, pp. 15–17]. Table 7.5.4 lists the attributes labels. Each activity class is represented by a set binary attribute-vectors $a \in \mathbb{B}^{M=19}$, where M is the number of attributes defined in the LARa dataset. Attributes were annotated alongside the activities. As a result, there are different attribute representations per class. LARa provides a set of J tuples of an activity class and attribute vector $\mathbf{A} = \{(y, \mathbf{a})^{(j)}\}_j^J$, with the attribute matrix $\mathbf{a} \in \mathbb{B}^{[M=19]}$ and a corresponding class vector $y \in Y$. Each vector $a_j \in \mathbf{A}^{[J,M]}$ has a corresponding class $y_j \in Y$. This way, the attribute representation $\mathbf{A}$ maps the $J = 304$ attribute vectors to the $Y = 8$ activity classes.

LARa-M consists of recordings from 3D poses of 22 human joints with a recording rate of 200 Hz. A pose is a vector representation with a joint's position and rotation coordinates. These are all centred with respect to the lower back of a subject. LARa-Mb and LARa-MM provide recordings from five and three OBDs respectively. Each OBD records 3D linear acceleration and angular velocity with 100 Hz. As the recordings are carried out in a controlled environment, LARa-Mb does not consider measurements from magnetometers. LARa-M contains recordings from 14 subjects. However, LARa-Mb contains recordings from 8 subjects. In general, LARa provides 714 min of annotated recordings, being a large annotated dataset for M-HAR.

For M-HAR using human poses, each of the 3D poses of the 21 joints from the LARa-M, excluding the reference joint, is considered as a separate channel along each axis. There are in total $S = 126$ channels. This setting differs from the tree-like structure of human poses in [KY18]; see Section 3.2. For the LARa-Mb, there are in total $S = 30$ channels considering the five OBDs with 3D linear and angular accelerations. For LARa-MM, there are $S = 27$ sensor channels with 3D linear and angular accelerations and magnetometer measurements.

Figure 6.1.1 shows an example of LARa in the intralogistics, the marker-based MoCap suit and OBDs on a subject, and the recorded human pose. The dataset is highly unbalanced, where 50% of the recordings contain the *Handling (centred)* activity, given that the warehousing scenarios are related to order-picking activities, as Table 6.1.2 shows.

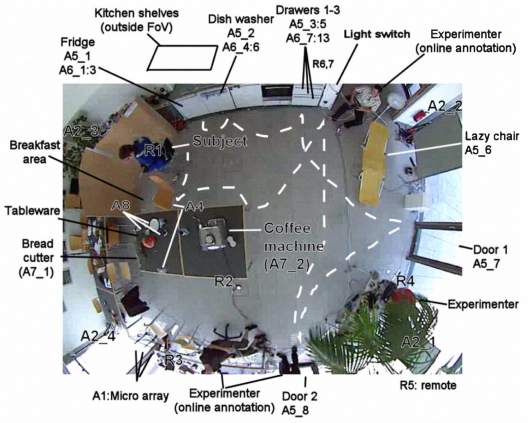
Following [RS12b, CSC+13b], the datasets are split into non-overlapping training, validation and testing sets. For marker-based MoCap, the validation and testing sets contain recordings from subjects [5, 11, 12] and [6, 13, 14], respectively. The training set contains recordings from the other eight subjects of marker-based MoCap. Similarly, for LARa-Mb and LARa-MM, the training set contains recordings from subjects [7, 8, 9, 10], the validation from subjects [11, 12], and testing from subjects [13, 14].

Table 6.1.2: Proportion [%] of activity classes in LARa-M dataset.

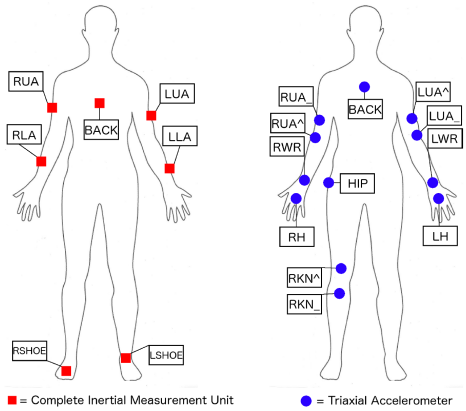
Standing	*Walking*	*Cart*	*Handling (upward)*	*Handling (centered)*	*Handling (downward)*	*Synchronization*	*None*
10.29	10.35	8.26	7.71	50.56	5.44	2.16	5.24

Opportunity Dataset

The Opportunity Challenge Dataset (Opp) dataset [RCR+10] is a large dataset containing recordings of 12 subjects performing ADL in a sensor-rich environment. The dataset consists of recordings from 12 OBDs, providing recordings from 72 sensors of 10 modalities in 15 wireless and wired networked sensor systems in the environment, in objects, and



(a) Recording room of the Opportunity dataset. Image taken from [RCR+10].



(b) Sensor configuration on a subject in the Opportunity Challenge Dataset (Opp). Image shows the five XSense OBDs and the two InertiCube3 in ■, and the 12 Triaxial accelerometers in ■. Image taken from [RCR+10].

Figure 6.1.2: Opportunity Challenge Dataset (Opp) Dataset setup.

on the subjets' body. In total, the dataset contains 25 hours of sensor measurements. The rich environment simulates a studio flat with a kitchen, deckchair, and outdoor access, as Figure 6.1.2a. The authors sought to record ADLs as much as realistic as possible.

A subject performs five times a scripted ADL run without a certain execution pattern. The ADL run consists of processes common in daily living. A process, in this context, means a structured series of activities towards a defined goal, e.g., preparing a sandwich. The subject performs the following processes: *get up*, *coffee*, *sandwich*, *clean*, *break*. Additionally, they carries out one *drill* run. In the *drill* run, the subject repeats 20 times a sequence of 17 activities. There were no restrictions on performing the activities. Table A.4.8 in the appendix presents the activities.

The Opportunity Challenge

The Opp [CSC+13b] is a subset of the Opportunity dataset, publicly available in [CSC+13a]. It comprises recordings of OBDs from four subjects performing ADLs, only addressing the gestures, Opp_{ges}, and locomotion, Opp_{loc}, settings of the Opportunity dataset. In the gestures setting (Opp_{ges}), the goal is to classify $Y = 18$ right-arm gestures, e.g., activities like opening and closing doors or drawers and drinking coffee. In the locomotion setting (Opp_{loc}), $Y = 5$ classes of movements and postures of the body are recognized—

specifically, the tasks are *walking*, *standing*, *sitting*, and *lying*. Both settings contain the *Null* class as the fifth class—this class covers all the human actions that are irrelevant to the task. Table A.4.11 and Table A.4.12 show the activity classes from both tasks.

The Opp includes recordings from: five XSense OBDs (accelerometer, gyroscope, and magnetometers) mounted on a custom-made motion jacket, 12 Bluetooth 3-axis acceleration sensors on the limbs and commercial InertiCube3 inertial sensors located on each foot, as Figure 6.1.2b shows. Each sensor axis is treated separately, thus, yielding a multi-channel Time-Series space of $S = 113$ channels. The authors recorded Opp with a sample rate of 30 Hz.

The sessions, called *ADL3* of *subject2* and *subject3*, are used as the validation set, and the sessions *ADL4* and *ADL5* of *subject2* and *subject3* are used as the testing set, and the rest of the sessions for training, following [CSC+13b], and the works in [RC15, OR16]. A sliding window approach is used for segmenting the sequences, with a window size of 720 ms or $W = 24$ and a step size of 360 ms or $Str = 12$.

Physical Activity Monitoring (Pamap2) Dataset

The Pamap2 dataset [RS12a, RS12b], publicly available in [Rei12], consists of recordings from nine participants carrying out locomotion activities. According to the dataset's protocol, the dataset contains $Y = 12$ action classes, as Table A.4.19 lists. Pamap2 provides recordings from four OBDs, placed on the hand, chest and ankle. Besides, the usual accelerometers, gyroscope, magnetometer, Pamap2's OBDs include temperature sensors, a heart-rate monitor. Overall, the dataset is defined by a multi-channel Time-Series space of $S = 40$ sensors from four OBDs. Pamap2 dataset uses a recording rate of 100 Hz.

Following [HHP16, OR16], recordings from *subject5* and *subject6* are used as validation and testing sets respectively. A sliding window approach with a window size of 3 s or $W = 100$ and step size of 660 ms or $Str = 22$ is used for segmenting the sequences. The window size is smaller than the one in [HHP16] for generating a larger number of segments. The step size allows a 78% overlapping as in [RS12b, HHP16]. In general, the size of the input sequences is $[W, H, C] = [100, 40, 1]$.

Order Picking Dataset

The OPD [FMHF16] consists of recordings from three subjects performing order picking activities in two logistic scenarios, denoted as OPD_A and OPD_B. This is the only dataset for production and logistic that solely uses real-life data. The dataset contains 10 min and 23.30 min long recordings for OPD_A and OPD_B respectively. The subjects wear three OBDs on the wrists and torso. This setup was suggested by logistics experts to ensure the

sensors do not interfere with the actual work. Each OBD captures 3D accelerometer, a gyroscope and magnetometer measurements with a sample rate of 100 Hz. Thus, there are, in total, $S = 27$ sensor channels.

There are seven action classes in the dataset, as Table 6.1.3 shows. The dataset includes two background classes *unknown* and a *sensor flip*. These activities help for synchronization and marking the beginning and end of an order line. Following [FMHF16], the *Null* class is not considered, as this class represents non-annotated material, e.g., faulty measurements. Besides, activities in both warehouses are disjoint, i.e., not all action classes are used in both warehouses. Subjects in OPD_A use a paper list as guidance, annotated as *info* activity. They also acknowledge each picking by manually signing the list. In OPD_B, subjects use a handheld device as guidance, and the *info* action represents the interaction with the handheld device. Instead of the *acknowledging* activity, subjects use the handheld device for *scanning* an article.

Table 6.1.3 presents an overview of the activities and their proportions in the dataset. The OBD is highly unbalanced, having that the *walking* and *picking* are the action classes with the highest number of samples, similar to LARa. A sliding-window approach with a window size of 1 s or $W = 100$ with a step size of 10 ms or $Str = 1$ is used for segmenting the sequences. Thus, all possible windows are extracted from the sequences; as a result, multiple windows represent the same activity for each event in a sequence.

Table 6.1.3: Overview of the Order-Picking Dataset (OPD) and number of frames for each of the activity classes in the different parts of the dataset. [GLMR+17].

Warehouse	Walk.	Search.	Pick.	Scan.	Info.	Carry.	Ack.	Unknown	Flip
OPD_A	21465	344	9776	0	4156	1984	5792	1388	1900
OPD_B	32904	1776	33359	6473	19602	0	0	264	2933

MotionMiners Dataset

The MM[1] consists of recordings from three subjects performing order picking activities in two warehouses. Similarly to OPD, the subjects wear three OBDs on the wrists and torso. This set-up is the current deployed at *MotionMiners GmbH*. Each OBD records 3D accelerometer, gyroscope, and magnetometer measurements with a sampling rate of 100 Hz. Thus, there are $S = 27$ sensor channels.

There are seven action classes in the dataset, concretely *Null*, *Ignore*, *Walk*, *Stand*, *Handle*, *Drive* and *Sit*. The *Ignore* labels refers to recording material that are not considered due to

1 This dataset is provided by *MotionMiners GmbH*, https://www.motionminers.com.

faulty measurements or sensors not placed on the subject, and it is not used at training, following *MotionMiners*. *Handle* refers to handling or picking items. Finally, the *Null* is reserved for activities that are not relevant for a logistic task.

Similar to the Opp, OPD, and following the experimental results provided by *MotionMiners GmbH*, a sliding-window approach with window size of 1 s or $W = 100$ with a step size of 10 ms or $Str = 1$ is used for segmenting the sequences.

The author of this thesis assigned manually an attribute representation from the LARa to the activity classes of MM. They used the semantic definition of the activity classes provided by *MotionMiners GmbH*. This attribute annotation does not utilize video-based annotation, but mainly the semantic description of the activities and the granular annotation from *MotionMiners GmbH*.

6.1.2 *Video-based Datasets*

J-HMDB Dataset

J-HMDB [JGZ+13] contains 928 video clips of different human actions in the wild. The J-HMDB is a subset of the Human Motion DataBase (HMDB) [KJG+11], with human joint annotations. The dataset comprises 21 activities. The person performing the activity in each frame is manually annotated with their 2D joint positions, scale, viewpoint, segmentation, puppet mask and puppet flow (not relevant for this thesis). Every activity class contains 36 to 55 video clips, and each clip contains 15 to 40 frames. The video clips are recorded at a rate of 25 Hz. The duration of activities ranges from 0.5 to 2 seconds. The J-HMDB is a small dataset with a total duration of all video clips of around 20 min.

CAD-60 Dataset

Cornell Activity Dataset (CAD-60) [SKSS14] contains RGB-D video sequences of human activities, which are recorded using the Microsoft Kinect v1 [Zha12]. The dataset contains $[320 \times 240]$ sized RGB-D motion sequences and human joint poses acquired at 30 Hz. The human joint poses are composed of 15 3D-joint positions per subject; here, the z-axis or depth is measured by the Microsoft Kinect v1. The activities are performed by four subjects in five different constrained environments: office, kitchen, bedroom, bathroom, and living room. The subjects carry out 12 activities. The total recorded time for all the activities is approximately 47 min.

NTU RGB+D Dataset

A large Scale RGB+Depth Dataset for 3D HAR from the Nanyang Technological University (NTU RGB+D) [SLNW16] contains 60 activity classes and 56880 video samples. The dataset contains RGB videos, depth map sequences, 3D joint poses, and infrared (IR) videos for each sample. Videos are captured from 40 different human subjects, using the Microsoft Kinect v2 at 30 Hz. Human joint poses consists of 3D coordinates of 25 major body joints for detected and tracked human bodies.

The actions are performed in different constrained environments: 40 ADLs, e.g., drinking, eating, and reading; 9 health-related actions, e.g., sneezing, staggering, and falling down, and 11 mutual actions, e.g., punching, kicking, and hugging.

6.2 EVALUATION METRICS

Following the revision in [RNMR+19], the literature evaluates the M-HAR methods using the accuracy (*Acc*), Precision (*Pr*), Recall (*Rec*), weighted F1 (*w*F1), and mean F1 (mF1). The most used metric is the accuracy.[2] For the overall performance across all activity classes, the accuracy is computed as the number of correctly recognized samples divided by the number of all samples N in the test set. Nevertheless, the *Acc* does not consider the unbalance problem[3] of the M-HAR datasets. F1 metrics could give a more impartial conclusion of the performance for M-HAR [CSG+13, OR16]. F1 metrics consider the correct classification of each class equally. They compute the mean and weighted average of the precision and recall, Equation 6.2.1, and the proportion of class in the dataset.

For each activity class, the precision and recall are,

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}\,, \tag{6.2.1}$$

with True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). These metrics show different aspects of the performance of an HAR system. Pr indicates the percentage of times that a recognition result is correct. Rec means the percentage of times that an activity is detected by the HAR system.

2 Generally, reported performances in the publications show relative good results, approaching 100% accuracy. Nevertheless, these performances must be revised, as suggested in [RNMR+19].

3 Some activity classes contain more samples than other classes.

Being $y \in \mathcal{Y}$ an activity class of a set of activities of a dataset, $n^{(y)}$ the number of samples of the activity class y, and N the total number of samples in the dataset, the mean F1 ($mF1$) is calculated as,

$$mF_1 = \sum_{y}^{Y} 2 \times \frac{Pr^{(y)} \times Rec^{(y)}}{Pr^{(y)} + Rec^{(y)}}, \tag{6.2.2}$$

and the weighted F1 ($wF1$) is calculated as,

$$wF_1 = \sum_{y}^{Y} 2 \times \frac{n^{(y)}}{N} \times \frac{Pr^{(y)} \times Rec^{(y)}}{Pr^{(y)} + Rec^{(y)}}. \tag{6.2.3}$$

The results presented in this work are the mean and SD ($\mu \pm \sigma$) from five runs on the testing and validation sets under the similar training hyperparameters and controlling the source of the randomness of Python and PyTorch with a fixed random seed, similar to [OMR16], as seen in Subsection 3.4.1; this due to time constraints considering the large amount of experiments, including datasets and network hyperparameters.

A permutation test is performed to evaluate the performance changes, stating whether a certain performance is significant or not. The testing accuracy will be considered for this permutation test.[4] The randomization tests are carried out comparing the classification testing accuracies between the proposed method and the baseline and the state of the art [OG09].

The values in bold in all the tables in Appendix A have the corresponding testing accuracy significantly higher than the baseline, based on a permutation test.

6.3 RESULTS

6.3.1 *Baseline for Multi-Channel Time Series Datasets*

To compare with the proposed Adaptable Attribute IMU-Temporal Convolutional Neural Network (Adaptable Attr-IMU-tCNN), a tCNN baseline is evaluated following the architecture designs present in [HHP16, OR16]. The tCNN has temporal convolutions over all OBDs recordings. It is as a simpler IMU-tCNN with a single temporal branch for all of the OBDs sequences. This thesis also compares using three different fusion alternatives,

4 A performance improvement is considered significant if the ρ-value is less than 5.0×10^{-2} and highly significant if the ρ-value is less than 1.0×10^{-2}.

namely, using LSTM layers following [HHP16, OR16], FCN layers following [YLSR18] and FC layers. The architectures with an LSTM layer are similar to the DeepConvLSTM from [HHP16, OR16]. The DeepConvLSTM also contains four temporal convolutional layers of 128 units with ReLU activation functions, but it uses two LSTM layers instead of the FC layers. The LSTMs capture the global temporal dynamics of the input data. Additionally, the architecture using the FCN as the fusion layer before the classifier layer is similar to the one proposed in [YLSR18]. The FCN layers contain $[1, 1, C]$ convolutions connecting the entire input along the deep dimension to the output in a fully-way fashion. This convolution layer is very efficient, keeping temporal relations. Architectures containing $[2, 1]$ max-pooling layers with a stride of 1 are also considered; see Section 3.1 for details of these architectures.

For all architectures, each of the convolutional layers has $C = 64$ filters of size $[5, 1]$ performing convolutions only on the time axis. Both the FC and the LSTMs layers contain 128 units. Max pooling with a filter size of $[2, 1]$ with a stride of 1 is used.

A nomenclature will denote the networks following *Network-Pooling*$^{actlayer}_{fusion}$, e.g., the here proposed limb oriented tCNN with a TPP fusion layer is denoted as, *IMU-TCNN*$^{sigmoid}_{TPP}$, or the DeepConvLSTM proposed in [HHP16, OR16] is denoted as *TCNN-[1-2]MaxPool*$^{softmax}_{LSTM}$, and the the tCNN with FCN in [YLSR18] is denoted as *TCNN-[1-2]MaxPool*$^{softmax}_{FCN}$.

Figure 6.3.1 shows a summary of the evaluation presented in Appendix: Section A.4 for all the datasets. It presents the testing performance in terms of wF1$[\%]$[5] using the *IMU-TCNN*$^{softmax}_{fuse}$. The *IMU-TCNN* is robust performance compared to *TCNN*, as Table 6.3.1 shows for the LARa datasets.

The outcomes result from a detailed tuning of different training hyperparameters commonly evaluated using deep architectures. Here, the performance of the architectures with respect to the learning rate is relevant. Channel-wise normalisation of the recordings, bringing the sequences of a range of $[0, 1]$ allows not only the temporal architectures to learn, as early layers do not saturate in dead regions following [LBOM12, IS15], and filters are shared among sensors with different amplitudes or scales; but also to deploy similar training hyperparameters, and thus for a fair comparison under similar training and experimental scenarios. Figure 6.3.1 shows the performance of the architectures with respect to different learning rates. The $lr = 10^{-3}$ suits all the datasets.

In general, FCN as fuser layer deteriorates performance. This outcome is constant for all datasets. This result is because the FCN considers dense connections to the deep dimension of the feature map, and it depends on the receptive field of the filters with

5 The acc and mF1 show similar outcomes, so they are not shown.

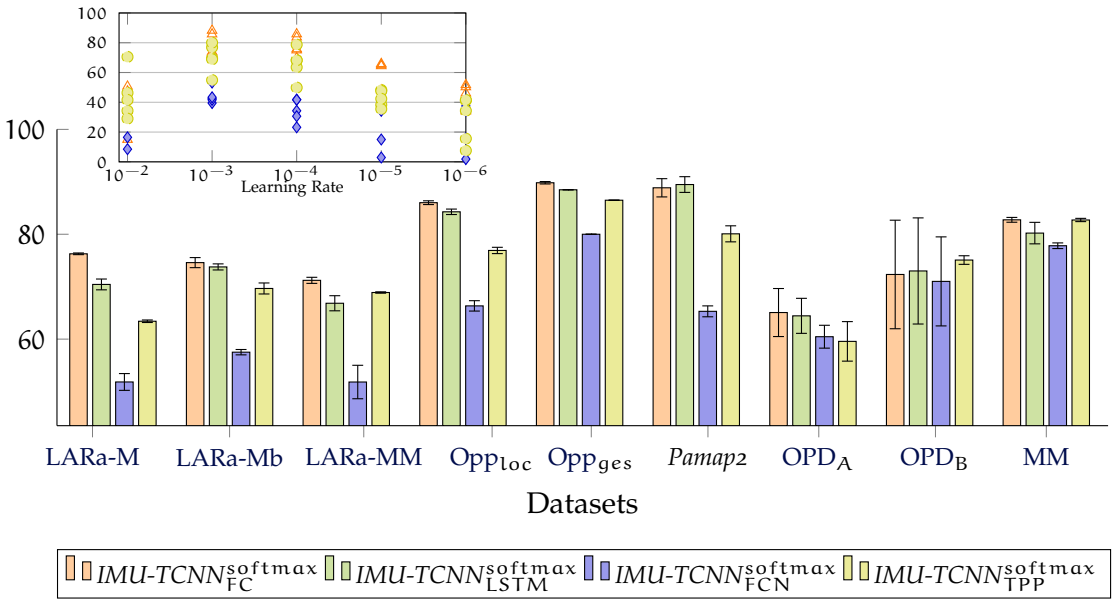


Figure 6.3.1: Best testing performances in terms of wF1 [%] on the OBD datasets for different configurations of the *IMU-TCNN*.

respect to the input. Thus, the architecture with an FCN has a partial view of the input, just performing a final aggregation of the entire sample predictions before the classifier.

The LSTM does not improve M-HAR performance for a dataset with no restrictions or scripts posed to the subjects when performing an activity, e.g., the LARa dataset; that is, all subjects carry out the activities freely following a specific goal given by the BPM, but not a script. This outcome is different for the laboratory setups, Opp and Pamap2, where the subjects repetitively follow a script. This is because the architectures with an FC layer as an aggregator are over-parametrised compared to LSTM and FCN layers. This over-parametrisation is damaging when dealing with a low quantity of data, e.g., Pamap2, but shows advantages for a large dataset, e.g., LARa-M dataset. This outcome can also be seen comparing the performance on the LARa-M with respect to LARa-Mb and LARa-MM, which contain a lesser amount of channels and subjects. This layer increases the descriptiveness when data contains significant variation.

One of the reasons networks with LSTM and FCN layers were used is their relatively lesser amount of parameters compared to FC layers, providing similar performance, as shown in [JK19]. The parameter reduction does not limit LSTM networks' usage for long-temporal abstract relations from previous predictions, which are handy for

structured activity sequences, where possible actions can strongly depend on previous actions when actions have precondition-effect relations [HHP16, OR16]. However, this is different for short and not repetitive temporal relations directly from sensor measurements, where the FC shows an advantage. Besides, they provide sample-wise predictions—in this case, computation becomes very demanding. Un-segmenting the predictions considering the window size and step and computing the mode of the overlapping windows predictions for all the samples from the networks with FC and TPP is enough to boost performance. Un-segmenting avoids the memory-demanding networks using FCN and LSTM at deploying, as they can carry out sample-wise predictions rather than a single prediction per window. These non-dense predictions are also advantageous when considering attribute representations. For example, on the LARa-Mb, the *IMU-TCNN*$_{\mathrm{LSTM}}$ shows a *w*F1 of 73.77 ± 0.58[%] for window-based predictions, and *w*F1 of 74.64 ± 0.54[%] for sample-wise predictions; the *IMU-TCNN*$_{\mathrm{FC}}$ a *w*F1 of 74.59 ± 0.96[%] for window-based predictions and *w*F1 of 75.55 ± 1.03[%] sample-wise predictions after un-segmenting.

The TPP initially shows a negative influence on performance. Considering that TPP is a pooling method, relevant features from not necessarily the strongest movements in the non-active limb might get lost. However, The TPP pools relevant features per limb when considering the *IMU-TCNN* with late FC fusion. This advantage is particularly clear when using a TPP layer. The limb-oriented structure of the *IMU-TCNN* shows an advantage with respect to the *TCNN*. Besides, it shows a more stable performance when repeating the experiments. This outcome is pronounced for the OPD$_{\mathrm{B}}$ dataset.

In general, for the performances on the Opp$_{\mathrm{loc}}$, Opp$_{\mathrm{ges}}$ and Pamap2 datasets and contrary to [RSA15], CNNs benefit from max-pooling operations, even when sequences are short. The Spectral Pooling reduce the number of parameters by $\approx 56\%$ and $\approx 80\%$ respectively, without sacrificing performance. The Spectral Polling improves performance on the LARa-Mb, Pamap2, and MM. The spectral pooling does not negatively affect the networks' performance on the LARa-MM.

IMU-TCNN$_{\mathrm{TPP}}$ will be very relevant for finding semantic attributes on the target datasets and for allowing to a certain extent TL for M-HAR.

6.3.2 *Synthetic On-Body Devices (SOBDs)*

SOBDs from LARa

Following Section 5.3, M-HAR is addressed considering a SOBD located on a given joint. The total acceleration considering the linear and angular changes of the joint poses along time is computed. These SOBDs will be used as source domains for parameter-based

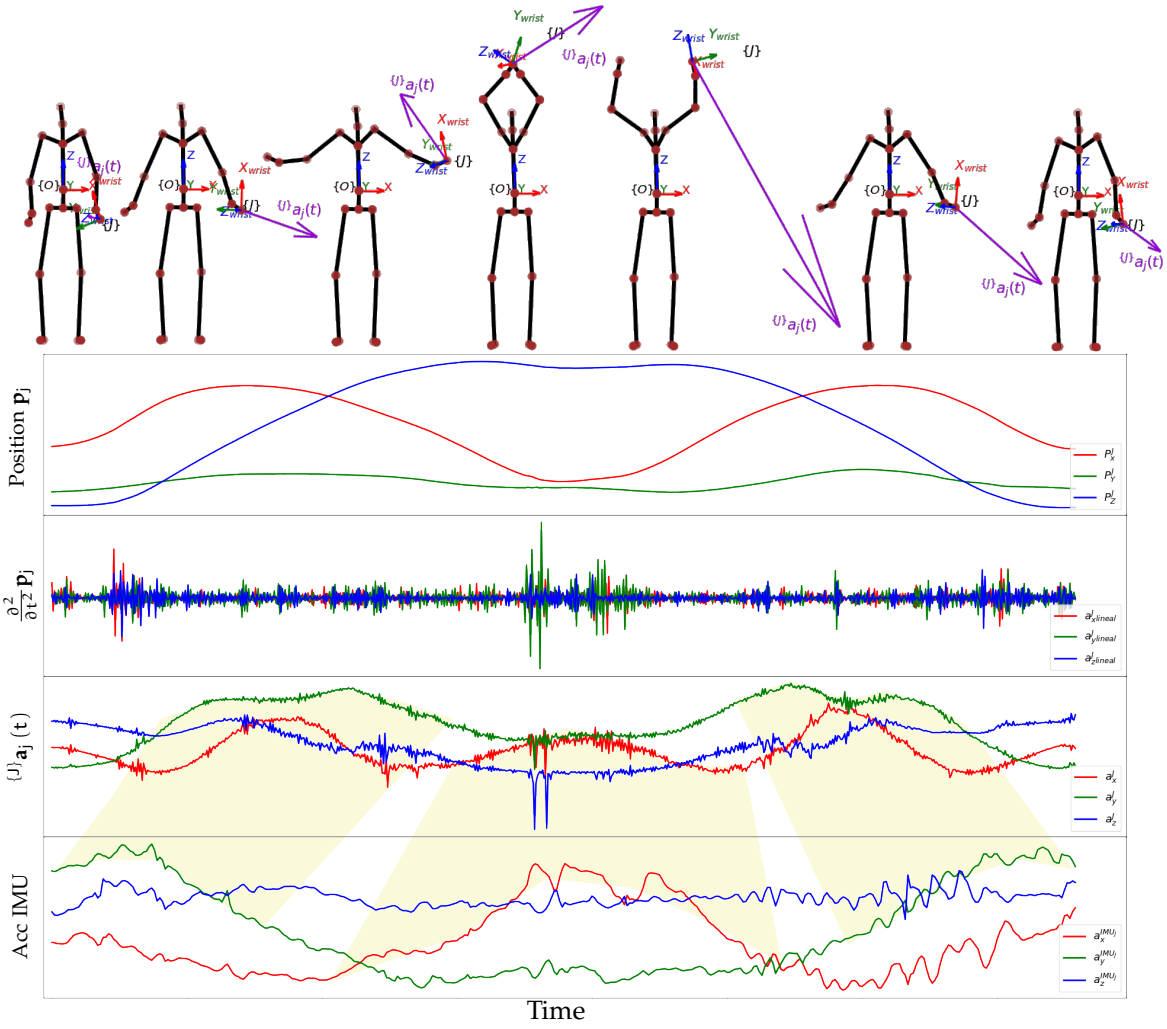


Figure 6.3.2: The figure presents an example of the LARa-SOBD for the *Synchronisation* activity class. The linear acceleration vector of the left arm is shown in ■ on the skeleton. It shows the 3D linear acceleration for [X] axis in ■, [Y] axis in ■, and [Z] axis in ■, from the second order derivative, and the LARa-SOBD following the method, of the left arm. Besides, it shows linear acceleration measured by OBD located on the right arm from the LARa-Mb set.

Transfer Learning for HAR. The LARa-M is deployed for creating the subset LARa-Synthetic OBD (LARa-SOBD). Figure 6.3.2 presents an example of the LARa-SOBD for the *Synchronisation* activity class.[6] It shows the measurements per axis [X, Y, Z] from the left

6 The subject keeps a standing position while moving the arms upwards and downwards synchronously.

wrist pose, its second-order derivatives along time, $\frac{\partial^2}{\partial t^2}\mathbf{p}_j$, and the corresponding linear acceleration of the LARa-SOBD considering a kinematics perspective, ${}^{\{J\}}\mathbf{a}_j(t)$. Besides, it shows measurements of each axis $[X, Y, Z]$ from an LARa-Mb located on the left wrist as a comparison. The LARa-SOBD is sampled at 200 Hz while the LARa-Mb at 100 Hz. The similarities of the SOBDs to the OBD are noticeable. For example, the left arm SOBD presents the highest acceleration magnitudes when the arm velocity changes its direction from upwards to forward or from the torso to the sides and back to the torso.

Table 6.3.1: The testing *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the LARa-Ms, LARa-Mbs, and LARa-SOBDs. The mean and SD from the *w*F1 [%] are given as training method is repeated five times. Values in **bold** are significant with respect to the architectures.

Dataset	Baseline		LARa-Mb$_{100}$	LARa-Mb$_{30}$	LARa-SOBD$_{100}$	LARa-SOBD$_{30}$
	LARa-M$_{100}$	LARa-M$_{30}$				
TCNN	75.80 ± 0.15	75.41 ± 0.35	73.80 ± 0.4	74.59 ± 0.88	$\frac{\partial^2}{\partial t^2}\mathbf{p}_j = 55.46 \pm 0.6$ ${}^{\{J\}}\mathbf{a}_j(t) = 63.87 \pm 0.51$	$\frac{\partial^2}{\partial t^2}\mathbf{p}_j = 56.16 \pm 0.3$ ${}^{\{J\}}\mathbf{a}_j(t) = \mathbf{62.26 \pm 0.14}$
IMU-TCNN	$\mathbf{76.27 \pm 0.16}$	$\mathbf{76.16 \pm 0.14}$	$\mathbf{74.59 \pm 0.96}$	74.09 ± 0.30	$\frac{\partial^2}{\partial t^2}\mathbf{p}_j = \mathbf{60.06 \pm 0.3}$ ${}^{\{J\}}\mathbf{a}_j(t) = \mathbf{64.42 \pm 0.13}$	$\frac{\partial^2}{\partial t^2}\mathbf{p}_j = 56.10 \pm 0.4$ ${}^{\{J\}}\mathbf{a}_j(t) = 56.20 \pm 0.65$

Table 6.3.1 compares the classification performance, in terms of the testing *w*F1 [%], on the testing set from the two LARa-SOBD sets, the second-order derivatives from the LARa-M, $\frac{\partial^2}{\partial t^2}\mathbf{p}_j$, and the acceleration considering a kinematics perspective ${}^{\{J\}}\mathbf{a}_j(t)$. Since the target domains in Subsection 6.1.1 have different recording rates, two subsets are created from LARa-SOBD. These subsets are sub-sampled from 200Hz to 100Hz and 30Hz. The subset will be denoted with the sampling rate. Table A.4.5 presents performance with regards to the architectures with pooling and a TPP layer. The performances using the LARa-M and LARa-Mb sets are higher with respect to the LARa-SOBD; this is a result of the loss of information with finite derivations. However, the SOBD, including components from linear and angular acceleration, shows a better performance than the second-order derivative suggested by [HKA+18, KY18, MRF21]. Including kinematics preprocessing for SOBDs improves classification. This outcome is explained as the acceleration by OBDs is measured with respect to a point located on themselves and not a global frame, capturing a resultant acceleration at the point. This resultant acceleration is due to all the dynamics of the human body. The LARa-M captures a fraction of all these dynamics on the body; thus, SOBD from the LARa-M are limited. Human poses given in 3D coordinates with respect to a global frame, or even with respect to a frame attached to the human body, have to be processed considering rotation and linear rotation of as many as possible points on the human body. Even though SOBDs are approximations of OBD, they become a source for TL, as will be shown.

SOBDs from Video-based Datasets

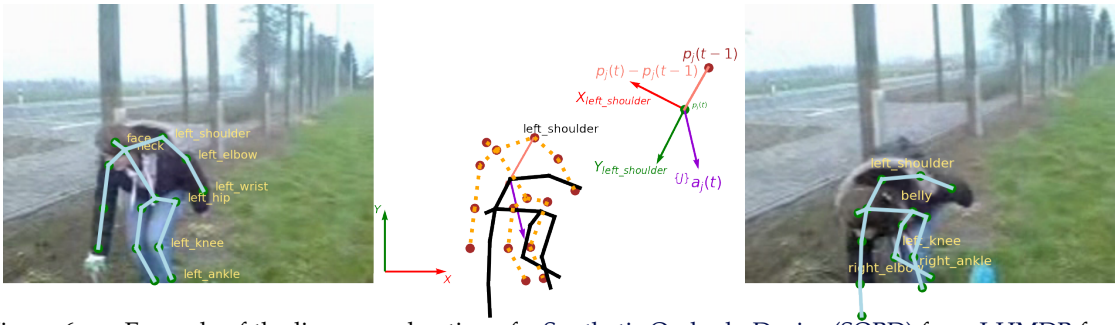


Figure 6.3.3: Example of the linear acceleration of a Synthetic On-body Device (SOBD) from J-HMDB from two consequetive poses of the joint $\{J\}$ located on left wrist of the J-HMDB dataset. A frame attached to joint $\{J\}$ is represented by the unitarian vectors towards (X, Y), specified by ■, ■. The linear acceleration ${}^{O}_{\{J\}}\mathbf{a}_j\,(t)$ on origin frame is presented in ■. The vector from pose $\mathbf{p}_{\{J\}}\,(t-1)$ to pose $\mathbf{p}_{\{J\}}\,(t)$ is given in ■. The linear acceleration of the joint $\{J\}$ given in origin frame O and frame $\{J\}$, including the gravity ${}^{\{J\}}\mathbf{a}_j\,(t)$ is shown in ■. The gravity vector $\mathbf{g}$ is considered vertical and with direction downwards.

Transfer Learning for HAR will consider transfer learning from ground-truth pixel annotations of joint poses from video-based HAR datasets to real OBD data. Similarly to the LARa-M and their SOBD, the sequences of joint-pose annotations in pixel coordinates are considered as multi-channel Time-Series data for M-HAR, and not tree-like tensors, usually seen in the literature; see Section 3.2. Besides, SOBD are computed from these multi-channel Time-Series, following Section 5.3, and deployed as a source for M-HAR. Considering the joint-pose annotations as M-HAR defers from the video-based HAR approaches intended for $\mathcal{D}_{\mathtt{Source}}$. This consideration takes the advantage that the *IMU-TCNN* and *TCNN* process sequences per channel with late fusion and local temporal-neighbourhoods of sequences are likely correlated.

The J-HMDB, CAD-60, and NTU RGB+D will be used as $\mathcal{D}_{\mathtt{Source}}$. Figure 6.3.3 shows an example of these multi-channel Time-Series data from the J-HMDB. J-HMDB provides only 2D pixel coordinates of the human joints, as Figure 6.3.3 shows. These coordinates are normalised with respect to the subjects' size. For the SOBD, the orientation of each joint will be fixed with respect to the body and will be given by the angle between neighbouring joints, e.g., vector in colour ■ in Figure 6.3.3. The CAD-60 and NTU RGB+D provide 3D joint coordinates. The orientation of a joint is fixed in the direction of the vector from two neighbouring joints. Figure 6.3.4 presents the $[X, Y]$ sequence of the SOBD located on the left arm of the subject while performing a *Handling Downwards* activity. It shows the measurements per axis $[X, Y]$ from the left wrist pose, its second-order derivatives along time, $\frac{\partial^2}{\partial t^2}\mathbf{p}_j$, and the corresponding linear acceleration of the LARa-SOBD considering

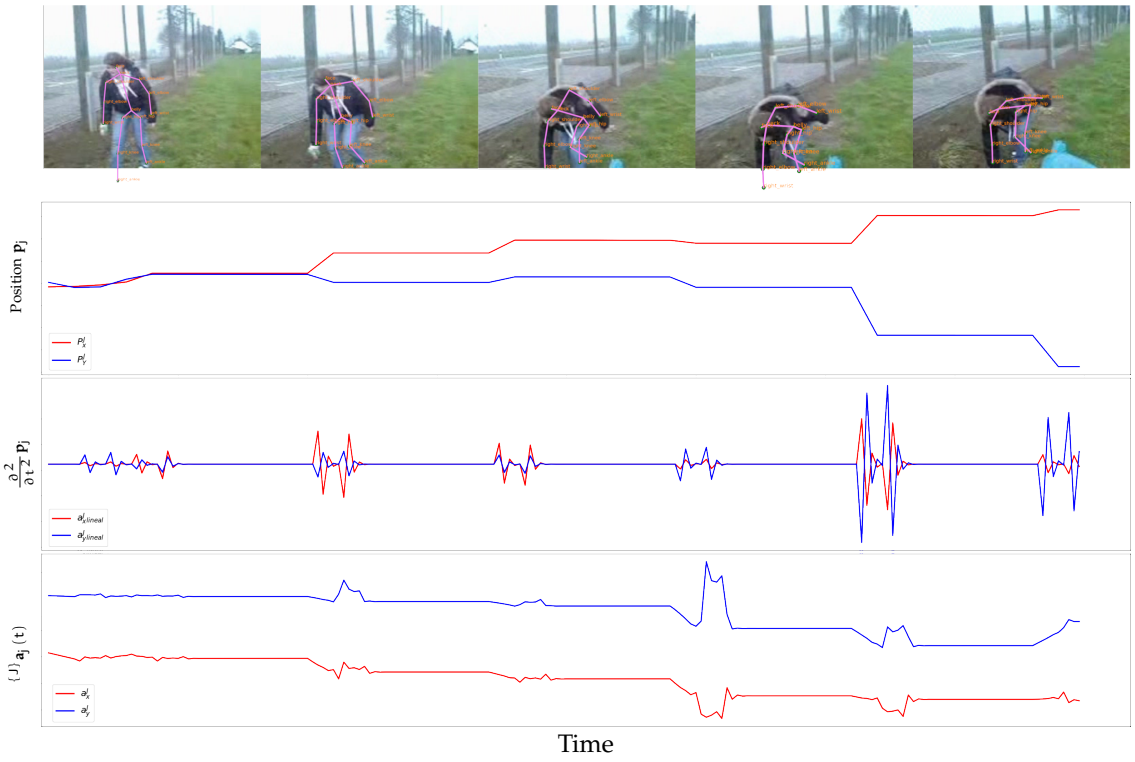


Figure 6.3.4: Figure presents the an example of the SOBD for the J-HMDB dataset, *Pick* activity class. The linear acceleration vector of the left arm is shown in ■. It shows the 2D linear acceleration for [X] axis in ■, [Y] axis in ■, from the second order derivative, and the LARa-SOBD following the method, of the left arm.

a kinematics perspective, ${}^{\{J\}}\mathbf{a}_j(t)$. Compared to previous work [AMRF22], following [HKA+18, KY18], the SOBD retains previous changes of the acceleration that remain while performing the activity along a fix position of the SOBD on the human joint.

Table 6.3.2 shows the classification performance in terms of wF1[%] on the testing sets from the J-HMDB, CAD-60, and NTU RGB+D and their SOBDs. The wF1[%] on the J-HMDB$_{SOBD}$ and CAD-60$_{SOBD}$, the SOBD decreases when compared to pose data. These results are due to the approximations involved in the derivations when creating synthetic data. The M-HAR performance increases when up-sampling for J-HMDB$_{SOBD}$ from 30 Hz to 100 Hz. M-HAR is approached using joint poses differently from the approaches in [CWRS18, KY18, JMA22] focusing on video-based HAR, using RGB frames, Optical Flow and Pose; see Section 3.2 for details. Interestingly, for J-HMDB$_{synth.}$ upsampled to 100 Hz, the mean classification accuracy is of 90.53%, which is significantly higher than

the one reported by [CWRS18] (85.5%), [KY18] (83.1%), and [JMA22] (77.54%). These classification performances are denoted only for giving a sort of proportion about the activity classification. However, these works consider video-based HAR. The predictions were unsegmented, and a majority voting was deployed before computing the wF1[%] so that a comparison would be fair.

Table 6.3.2: The wF1[%] for M-HAR on the three $\mathcal{D}_{\text{Source}}$: J-HMDB, CAD-60, and NTU RGB+D. The predictions on test set were unsegmented before computing the wF1[%]. Multiple experiments were performed varying the stride Str for each $\mathcal{D}_{\text{Source}}$. J-HMDB is not up-sampled for $\text{J-HMDB}_{\text{synth30}}$.

Dataset	Poses	SOBD [30Hz]	SOBD [100Hz]
J-HMDB [25Hz]	50.90 ± 0.05	26.58 ± 0.06	85.68 ± 0.81
CAD-60 [30Hz]	75.75 ± 0.02	50.11 ± 0.06	57.16 ± 0.06
NTU RGB+D [30z]	30.07 ± 0.15	6.70 ± 1.35	36.59 ± 0.49

The SOBDs from the poses recorded by a marker-based MoCap and given by the pixel coordinates from videos will be considered for Transfer Learning for HAR.

6.3.3 *Attribute-based M-HAR*

The Algorithm 2 in Section 5.2 finds the function $\mathbf{r}(\cdot)$, Equation 5.1.8, which is the best suitable binary attribute representation for a given dataset. The $\mathbf{r}(\cdot)$ is represented by a binary attribute representation matrix $\mathbf{A}_{best} \in \mathbb{B}^{[Y,M]}$ with Y the number of activity classes and M the number of attributes, called *Activity-Attribute Matrix* in Section 4.4. Two EA approaches following Algorithm 2 find an attribute representation $\mathbf{r}(\mathbf{a}) = y$ or A_{best} suitable for each dataset by: EA-$TCNN_{FC}^{attribute}$ iteratively training and validating a tCNN using attribute configurations, which mutate starting from random; EA-Attr-IMU-tCNN iteratively validating the Attr-IMU-tCNN trained on the LARa and mutating the attribute representation.

A Nearest Neighbour Approximation (NNA) was used for predicting the class y by measuring the Binary Cross-Entropy (BCE) distance from the attribute vector prediction from the $TCNN_{FC}^{attribute}$ and Attr-IMU-tCNN predictions $\mathbf{g}(\mathbf{X})$ to the set $\mathbf{a} \in A_{n-gen}^{parent1}$, following Subsection 5.1.2. The parent 1 is used as the representation for each generation. The validation wF1 [%] is used as the fitness metric.

For the EA-$TCNN_{FC}^{attribute}$, different numbers M of attributes $\in \mathbb{B}^{[M]}$ for representing a class y are evaluated depending on the number of total classes Y per dataset. Based on training and validating the *TCNN* on random attributes for different $M = [16, 32, 64, 128]$, a set $M = [10, 32, 24, 19, 19]$ number of attributes is finally set for the Opp_{loc}, Opp_{ges}, and

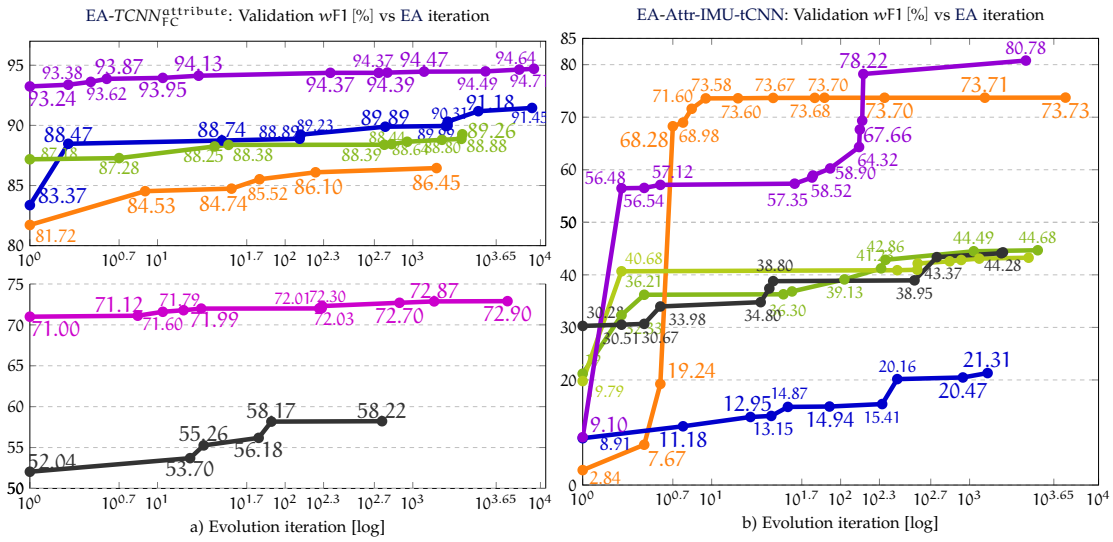


Figure 6.3.5: The validation *w*F1 [%] of the two EA algorithms, EA-$TCNN_{FC}^{attribute}$ and EA-Attr-IMU-tCNN, on the OBD datasets: LARa-Mb in ■, Opp_{loc} in ■, Opp_{ges} in ■, Pamap2 in ■, OPD_A in ■, OPD_B in ■, and MM in ■.

Pamap2, OPD, and MM datasets. An attribute representation is also found for the LARa, using the LARa-Mb set. The $TCNN_{FC}^{attribute}$ is trained from scratch for a fixed number of epochs in each EA iteration.

Figure 6.3.5 shows the validation *w*F1 [%] evolution vs. the EA iterations for the two EA approaches on the OBD datasets.

The EA-$TCNN_{FC}^{attribute}$ adapts the network parameters according to the attribute representation generation. The $TCNN_{FC}^{attribute}$ is able to perform classification starting from a random attribute representation. Here, the initial attribute representation generations $A_{1-gen}^{parent1}$ and $A_{1-gen}^{parent2}$ already restrict the representation, such that the activities do not share the same representation among themselves.

The *w*F1 [%] on the Pamap2 presents the biggest improvements of 8.08% between the first and the last attribute generation at $niter = 8592$ iterations. The *w*F1 [%] on the Opp_{ges} and Opp_{loc} shows *w*F1 [%] improvements of 2.08 [%] and 4.73 [%] respectively. It is noticeable that mostly the performance of the networks presents a relatively good performance despite the randomness of the initial attribute generations. This outcome could be explained as the Opp_{ges} and Opp_{loc} datasets are strongly unbalanced towards the NULL class, and they predict sequence segments correctly with that label. However, as the attributes evolve, the correct predictions increase, showing the potential of the attribute search. Moreover, common attributes among classes are found, which helps

share strength between the more and the less frequent classes. In addition, there is a substantial dimensionality reduction.

Table 6.3.3 shows the learned attribute representation for EA-$TCNN_{FC}^{attribute}$ on the Opp_{loc} dataset after 2443 iterations. There is a sort of semantic relation for the attribute shared among action classes. For example, the *Stand* class shares 4 and 5 attributes with classes *Sit* and *Lie*. These three action classes share less with the *Walk* class with 3 and 2 shared attributes. One advantage of this attribute sharing is that the *Null* class attributes become a source for learning attributes shared with infrequent classes. There is no specific definition of each attribute, but the sort of basic movements or states in each action class keeps a certain relation. One uses these relations to learn a better representation of the actions. The EA-$TCNN_{FC}^{attribute}$ also controls the number of attributes. For example, the attributes a^4 and a^9 can be removed from the representation.

The learned attribute representation for Opp_{ges} dataset shows multiple relations. The activities' attributes are mixed because the dataset contains mainly the opening and closing of doors, fridges, dishwashers, and drawers. These labels also partly explain the relatively good performance of the initial random representation on this dataset. However, after the attribute evolution, classes involving either merely closing or exclusively opening movements display a strong attribute sharing with 20 to 22 common attributes.

The learned attribute representation presents different relations for the Pamap2 dataset. This dataset has different activities, including moving and static ones, but without the *Null* class. The classes are more diverse, having more distinctive action classes. So, the shared attribute is also more diverse, e.g., it keeps relations among classes *Rope Jumping*, *Lying*, *Cycling*, and *Ironing*.

Table 6.3.3: Attribute representation $A \in \mathbb{B}^{10}$ of the Opp_{loc} dataset found using the EA-$TCNN_{FC}^{attribute}$.

Activity	**Attributes**									
	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
Null	1	0	0	0	0	1	0	0	1	1
Stand	0	1	1	0	1	1	1	1	1	1
Walk	0	1	0	0	0	0	0	0	1	1
Sit	0	0	0	0	1	1	0	0	1	1
Lie	1	1	1	0	1	0	0	1	1	0

The attribute representation size remains fixed for the EA-Attr-IMU-tCNN. This EA uses the $IMU\text{-}TCNN_{TPP}$, evaluated in Subsection 6.3.1 using a softmax classifier, trained on the LARa, and considering the LARa attribute representation. The $IMU\text{-}TCNN_{TPP}^{attribute}$ is transferable to different target datasets. It computes fixed size representations per limb with late fusion independently of the number of OBDs. The EA-Attr-IMU-tCNN

finds the mapping $\mathbf{r}(\cdot)$, such as the *IMU-TCNN*$^{attribute}_{TPP}$ trained on LARa becomes fully transferable. In addition, the EA finds semantically interpretable representations based on the LARa annotations.

On the one hand, the attributes found with the EA-*TCNN*$^{attribute}_{FC}$ adapt well to the datasets, where the classification performance is rather high; on the other hand, they are not interpretable. The EA-Attr-IMU-tCNN finds semantic attributes using the representation from the LARa. Table 6.3.4 shows the attributes found for the Opp_{loc} dataset. Activities not belonging to the LARa dataset obtain attribute representations that are somehow semantically different from what is expected. However, the EA finds the attribute *Error* that was intended for activities that are not considered in the annotation process of the LARa dataset, e.g., *Sit* and *Lie* activities. The Attr-IMU-tCNN does not contain filters that activate the non-expected activities; however, it is able to provide an attribute that classifies the activities as strange. The activity *Stand* obtains the attributes *Standing Still* and *Step* that were conceived by the LARa authors when the subject is not changing its position considerably. Similarities can be found for the other target datasets. Activities considering *Walking*, *Running*, and *Cycling* can be represented with attributes *Gait Cycle* and *Step* from the LARa dataset. Activities, where the subject keeps a bending torso pose, are described with the *Upwards* or *Downwards*.[7] The EA also controls the size M of the representation, as attributes get discarded.

Worth mentioning, the large computing time difference between the EA-*TCNN*$^{attribute}_{FC}$ and EA-Attr-IMU-tCNN. While the EA-*TCNN*$^{attribute}_{FC}$ has to train for a fixed number of epochs the *TCNN* for computing the fitness, the EA-Attr-IMU-tCNN only computes the validation performance as the fitness without any training, directly using the *IMU-TCNN*$^{attribute}_{TPP}$ $\uparrow_{LARa}$. For example, the EA-*TCNN*$^{attribute}_{FC}$ on Opp_{loc} takes ≈ 90 days for carrying out an evolution with $niter = 10000$ iterations or generations, where each generation contains 10 offsprings, contrastively, the EA-Attr-IMU-tCNN takes 3 days.

6.3.4 *Transfer Learning for Attribute-based M-HAR*

Attr-based Transfer Learning for M-HAR seeks to combine the Attr-IMU-tCNN architecture and the semantic attributes to transfer learning from an attribute-based source dataset, LARa, to different target benchmark datasets. Attr-based Transfer Learning for M-HAR handles challenging TL scenarios, where the source and target datasets multi-channel time-series Space and tasks differ; so, the number of sensors, devices and activities are different. However, these datasets contain overlapping activity classes with LARa,

7 Attribute representations of the other target datasets can be found in Appendix A.

Table 6.3.4: Attribute representation $A \in \mathbb{B}^{19}$ of the Opp_{loc} dataset found using the EA-Attr-IMU-tCNN$_{LARa-M}$.

Activity	Attributes																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
Null	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Stand	0	1	1	0	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0
Walk	1	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0	0	1	0
Sit	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1
Lie	1	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1	1	1	1

i.e., *Standing*, *Walking*, and *Handling* tasks. Here, the impact of this transfer learning is evaluated under cases where the target data is limited.

Figure 6.3.6 summarizes the testing performance of the Attr-based Transfer Learning for M-HAR on the target datasets and compares it against their baseline when considering different proportions of the training set. Attr-based Transfer Learning for M-HAR allows transferring not only the parameters of a network trained on the LARa but also its representation. Besides, it allows performing HAR when no fine-tuning of the architecture is required with material from the target datasets—here, considering the validation set for finding the attributes—, but even when manually assigning the attributes, in such cases with [0%] of the training material. Attr-based Transfer Learning for M-HAR predicts semantic attributes related to the activities to be computed. Predictions are enough to further post-process the predictions with more global temporal methods, as, for example, the [LMRA+21], or to improve considerably the annotation process, as Chapter 7 will show; this considering that the Attr-IMU-tCNN$_{LARa}$ can be used out of the shelf.

In general, architectures' performances improved for all five target datasets, most prominent when deploying a proportion of the data for training—simulating cases under the assumption that only a fraction of the annotated data is available for training. These findings suggest that the here presented method of Attr-based Transfer Learning for M-HAR is able to learn features and high-level descriptors that are somewhat general as they learn local temporal relations of short-lasting human movements, which are independent of the number of OBDs and their type. Besides, the learnt filters and descriptors are activity-class dependent. As a result, the classification performance

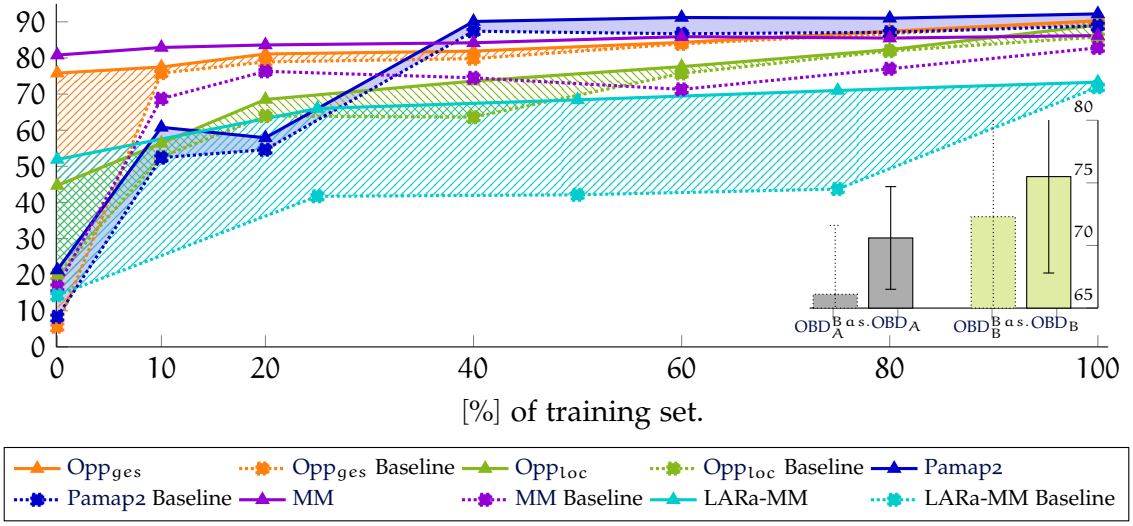


Figure 6.3.6: Testing *w*F1[%] of the Attr-based Transfer Learning for M-HAR on the Pamap2, and Opp$_{loc}$, Opp$_{ges}$, OPD and MM datasets under different proportions of training sets. Dotted lines represent the performance baseline.

improves on the activities that are shared among the datasets. Moreover, this outcome is valid for predictions from relatively short segments of movements.

Furthermore, Attr-based Transfer Learning for M-HAR with LARa as source significantly impacts the performance on the MM, which are conceived for intralogistics purposes. The LARa also was designed with the perspective of transferability to real scenarios. Subjects carry out activities using materials and tools that are found in real scenarios. The impact of a well-designed dataset for TL from the application's perspective is discussed deeply in [Rei21].

Figure 6.3.7 summarizes the findings when performing parameter transfer learning using pose annotations from video datasets as an input stream for improving M-HAR. Three different datasets comprising ground truth pose estimation from videos are deployed as the $\mathcal{D}_{Source}$. The learned temporal convolutional layers of the *IMU-TCNN*$_{fc}^{softmax}$ are used to initialize architectures on three benchmark datasets for M-HAR. The *IMU-TCNN*$_{fc}^{attributes}$ were not providing good performance on the $\mathcal{D}_{Source}$. These datasets' sampling rates are 25 and 30 Hz, and they do not contain enough information for predicting short-duration movements, which are represented by attributes. Besides, the sequences of pixel coordinates of human joints are considered as simplifications of pose estimations.

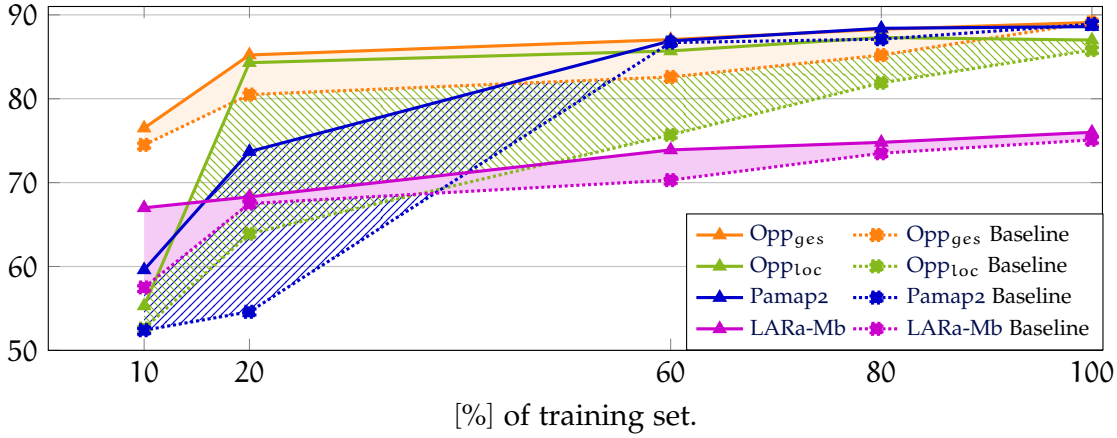


Figure 6.3.7: Summary of the testing *w*F1[%] of the parameter-based Transfer Learning for HAR on the Pamap2, and Opp$_{loc}$, Opp$_{ges}$, OPD and MM datasets under different proportions of training sets using the *IMU-TCNN*$^{softmax}_{fc}$ and the pixel-based video-based HAR datasets. Dotted lines represent the performance baseline.

Experiments regarding the number of transferred layers, the two versions of the $\mathcal{D}_{Source}$, poses and SOBDs, and a proportion of the $\mathcal{D}_{Target}$ are carried out. This parameter-based TL considers only the feature extractors of the neural network, so it does not consider a semantic representation of the sequences. Nevertheless, when transferring only the first convolutional layer, the performance positively influenced the task, regardless of the data source. Parameter Transfer Learning helps when a small proportion of the training target set is available. Besides, source and target datasets with shared activities showed some improvements. These findings suggest that simple local temporal relations are rather generic and thus transferable. On the other hand, the more complex and task-related filters are not transferable when considering pixel-coordinates of poses as $\mathcal{D}_{Source}$.

6.4 DISCUSSION AND COMPARISON WITH STATE OF THE ART

Attr-based Transfer Learning for M-HAR is a combination of a descriptor extractor and a semantic attribute representation exploiting a fine-grained annotated source dataset for solving different TL scenarios. These scenarios handle situations when the source and target domains differ, specifically the multi-channel time-series Space, and the tasks, as Section 2.3 discusses. The Attr-based Transfer Learning for M-HAR is a composite of functions, namely $\mathbf{r}(\mathbf{h}(\mathbf{g}(\cdot)))$, following the proposed method shown in Figure 5.1.1.

This composite was built upon evaluating each of these functions in four parts shown in Section 6.3, as follow:

1. The composite function $\mathbf{h}(\mathbf{g}(\cdot))$ is given by the feature extractor and the fuser TPP layer of the proposed limb-oriented tCNN, namely the *IMU-TCNN*$_{\mathtt{TPP}}$. An evaluation of the *IMU-TCNN*$_{\mathtt{TPP}}$ is carried out on the source and target OBD datasets.

2. The *IMU-TCNN*$^{\mathtt{atrtibute}}_{\mathtt{TPP}}$ is subsequently evaluated on attribute-wise annotated LARa. This *IMU-TCNN*$^{\mathtt{atrtibute}}_{\mathtt{TPP}}$ is called Attr-IMU-tCNN. Additionally, Synthetic On-body Device (SOBD) exploiting LARa is also considered. SOBD from unconventional sources for M-HAR are also evaluted.

3. The function $\mathbf{r}(\cdot)$ or attribute representation, which is derived on target datasets $\mathcal{D}_{\mathtt{Target}}$ using the EA-Attr-IMU-tCNN.

4. The composite function $\mathbf{r}(\mathbf{h}(\mathbf{g}(\cdot)))$, Adaptable Attr-IMU-tCNN, is evaluated on different target datasets.

Table 6.4.1 presents a comparison of the best testing wF1 [%] from the overall method of this thesis on the target benchmark datasets with the state-of-the-art performance using very similar architectures. It shows either values taken from the literature or replicated by the author of this thesis following the training procedure given by the literature. Experiments within this work are carried out following the training procedure given in Section A.3 using Pytorch and an NVIDIA Geforce RTX 3090, repeating the experiments 5$\times$ under the similar training hyperparameters and controlling the source of the randomness of Python and Pytorch. This is for ensuring an evaluation with as controlled hyperparameters as possible, measuring the influence of the proposed four steps for evaluating the method, and ensuring repeatability.

TL in M-HAR was showing limited outcomes in the literature, mainly because the source and the target domains differ, e.g., the parameter TL in [OMR16] from the *Skoda* dataset to the Opp$_{ges}$, explained in Section 3.4; results shown in Table 6.4.2. The Attr-based Transfer Learning for M-HAR addresses TL via not only by parameter transfer but also via representation. Attr-based Transfer Learning for M-HAR allows improving M-HAR, even with a fraction of the training material, obtaining performance near similar

8 Experiments on Opp$_{ges}$ were carried out using an *TCNN-LSTM* with a one 128-LSTM layer after feature extraction and with a sliding window of 0.5 s with an overlap of 50%.

9 This performance is provided by MotionMiners GmbH, which is computed using a *TCNN* trained with their own company large dataset material.

Table 6.4.1: The wF1% of the best Attr-based Transfer Learning for M-HAR on the target datasets compared to similar related works using similar architectures. Results obtained within this thesis are given with the mean and SD from the wF1 [%] from carrying out 5× the experiments. These results are either replicating the related work's architectures, and deploying the proposed method, and are shown with colour ■. The reported results by the related work are shown in colour ■. Values marked with the symbol ▲, ▲, and ▲ refer to Transfer Learning for HAR using [LARa-M, LARa-Mb, LARa-SOBD] as the $\mathcal{D}_{Source}$, respectively. Values from OPD_A and OPD_B are computed from a 3-fold validation. $\mathcal{D}_T$ stands for $\mathcal{D}_{Target}$.

Networks	LARa-MM	Pamap2	Opp_{loc}	Opp_{ges}	MM	OPD_A	OPD_B
B-LSTM	-	87.2[HHP16] 84.2[TDF+18]	-	90.8[HHP16]	-	-	-
TCNN-LSTM	66.89 ± 1.12	86.91 ± 0.44	89.5[OMR16] 86.52 ± 0.24	91.5[OMR16] 51.0[BHMVL21][8] 89.14 ± 0.33	82.87 ± 0.63	65.10 ± 4.5	73.72 ± 7.25
TCNN	71.88 ± 0.34 65.71 ± 0.01 [AMRF22]	74.0[TDF+18] 88.43 ± 1.52	86.5[YNS+15] 87.8[OMR16] 86.52 ± 0.24	93.9[YNS+15] 85.1[OMR16] 89.23 ± 0.18	82.89 ± 0.58 84.00 [Rei21] 83.10[9]	66.47 ± 4.73 Acc = 69.2 ± 1.8 [GLMR+17]	74.60 ± 5.79 Acc = 73.9 ± 4.6 [GLMR+17]
IMU-TCNN^{attr}	73.04 ± 0.97	86.95 ± 0.50	83.54 ± 0.68	88.22 ± 0.30	84.76 ± 0.34	65.07 ± 4.58	74.28 ± 6.70
Proposed Method							
[100%]$\mathcal{D}_T$	73.26 ± 0.41 ▲	92.21 ± 0.31 ▲	89.09 ± 0.04 ▲	90.28 ± 0.17 ▲	86.22 ± 0.50 ▲	70.56 ± 4.14 ▲ Acc = 72.05 ± 1.97	75.52 ± 7.66 ▲ Acc = 76.09 ± 7.20

methods using the entire training material. Following the results in Figure 6.3.6 and summarised in Table 6.4.2, Attr-based Transfer Learning for M-HAR on the LARa datasets improves the performance for all target datasets in all the scenarios where the proportion of the training material is limited, considering $[10, 20, 40, 60, 80]$% of the training material from the target datasets. Furthermore, it considers when the amount of OBDs is different, e.g., TL between LARa-Mb with five OBDs to LARa-MM with three OBDs within the same domain and task, to OPD with three OBDs within the different domain and similar tasks, to Pamap2 with four OBDs within the different domain and tasks. These outcomes show that the temporal relations and the representation extracted from the LARa can be used for improving classification on datasets with different class activities.

The MM dataset recorded from real and challenging intralogistics scenarios profits enormously from the Attr-based Transfer Learning for M-HAR and the LARa dataset. The LARa dataset is a laboratory dataset created to address intralogistics M-HAR problems involving recording densely sampled human poses and inertial measurements. The *IMU-TCNN^{attr}* trained with the LARa can be used for predicting activities of the MM dataset. A function $\mathbf{r}(\cdot)$ for the MM in terms of the attribute representation of the LARa can be manually given[10], as these datasets belong to the same application. The *IMU-*

10 The author of this thesis gives this mapping following a coarse description of the MM activities provided by MotionMiners GmbH.

Table 6.4.2: The wF1% of the best Attr-based Transfer Learning for M-HAR using the *IMU-TCNN*^{attr} on the different proportions of the target datasets, shown in colour ■. Results obtained within this thesis are given with the mean, and SD from the wF1 [%] from carrying out 5× the experiments. The reported results by the related work are shown in colour ■. Values marked with the symbol ▲, ▲, and ▲ refer to Transfer Learning for HAR using [LARa-M, LARa-Mb, LARa-SOBD] as the $\mathcal{D}_{Source}$ respectively, where the symbol attached to each of the classification performance wF1% shows the best performance out of the three the source datasets, $\mathcal{D}_{Source}$. Experiments for all the $\mathcal{D}_{Source}$ are found in the Appendix Section A.4. Values from OPD_A and OPD_B are computed from a 3-fold validation. $\mathcal{D}_T$ stands for $\mathcal{D}_{Target}$.

Proportions	LARa-MM	Pamap2	MM	Opp_{loc}	Opp_{ges}	Opp_{ges}[OMR16]
[100%]$\mathcal{D}_T$	73.26 ± 0.41 ▲	92.21 ± 0.31 ▲	86.22 ± 0.50 ▲	89.09 ± 0.04 ▲	90.28 ± 0.17 ▲	≈41.0[100%]$\mathcal{D}_T$ ≈30.0[100%]$^{\mathcal{D}_T}_{Skoda}$
[80%]$\mathcal{D}_T$	70.39 ± 1.18 ▲ [75%]$\mathcal{D}_T$	90.91 ± 1.17 ▲	85.51 ± 0.68 ▲	81.62 ± 0.62 ▲	87.37 ± 0.17 ▲	≈30.0[75%]$^{\mathcal{D}_T}_{Skoda}$
[60%]$\mathcal{D}_T$	68.42 ± 0.62 ▲ [50%]$\mathcal{D}_T$	91.21 ± 0.47 ▲	85.86 ± 0.66 ▲	77.58 ± 1.18 ▲	84.35 ± 0.17 ▲	≈26.0[50%]$^{\mathcal{D}_T}_{Skoda}$
[40%]$\mathcal{D}_T$		90.95 ± 0.60 ▲	84.19 ± 0.81 ▲	73.56 ± 2.66 ▲	81.85 ± 0.35 ▲	
[20%]$\mathcal{D}_T$	64.22 ± 1.15 ▲ [25%]$\mathcal{D}_T$	57.94 ± 1.81 ▲	83.61 ± 0.70 ▲	67.77 ± 0.87 ▲	81.07 ± 0.09 ▲	
[10%]$\mathcal{D}_T$		60.78 ± 1.47 ▲	82.92 ± 1.12 ▲	56.40 ± 2.69 ▲	77.54 ± 0.77 ▲	

$TCNN\uparrow^{MM}_{LARa}$ using this manually given mapping performs M-HAR with a wF1 = 80.80[%] of performance. Here, the architecture is directly deployed with [0]% training material, with only a coarse description of the activities, enough to relate the LARa attributes. This outcome shows the feasibility of transferring not only the temporal relations per limb but also a late fusion strategy of the limb temporal features and a compact descriptor representative of the activities, along with an activity representation within the application domain.

In comparison to the state-of-the-art architectures, it is noticeable the boost in the performance for the Pamap2 and Opp datasets. The $TCNN\uparrow^{\mathcal{D}_{Target}}_{\mathcal{D}_{Source}}$ pretrained with $\mathcal{D}_{Source}$ = [LARa-SOBD,LARa-M,LARa-Mb] on the Pamap2 show the best performances, even when using [60%] of the training material. The LARa-SOBD becomes the best source under the case of [10, 20, 40, 60, 80]% of the Pamap2 training material. This outcome shows the potential of the SOBD; hereafter, these SOBD are another way of exploiting the human poses measured with a high sampling rate from the marker-based MoCap. On the Opp_{ges} dataset, Attr-based Transfer Learning for M-HAR makes transferability possible when having [10, 20, 40, 60, 80]% of the training material as compared to [OMR16] using [50, 75, 100]%. The performance on the Opp is significant compared to the same replicated baseline of the literature.

Furthermore, not only the SOBDs from marker-based MoCap improves performance, but also SOBDs from human pose annotations of video-based HAR can serve as source material for improving classification under the case of having a scarce amount of data, following Figure 6.3.7. This parameter TL scenario provides samples of ADL. However, sequences of human poses as pixels depend strongly on the video sampling rate, which

limits the usage of these SOBD for computing short human movements like attributes, as this thesis considers. In that case, the SOBDs from LARa and a marker-based MoCap show an advantage. Nevertheless, SOBD from human pose annotations of video-based HAR improve performance on shared activities between the source and the target datasets. They become handy as usually video-based HAR datasets contain a wider range of activities.

The EA allows finding semantic attributes in the target dataset within the representation from the LARa dataset. The EA uses a pretrained network that handles the input of different multi-channel time-series Space and is trained on the LARa for finding the attributes on the target sets efficiently. This is because the EA does not need to train its network. However, fine-tuning the architecture on the target dataset is required for adjusting to the new task.

The Attr-IMU-tCNN$_{LARa}$ is anyhow limited to $M = 19$ semantic attributes that are related to intralogistics activities; yet, these attributes are still enough to be transferable. In summary, the learnt attribute representations of the target datasets are semantically plausible to the activities. This learnt representation allows direct deployment of the somehow general Attr-IMU-tCNN$_{LARa}$ to a new scenario. The Attr-IMU-tCNN can serve as an out-of-the-shelf classifier that provides semantic attributes of the activities of a target dataset. These attribute predictions might become helpful at first glance over a target scenario where no training material is available. Besides, they become handy for facilitating the annotation process of multi-channel Time-Series data for M-HAR, as Section 7.2 will present.

7 PRACTICAL APPLICATION - HUMAN ACTIVITY RETRIEVAL FOR ANNOTATING HAR DATA

M-HAR classifies sequential data of humans performing a task. Irrespective of M-HAR methods, annotated datasets remain of limited size. The recording and annotation process of measurements from OBDs is very time-demanding, monotonous, and labour demanding [YPS+18, AAR+21]. Due to the large intra- and inter-class variability of human motion, a high quantity of observations with motion repetitions from the same or different persons is necessary [OR16]. However, data collections for a M-HAR system face challenges regarding environment settings, number of participants, number of devices, and configurations [BBS14, TDF+18]. For example, OBDs may be configured with different sampling rates and resolutions. Besides, the number of devices and their position on the human body may vary [RFCT13].

Mainly, the activity labels from a large number of observations have to be manually annotated. Nonetheless, raw inertial measurements from OBDs are visually hard to interpret, i.e., the activity is not apparent only by regarding sequences of sensor measurements. Therefore, additional video streams are necessary to make an activity distinguishable by the manual annotator. This alternative involves synchronising the OBD measurements with the video stream to observe the human action for setting the corresponding label. This form of annotation becomes a problem in real scenarios where video streams might not be allowed [BBS14, FAt16, Rei21]. Besides, simultaneous recordings and careful synchronisation is of need. Furthermore, they also increase the computational requirements. For example, the authors in [FAt16] took 26 min per 1 min of annotated sequence using synchronized videos and OBD data. Fine-grained annotations or including semantic information additionally increase the manual effort; for example, the annotators in [RMRN+20, AAR+21] respectively took 42.5 and 35.3 min per 1 min annotating attribute representations of very short-term windows.

Nevertheless, manual annotations are still prone to inconsistencies [NDCT17]. These inconsistencies are mainly because of the variability of human movements and the lack of a standard definition of human actions. Furthermore, dataset authors rarely mention

or document the annotation protocol, effort and consistency [RNMR+19]. Therefore, the data collection and annotation for M-HAR require much manual effort, clear instructions or training, and a practical annotation tool.

Exploiting that a considerable amount of data is available, but annotations are missing, supervised, and unsupervised methods automatically predict pseudo-labels for a given set of data [EH20]. However, the automatically produced labels need to be revised as the models tend to estimate wrong predictions with high confidence [APH+21, AAR+21]. For example, the authors in [AAR+21] take 18.16 min per 1 min of annotated sequence when manually revising activity predictions of a tCNN. This annotation effort is roughly halved when comparing to entirely manual.

Semantic attributes can represent human activities helping in Zero-Shot HAR and Transfer Learning for HAR. Their advantage is that classification tasks can be carried out without much annotation effort, or in cases where data are highly unbalanced, the quantity of samples is large, and the testing set contains unseen object classes, as Chapter 5 and Chapter 6 presented. They are also helpful for searching tasks, e.g., word spotting, as Chapter 4 presents.

This chapter proposes HARetr for annotating multi-channel Time-Series data for M-HAR. This retrieval ranks segmented windows of data. It exploits predictions of attributes predictions from the Attr-IMU-tCNN of segmented marker-based MoCap data and ranks them according to some similarity to a query. A simple manual annotation then consists of accepting or rejecting the ranked segments. This revision further decreases the burden of an annotator compared to semi-automated annotation, where revision happens along the entire recording.

This chapter first presents a related work of semi-automated annotation for M-HAR. Second, it proposes Human Activity Retrieval as semi-automated annotation method, introducing along two retrieval metrics. Then, it explores two similarity measurements. It finally compares the performance of the retrieval against manual annotations.

7.1 SEMI-AUTOMATED ANNOTATIONS FOR HAR

Apart from the burden of gathering data, recording in real scenarios is prone to be disturbed by external factors. Controlled environments are appealing for gathering data, as sensor measurements are lesser affected by noise [VEA+15, DTR18], and recording sessions can be conducted and repeated under different settings. Marker-based MoCap systems are currently used for conducting measurements in controlled environments. For example, Dalmazzo et al. [DTR18] have used marker-based MoCap and OBD datasets for

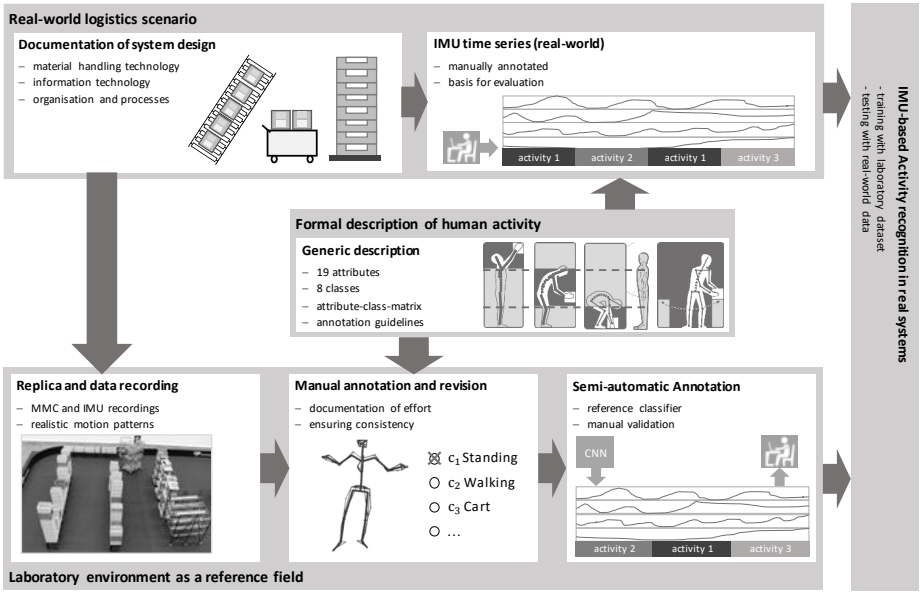


Figure 7.1.1: Semi-automated annotation method for logistics scenarios. Recording and annotating large amount of data is tackled by: training a CNN with data recorded from a controlled environmend closed to reality. Image taken from [Rei21].

predicting violin bow activities. They concluded that natural human movements could be achieved in such controlled scenarios.

Dr. Reining [Rei21] proposed gathering high-quality data from a controlled environment or laboratory scenario for intralogistics. This data is close to reality and has proven useful for real-cases scenarios. He proposed a semi-automated annotation of multi-channel Time-Series data. They created laboratory setups of three scenarios in the intralogistics. Domain experts set up these laboratory scenarios as much as near to reality with regards to the characteristics of human activities. OBD recordings were annotated based on annotations of simultaneously-recorded marker-based MoCap recordings.

Subjects carried out activities of three manual processes happening in the intralogistics. In the first scenario, subjects pick up items from boxes at three different heights, recreating picking from shelves. Afterwards, they put the items in boxes on a cart. Once the cart is full, subjects carry the cart to a conveyor-like area, where boxes are unloaded. In the second scenario, subjects pick up items from shelves at different heights and place them on a flow-through cart. Here, the items are scanned first before picking. Finally, in the third scenario, the subjects pick up items from the loaded flow-through cart and place them on a packing table. Subjects locate the item inside an empty box and fill the box

with wrapper plastic. Once a carton box is full, the subject closes and taps it. In the end, subjects pick up the packed box on a conveyor-like table. Subjects were neither instructed how to pick the item or boxes, nor how to wrap or tap the filled boxes. They were only provided with the task, and they performed it in a way they felt comfortable with to ensure natural motion behaviour.

Dr. Reining [Rei21] proposed deploying marker-based MoCap recordings for annotating OBD recordings. The annotations consist of activities and semantic attributes for short-duration window sequences—with a minimal activity duration of 100 ms. The authors compare manual annotation vs. semi-automated annotation, collecting the results in [AAR+21]. A semi-automated approach is proposed using predictions of a IMU-tCNN from marker-based MoCap segments. Following this, a human revises the annotations, and they corrects the activity and its attributes mislabeling. The authors reduced the annotation time by half compared to manual annotation but kept annotation consistency.

Annotators correct the predicted activities and attributes of the segmented windows by observing the entire recording in an orderly manner. However, this revision also represents a burden for the annotators, who must observe and concentrate on the revision. This thesis proposes to use retrieval instead of the classification step in the semi-automated annotation.

7.2 HUMAN ACTIVITY RETRIEVAL

In contrast to classification tasks, retrieval returns a list of candidates ranked by a certain similarity score considering a user-defined query. There exist a variety of different retrieval scenarios depending on the query type. For example, in content-based image retrieval [ZLT17, CLW+21], one of the most prominent scenarios is to use an example image as query, referred to as Query-by-Example (QbE). In that case, a user first selects a sample from the data as the query.

The QbE was successfully applied in document analysis and exhibited a shortcoming [SF18]. However, the user first needs to browse through the data to perform retrieval, searching for a sample. This search can be highly time-consuming and tedious for rare examples. Furthermore, if the user cannot find an example, this might solve the QbE task. Therefore, redefining the query type is an appropriate approach to overcome the need for an example. Instead of retrieving candidates given an example-based query, a retrieval system sorts the candidates with respect to semantic information.

This retrieval is based on encoding semantics as binary attributes, sometimes called visual attributes [LNH14]. The retrieval system then has to learn a mapping from the data input to the attribute representation. It seeks to project the query class and the data

input into the attribute space in which the retrieval task boils down to a nearest neighbour search [SF18]. This semantic-based retrieval was successfully demonstrated for keyword spotting [SF18] and cuneiform spotting [RRMF18] were it is referred to as Query-by-String (QbS) and Query-by-eXpression (QbX), respectively, as Section 4.1 presents.

Some retrieval approaches use an attribute representation of data inputs and rank them according to a similarity metric to a query in attribute space. In [FEHF09, LNH09], attribute-based representations are introduced to the field of computer vision. Based on the idea of representing classes with binary attributes, Lampert et al. [LNH14] leverage these attributes to deal with disjoint training and testing sets, also known as zero-shot learning; see Section 2.4 for details.

7.3 ATTRIBUTE-BASED HUMAN ACTIVITY RETRIEVAL

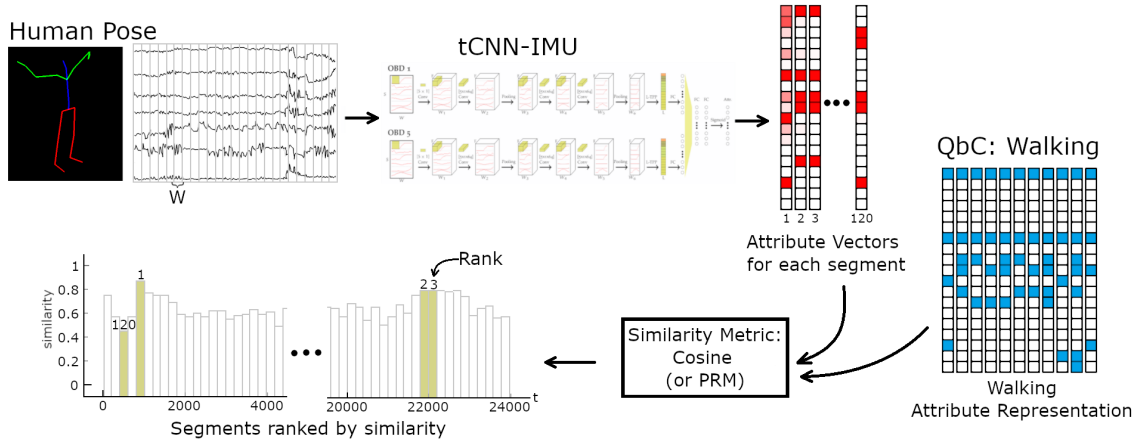


Figure 7.3.1: Human Activity Retrieval (HARetr) for annotating multi-channel Time-Series recordings for M-HAR. First, a recording of a human is segmented and forwarded through the Attr-IMU-tCNN. The resulting attribute vectors are compared against the attribute representation of a class using a similarity metric. The similarity of a segment is the similarity to the nearest neighbour of the queries attribute representation. The retrieval list is then sorted in descending order by the score value. The numbers on the highlighted bars show the order in which they would be presented to the Annotator.

The viability of retrieval for annotating multi-channel Time-Series data for M-HAR tasks is evaluated. The retrieval is based on works in word spotting [SF18, RRMF18] and will be here considered as an alternative method of obtaining automated annotations for the semi-automated annotation pipeline [Rei21], given in Section 7.1. HARetr replaces the automatic annotation but involves the annotator in the process. With retrieval, the

annotation process might be more straightforward as the human annotator or reviser can focus on one activity class at a time and they accepts or rejects samples of that activity class instead of having to switch mentally between classes within a complete recording.

In word spotting, there are usually two types of queries useful to search for a word: QbS and QbE. Here, the focus will be on three equivalents of QbS in M-HAR. There are three possible retrieval methods: Query-by-Class (QbC), Query-by-Attribute (QbA), and Query-by-Attribute Representation (QbAR).

Figure 7.3.1 presents the retrieval-based annotation method. HARetr provides a list of results sorted by relevance with respect to a given query. A Attr-IMU-tCNN computes a set $\hat{\mathbf{A}} = \left\{\hat{\mathbf{a}}^{(n)}\right\}_n^N$ of attribute representations from N segmented windows $\left\{\mathbf{X}^{(n)}\right\}_n^N$ out of a marker-based MoCap recording. Recap from Subsection 5.1.2, there is a mapping $\mathbf{r}(\mathbf{a}) = y$, where each attribute $\mathbf{a}$ corresponds to exactly one activity class y, i.e., $\mathbf{p}(y|\mathbf{a}) = [[y = \mathbf{r}(\mathbf{a})]]$. Instead of predicting a single class $\hat{y}$ using the Equation 5.1.10 and Equation 5.1.8, HARetr sorts the segmented windows according to a similarity of a query class y_q and its attributes $\{\mathbf{a}^{y_q}\} \subset \mathbf{A}$.

$$\mathbf{SW}_q = \underset{\{\mathbf{X}^{(n)}\}_n^N}{\operatorname{argsort}}\, d(\{\mathbf{a}^{y_q}\} - \mathbf{g}(\mathbf{X})), \tag{7.3.1}$$

with $d(\cdot)$ being either the cosine similarity or the PRM Equation 4.1.2 [RRMF18], presented in Section 4.1, and $\mathbf{g}(\cdot)$ the Attr-IMU-tCNN, predicting an attribute representation from a segmented window.

QbC sorts segments based on a Nearest Neighbour (NN) using Equation 7.3.1. In the case of QbAR, the query is a single attribute representation $\mathbf{a}_q$, so segments are sorted considering the similarity of the network outputs $\mathbf{g}(\cdot)$ and $\mathbf{a}_q$. For QbA, the only relevant network output is the one corresponding to the queried attribute, which can be used to sort out all segments. All other attributes can be ignored.

7.4 ANNOTATION TOOL

The author of this thesis in collaboration with Mr. Erik Altermann presented a tool, with a Git repository in [MRA22], for manual and semi-automated annotation of human activities based on sequences of skeletons and OBD data, detailed in [AAR+21, RMRN+20] with annotation results summarised in [Rei21]. An annotator visualises skeleton sequences from a human performing an activity for a given marker-based MoCap recording $r \subset R$ with R recordings from a dataset. The annotator segments the recording r in windows. For each window, the annotator manually sets the start and end of an activity and provides

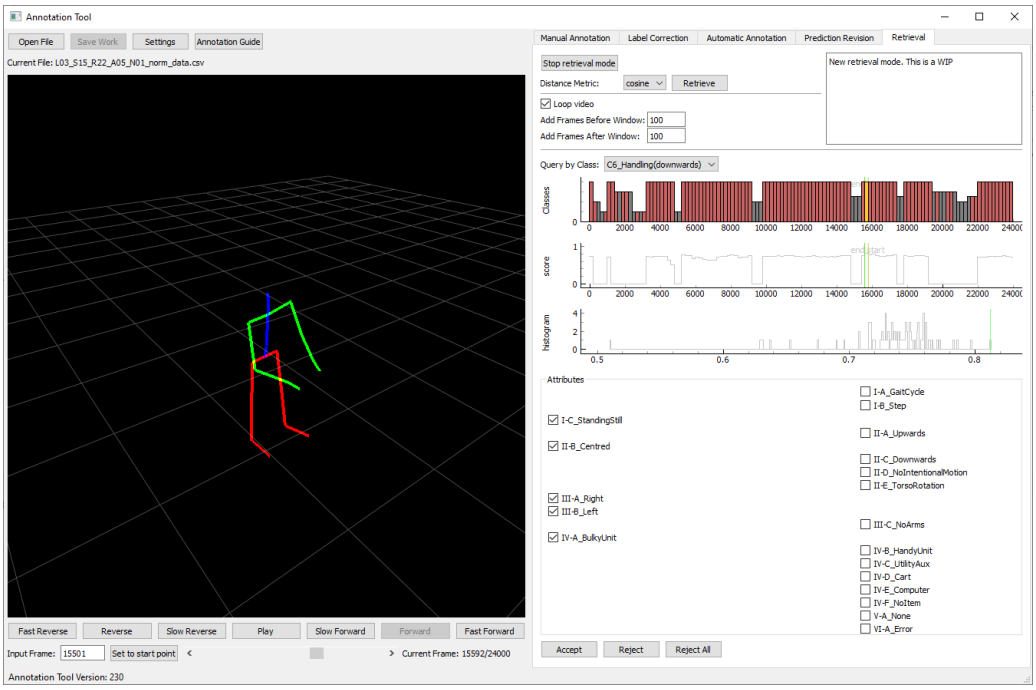


Figure 7.4.1: Annotation Tool with a retrieval annotation modus.

its activity class $y \in Y$ and its semantic attribute vector a^M. Among the attributes, there is an error attribute that supports manual revision. The annotation tool highlights windows with set error attribute for revision, using a manually crafted rule set for viable attribute combinations.

In [AAR+21], the annotation tool is extended to include predictions of DNNs. A pre-trained deep network predicts the activity classes and the corresponding attribute representation vector of non-overlapping windows of size W, i.e., it classifies all the segmented windows. Besides, annotators are liberated from manually setting the starting and ending of the activities. The annotators then revise the activity and attribute predictions.

This thesis further extends the tool with a retrieval mode. The annotation tool can be found in [MRA22]. The tool uses the Attr-IMU-tCNN to predict the attributes of a segment. First, the annotator chooses an activity class for QbC. Then, the segments get sorted using the pipeline shown in Figure 7.3.1. Finally, the annotator can accept or reject any segment based on whether that segment is relevant to the queried class.

7.5 ANNOTATION EXPERIMENTS

Two experiments are conducted to evaluate the retrieval-based annotations for M-HAR. First, HARetr is evaluated on the LARa dataset for M-HAR, measuring its global performance. Second, the annotation tool with a HARetr mode is used for an annotation task. Then, the effort of annotating recordings of a dataset is evaluated in terms of annotation time and consistency, following the same procedure from [AAR+21, Rei21] for comparison.

7.5.1 *Retrieval Evaluation*

Retrieval tasks can be globally measured for a dataset. A retrieval metric allows to quantify the performance of retrieval without carrying on an specific retrieval experiment.

Retrieval Metrics

The interpolated Average Precision (AP) is used as performance metric for retrieval tasks [BYRN11]. AP combines precision and recall for evaluating ranked lists. The idea is that the most relevant result to any given query should be at the start of the list. Precision is the percentage of retrieved relevant results over the retrieved results. Recall is the percentage of retrieved relevant results out of all relevant results.

The AP is computed for every single query as follows:

$$AP(q) = \frac{1}{T} \sum_{n=1}^{N} P(n) \cdot rel(n), \tag{7.5.1}$$

where $P(n)$ denotes the precision for a cut-off at position n in the retrieval list. The indicator function $rel(n)$ evaluates to 1 if the n-th position in the list is relevant with respect to the query and 0 otherwise. The total amount of relevant elements is represented by T, and the length of the retrieval list is represented by N. In LARa scenario, the length of the retrieval list, N, is defined by the number of windows within one single recording $r \in R$.

Equation 7.5.2 calculates the performance for each recording r by computing the mean overall average precisions for Q queries from a single recording, r.

$$mAP_r = \frac{1}{Q} \sum_{q=1}^{Q} AP_r(q), \tag{7.5.2}$$

where Q denotes the total amount of queries and $AP_r(q)$ the average precision for query q in a record r. Afterwards, Equation 7.5.3 computes the overall retrieval performance by calculating the mean over all mAPs from a set or R recordings.

$$mAP = \frac{1}{R}\sum_{r=1}^{R} mAP_r \tag{7.5.3}$$

This mAPs shows the viability of the method for the entire dataset and gives insights on how well it works on larger datasets.

Query-by-Attribute Representation (QbAR) and Query-by-Class (QbC)

Retrieval is evaluated for different sequence lengths W, and two similarities, the cosine and PRM. Multiple Attr-IMU-tCNN networks have been trained on $LARa_{MoCap}$ for $W = [50, 100, 150, 200, 400]$, following Section A.3. The sequence segments are extracted following a sliding-window approach with window size of W, step size of $s = \frac{W}{4}$, i.e., 75.0% overlapping. To compare, the average class length in the LARa dataset is 4.12s; that is approx. 800 frames at 200Hz.

Table 7.5.1 shows mAP vs. different W. For QbC, Attr-IMU-tCNN W200 outperformed every other network. For QbA, the Attr-IMU-tCNN W400 outperforms the Attr-IMU-tCNN W200 when using the cosine similarity. For the PRM similarity, the Attr-IMU-tCNN W200 outperforms the Attr-IMU-tCNN W400. As the Annotation Tool, in Section 7.4, uses QbC all further experiments will be done using the Attr-IMU-tCNN W200.

Table 7.5.1: mAP metric for QbAR and QbC using the Attr-IMU-tCNN on different window sizes W, and similarities on the $LARa^{Test}$ dataset.

Network	**QbAR**		**QbC**	
	COS	PRM	COS	PRM
Attr-IMU-tCNN W50	0.350	0.360	0.609	0.619
Attr-IMU-tCNN W100	0.382	0.396	0.629	0.635
Attr-IMU-tCNN W150	0.414	0.426	0.647	0.658
Attr-IMU-tCNN W200	0.433	**0.445**	**0.651**	**0.661**
Attr-IMU-tCNN W400	**0.439**	0.441	0.616	0.622

Query-by-Attribute (QbA)

Table 7.5.4 shows the mAP, column in colour , for each attribute. The attributes in the first three groups have high mAP values, meaning retrieval is viable for M-HAR. Since the mAP values for Attributes in group IV are all rather low, the annotator should pay special attention to these attributes to ensure that everything is correctly annotated.

7.5.2 *Annotation Evaluation*

A set of annotations for the $\text{LARa}^{\text{Test}}_{15,16}$ was carried out, evaluating the feasibility of retrieval on an existing semi-automated annotation framework. Four annotators—two experts and two beginners—annotated eight recordings from $\text{LARa}_{15,16}$ using the annotation tool described in Section 7.4. The tool uses the *Attr IMU-tCNN W200*. This annotation follows the same annotation procedure in [AAR+21, Rei21]. Here, the semi-automated annotation [AAR+21, Rei21] are benchmarked and compared against semi-automated annotation using retrieval mode. Furthermore, semi-automated annotation with retrieval is compared vs. manual+revision annotation process [Rei21].

Annotation Time

Figure 7.5.1 shows the annotation times of the retrieval-based annotation using the cosine and PRM similarities. Besides, it compares with the manual annotations of [AAR+21]. Annotations were faster using the retrieval-based annotations for both similarities vs. the manual annotations. A learning process by the annotators is seen as the variation in the annotation time decreases with more recorded units. Different from semi-automated annotations from [AAR+21], a revision is integrated into the retrieval process, as annotators get suggestions based on the ranked segments, which have to be accepted or rejected. The annotation tool also brings the option to change the attribute representation before accepting manually. Following a ranked order based on QbC, the expert annotators reported accepting the annotations and straight correcting the attribute representation as less mentally demanding. Beginners annotators experienced a comfortable annotation process for *Walking*, *Standing* and *Handling* without the attribute *Utility/Auxiliary*.

Class-wise Consistency

Cohen's Kappa (κ) is a widely-used consistency metric. Cohen's κ emphasizes agreement, and it does not consider one annotation as the ground truth, contrary to performance metrics such as accuracy or the F1-measure.

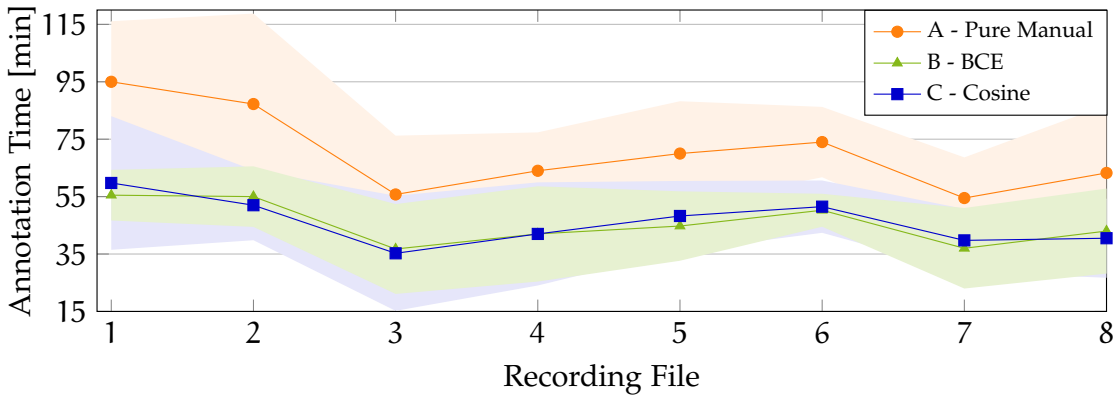


Figure 7.5.1: Annotation-time of retrieval-based and manual annotation [AAR+21, Rei21] procedures for each of the eight recordings considered from the $\text{LARa}^{\text{Test}}_{15,16}$. Retrieval-based approaches use cosine and PRM similarities.

The Cohen's κ is defined as

$$\kappa_{A_a,A_b} = \frac{Pr(a)_{A_a,A_b} - Pr(e)_{A_a,A_b}}{1 - Pr(e)_{A_a,A_b}}, \tag{7.5.4}$$

where $Pr(a)$ represents the actual observed agreement among two annotation runs A by the same or two different individuals; here, considered the case of two different individuals. $Pr(a)$ refers to the number of samples that both annotators agree on, divided by the total number of samples, also known as the accuracy. $Pr(e)$ is the expected chance agreement, and it considers that the two annotators may have guessed the same label by chance. It is defined as

$$Pr(e)_{A_a,A_b} = \frac{1}{M^2} \sum_{c=C}^{C} A_a^c A_b^c, \tag{7.5.5}$$

being M the number of samples, C the activity class or attribute, and A_a^c the number of times annotator a predicted class c [RMRN+20].

The semi-automated approach using retrieval is compared against the manual+revised annotation procedure from [AAR+21]. The later ones will be considered the ground truth. Table 7.5.2 presents the comparison of the annotations from the four annotators vs the ground truth of the $\text{LARa}^{\text{Test}}_{15,16}$. It shows different metrics, classification metrics (Acc, wF1 and mF1), and consistency ones κ—here, manual annotations are not considered ground truth. In general, the κ values lie within a small interval and hint towards a substantial agreement, i.e., $0.61 < \kappa \leqslant 0.8$.

Table 7.5.2: Consistency metrics: Acc. [%], wF1 [%], mF1 [%], and between-annotator consistency κ of activity class annotation of $\text{LARa}^{\text{Test}}_{15,16}$ by 4 annotators.

Metric	Cosine				PRM			
	A1	A2	A3	A4	A1	A2	A3	A4
Acc	86.75	84.87	85.42	83.45	85.14	82.08	83.32	83.12
wF1	87.51	86.64	87.52	85.43	86.45	85.35	85.69	81.86
mF1	61.88	58.33	56.87	56.78	59.90	53.32	55.65	57.88
kappa	79.61	76.47	78.01	74.44	77.11	71.78	75.02	74.41

Table 7.5.3: Between-annotator consistency κ of activity class annotation of 8 recordings (960s) from two subjects by 4 annotators. : consistency among annotators using the cosine similarity. : consistency among annotators using the PRM similarity.

Procedure		Annotator			
		A1	A2	A3	A4
Annotator	A1	-	0.7955	0.7704	0.7658
	A2	0.7652	-	0.7859	0.7960
	A3	0.7926	0.7414	-	0.7867
	A4	0.7623	0.7548	0.7725	-

Table 7.5.4: Semantic Attributes in LARa Dataset. In gray: mAP for QbA using the Attr-IMU-tCNN W200 on the $\text{LARa}^{\text{Test}}_{15,16}$ dataset. Each attribute's mAP was calculated independently. In white: and average between-annotator consistency κ of the four annotators following the retrieval-based approach on the $\text{LARa}^{\text{Test}}_{15,16}$. Manual annotations $*$ are taken from [AAR+21].

Attributes	mAP	Manual*	Cohen's κ	
			COS	PRM
I-A Gait Cycle	0.833	0.91	0.67	0.66
I-B Step	0.744	0.85	0.36	0.30
I-C Standing Still	0.887	0.87	0.69	0.64
II-A Upwards	0.888	0.98	0.83	0.81
II-B Centred	0.828	0.91	0.82	0.80
II-C Downwards	0.561	0.99	0.83	0.84
II-D No Int. Motion	0.838	0.92	0.81	0.80
II-E Torso Rotation	0.080	-	-	-
III-A Right Hand	0.941	0.92	0.67	0.74
III-B Left Hand	0.838	0.93	0.73	0.75
III-C No Hands	0.647	0.98	0.84	0.87
IV-A Bulky Unit	0.441	0.95	0.79	0.75
IV-B Handy Unit	0.560	0.87	0.55	0.63
IV-C Utility/Auxiliary	0.310	0.92	0.58	0.59
IV-D Cart	0.568	1.00	1.00	-
IV-E Computer	0.287	1.00	1.00	1.00
IV-F No Item	0.784	0.98	0.87	0.89
V-A None	0.072	-	-	-
VI-A Error	0.000	-	-	-
All Attributes	0.743	-	-	-

Attribute-wise consistency

Table 7.5.4 presents the between-annotator consistency in terms of κ among the four annotators and the two similarities for each attribute. After the semi-automated annotation, the between-annotator consistency for the attributes remains substantial. Even though the Attr-IMU-tCNN network performs poorly for attributes in the IV group, annotators correct the predictions—except on the *Handy Unit*. The attributes *Torso Rotation*, *None*, and *Error* were not selected by the annotators.

CONCLUSION 8

This thesis proposes an approach for Transfer Learning for Human Activity Recognition to improve automatic multi-channel time-series Human Activity Recognition (M-HAR). M-HAR task classifies human movements from sequential data recorded by from multiple On-body Devices (OBDs). Transfer learning allows the extend of the usability of usually large annotated data from source domains and trained models to related problems.

This thesis proposes a parameter- and feature-representation transfer learning method that uses a semantic attribute representation and a DNN suitable for M-HAR under different domains. This method is called Attr-based Transfer Learning for M-HAR. It seeks to address different transfer learning scenarios, such as the number of sensors on the subjects and different tasks or activities, without needing a large amount of annotated data from the target domain. These transfer learning scenarios handle situations where the source and target domains differ, i.e., multi-channel time-series Space and tasks.

Attr-based Transfer Learning for M-HAR considers different levels of transferability to solve M-HAR on benchmark target datasets: first, an architecture that handles different dataset configurations; it creates fixed-size representations of OBD recordings that are representative of the human limbs; second, semantic activity representations of activities that are found in target datasets; and third, data from a variety of source datasets.

Attr-based Transfer Learning for M-HAR proposes:

1. an architecture to compute a fixed-sized deep representation from sequence segments considering the subjects' limbs and a temporal pooling strategy. This architecture is referred to as IMU-tCNN. The IMU-tCNN is thought to allow transferability among datasets with different sensor setups. An evaluation of the IMU-tCNN is carried out on the different benchmark datasets. Relevant to be mentioned is the LARa dataset. This dataset comprises a large collection of subjects performing intralogistics activities. The dataset contains synchronised annotated recordings from three different sensor setups, namely, a human-joint poses recorded by a marker-based MoCap, inertial measurements from five human limbs by 5-set commercial OBDs, and inertial measurements from the torso and hands, recorded by a 3-set OBDs. Besides, LARa is densely annotated, with semantic attributes per sample.

Therefore, the LARa dataset and its attributes become the primary source dataset to be transferred.

- In addition, synthetic data for initialising a deep model is proposed. Synthetic On-body Device (SOBD) data is derived from sequences of human poses. This derivation takes advantage of the LARa-M and three datasets intended for video-based HAR and human pose estimation. The later ones concretely refer to the annotations of pixel coordinates as sequences of human joints. The sequences of pixel coordinates of human joints are considered simplifications of pose estimations. It considers the sequences of poses and pixel coordinates as individual channels.

2. an attribute-based classification of activities, considering multiple attribute representations per activity class. This Attr-IMU-tCNN is trained for predicting attributes. The attribute-based *IMU-TCNN* maps input sequences to a semantic attribute representation.

3. a search of semantic attributes on target datasets using the Attr-IMU-tCNN trained with the LARa and its attribute representation. An Evolutionary Algorithm (EA) algorithm is proposed to learn the attribute representation better suited for solving M-HAR. The EA randomly combines and mutates two parents' attribute representations, creating a set of offspring. These offsprings are evaluated using the Attr-IMU-tCNN from LARa. The EA selects the best offspring for the next generation using the validation performance as fitness. After several generations, the EA can find attribute representations of the target datasets in terms of the LARa attributes. Furthermore, these learnt attribute representations in terms of LARa are deployed for transfer learning. An additional EA finds attributes that better suit the target attributes by also training the Attr-IMU-tCNN. These attributes are better than target datasets, compared to the ones in terms of the LARa, as there is an additional degree of freedom with respect to the network. However, the attributes do not hold any semantic meaning, and the EA takes $30\times$ longer.

4. Adaptable Attribute IMU-Temporal Convolutional Neural Network (Adaptable Attr-IMU-tCNN). The Adaptable Attr-IMU-tCNN combines the limb-oriented Attr-IMU-tCNN and the learnt attributes. The Adaptable Attr-IMU-tCNN and the semantic attribute representations allow transferability across target domains with different domains and tasks. Transfer Learning for HAR is not only approached by parameter-based transfer via fine-tuning using the Attr-IMU-tCNN from LARa but also utilising a semantic attribute representation in terms of a known attribute representation.

Besides, it exploits physical but related measurements for transferability through the SOBDs. It handles situations with different numbers of devices, duration variations and disjoint activity classes. Furthermore, the Adaptable Attr-IMU-tCNN takes advantage of the LARa dataset, a large dataset composed of OBDs and human pose recordings and fine-grained annotations of semantic attributes. The Adaptable Attr-IMU-tCNN is evaluated on different target datasets. Experiments are carried out regarding the number of transferred layers, the OBD and SOBD of the source datasets, and a proportion of the target dataset.

In general, the performances of all architectures improved for all the target datasets, and most evidently when deploying a proportion of their training material. This outcome suggests that the temporal convolutional filters are rather general as they learn local temporal relations of human movements related to the semantic attributes, independent of the number of devices and their type. Adaptable Attribute IMU-Temporal Convolutional Neural Network (Adaptable Attr-IMU-tCNN) deploys the learnt attributes from the EA but also uses a manually given attribute representation given by the description of the target set. The target dataset, in this case, is domain related to the source dataset; thus, the manually annotated representation is still usable without the need to perform a search via a EA. Besides, the learnt convolutional filters are activity-class dependent. Hence, the classification performance on the activities that are shared among the datasets improves.

The proposed Adaptable Attr-IMU-tCNN closes the research gap of employing Transfer Learning for improving M-HAR as it gives an approach for addressing the large, broad range of human activities via semantic attributes, allowing sharing knowledge of movements rather than considering mutually exclusive activity classes; it addresses not only parameter based but also feature representation transfer learning; it extends Transfer Learning beyond the formulated solutions across the same domain, i.e., measurements from similar sensors from OBDs on specific locations of the human body; it addresses cases under different sensor setups, i.e., number and type of sensors, sampling rate to an extent, preprocessing, and target activities; it gives an alternative to address M-HAR from the definitions and semantics of activities; it exploits sources from physically related quantities of OBDs for transferability to M-HAR. In general, this Adaptable Attr-IMU-tCNN uses a known M-HAR system adapting its annotated data material and its pre-trained M-HAR model.

The Attr-based Transfer Learning for M-HAR also answers an open question formulated in [Rei21] regarding the problem of performing M-HAR for different logistics scenarios. The Adaptable Attr-IMU-tCNN trained with LARa can be directly used to a certain extent even when the different subjects, number of OBDs and their locations on the human body,

and recording settings vary in the new logistic scenario; shown with the real dataset provided by *MotionMiners GmbH*.

This thesis empirically shows a practical application of the attributes and the Attr-IMU-tCNN pretrained on LARa for improving the annotation process of multi-channel Time-Series data in the context of M-HAR. It proposes a semi-automated annotation method, exploiting the usage of attributes. This method is called Human Activity Retrieval (HARetr). It uses the Attr-IMU-tCNN pretrained on LARa to speed up the annotation process. HARetr ranks segmented windows of a recording according to its similitude in attribute space to a query. As a result, annotation time reduces by half without affecting the between-annotator consistency of the activity classes and attributes remains substantial.

Future work

The proposed method does not model temporal or causal relations between segmented sequences. Each multi-channel Time-Series segment is handled as an independent sample. Combining overlapping predictions of window-wise predictions via majority vote considers context to a certain extent, as used in this thesis. However, past predictions from a long sequence of continuous samples do not influence current predictions, which is a drawback for structured activity sequences—possible activity predictions strongly depend on previous activities when activities have precondition-effect relations, e.g., an intralogistics process.

RNNs, transformers, and conformers are special for modelling such temporal and causal relations. RNNs are used in combination with tCNN layers for feature extraction for M-HAR, as shortly evaluated in this thesis and [HHP16, OR16]. Similarly, transformers and conformers rely on the feature extractors from CNNs, and tCNNs [SK21, DLKP22, KCKL22]. Preliminary experiments on the LARa dataset for classification using the transformer from [SK21] show no improvement compared with the architectures that this thesis deployed. Nevertheless, all these architectures carry on with the limitations of the CNNs regarding the causal relations of samples, because they replace the MLP part of the architecture and do not exploit causal relations and dynamics among segment samples. Thus, they are not necessarily efficient for learning temporal relations of not scripted and repetitive activities of short duration, especially as they are learnt purely from data. This limitation can make RNNs, transformers and conformers sample-inefficient, specifically for the relatively small M-HAR datasets, if no additional structural and causal information is not considered.

Future work can integrate temporal and causal information through prior knowledge. For example, in cooking, prior knowledge concerns a recipe and ingredients; in intralogis-

tics, prior knowledge concerns possible orders of activities through preconditions and effects of activities involving objects and other context factors, like locations of workers. Business Process Models (BPMs) from warehouse processes in the intralogistics provide such prior knowledge and the dynamics of states and activity. Context information can be integrated systematically into a DNN-based M-HAR system, as [LMRA+21] preliminary demonstrated with a hybrid architecture that combines an attribute-based tCNN with shallow classifier. The shallow classifier predicts activity classes from the estimated attributes and context information provided by the BPM and the currently executed process step. This system allows integrating additional context information directly without re-training the DNN. This M-HAR system is suggested to be combined with the off-the-shelf Attr-IMU-tCNN presented in this thesis for Transfer Learning for HAR, for a more realistic case where the process step is estimated from external sources, as well as on additional M-HAR datasets.

Symbolic HAR methods specify existing prior domain knowledge in symbolic formalisms to predict activities, e.g., Markov Logic Networks (a relational logic) [CSR+21], ProbLog (a probabilistic programming language) [SCS18], the Planning Domain Definition Language (PDDL), typically used for specifying classical planning problems [KNY+14], or multiset rewriting systems [LSB+18]. A first attempt to combine symbolic models of dynamic systems and neural networks has been made by the author of this thesis [MRLS+19], using a pre-trained tCNN as the observation model of a CSSM. The results show the general feasibility of this approach. As future work, the author of this thesis suggests to develop a hybrid model, using the Attr-IMU-tCNN as the observation model, and the system dynamics is represented by a symbolic model constructed from domain knowledge and from attribute representations. It is suggested to consider not only the attribute representation but also high-level, symbolic domain knowledge as an embedding for representing sequences. These embeddings with the Attr-based Transfer Learning for M-HAR proposed in this thesis could be used to transfer across more challenging scenarios, considering structural and causal relations of activities and a fully transferable architecture. Furthermore, tighter integration of symbolic knowledge into a neural network might be worth investigating, e.g., an end-to-end-trainable architecture for activity and context recognition.

Questions regarding Transfer Learning for HAR remain open—for example, transferability among different populations and cultures. Besides, the physical characteristics of humans might also be considered, for example, as part of an attribute representation or an embedding along with symbolic information. Finding sequential profiles of human characteristics from the sequences might be worth investigating. With these, an unbiased dataset could be created where subjects' intrinsic movement patterns are removed from the inertial profiles. Besides, a complete dataset with window-wise and sample-wise

annotations of processes, activities, and attributes of various durations, considering human poses, different recording sessions, OBD locations on the subjects, human physical characteristics, and global context, could be considered for future work. This dataset might be more suitable for transferability than, for example, LARa, and for facilitating research not just towards Human Activity Recognition, but inclusive of shareable and virtual data.

Appendices

APPENDIX

A.1 BACKPROPAGATION

A.1.1 *Example MLP*

For better describing the back propagation, one takes the MLP in the Figure 2.1.2, the TSSE (Equation 2.1.16) as the objective function $\mathbf{E}(\mathbf{w})$ and a training set $\mathbf{D}$. First, the MLP processes samples $\mathbf{X}$ computing a K-dimensional hidden vector $\mathbf{z}$, and a M-dimensional output vector $\hat{\mathbf{y}}$. The relations among the three vectors are described in Equation 2.1.5 and Equation 2.1.5. Second, the backpropagation implements the backward step; that is, it computes the gradients for each layer, and it updates the weights following the updating rule of GD, Equation 2.1.10. The weight-updates for the MLP are:

$$\mathbf{W}_{(2)i+1} = \mathbf{W}_{(2)i} - \gamma \frac{\delta \mathbf{E}(\mathbf{W}_1, \mathbf{W}_2)}{\delta \mathbf{W}_2}, \tag{A.1.1}$$

$$\mathbf{W}_{(1)i+1} = \mathbf{W}_{(1)i} - \gamma \frac{\delta \mathbf{E}(\mathbf{W}_1, \mathbf{W}_2)}{\delta \mathbf{W}_1}, \tag{A.1.2}$$

being $\mathbf{W}_{(1)} \in \mathbb{R}^{[N,K]}$ and $\mathbf{W}_{(2)} \in \mathbb{R}^{[K,M]}$.

Each of the weight-update rules has a gradient of the objective function with respect to each of the weights, in this case all the terms in the matrices $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$. Rewriting the objective function $E(\mathbf{w})$ (see Equation 2.1.16) in terms of $\mathbf{W}_{(1)}$ and $\mathbf{W}_{(2)}$, one has:

$$E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right) = \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \frac{1}{2} \sum_{m=1}^{M} \left(\mathbf{y}^{(n)} - \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(n)}\right)\right)^2 \tag{A.1.3}$$

$$E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right) = \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \frac{1}{2} \sum_{m=1}^{M} \left(\mathbf{y}^{(n)} - \boldsymbol{\varphi}\left(\mathbf{W}_{(2)}^{T} \boldsymbol{\varphi}\left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(n)}\right)\right)\right)^2 \tag{A.1.4}$$

Based on that, one can see that each of the final neurons adds its quadratic deviation to the objective function $E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)$. If we consider each of the quadratic deviations separately, the gradient of the objective function Equation A.1.3 with respect to each of the weights in the matrix $\mathbf{W}_{(2)}$ will be:

$$\frac{\partial E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)}{\partial \mathbf{W}_{(2)}} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right) \boldsymbol{\varphi}'\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(n)}\right) \mathbf{z}^{(n)} \tag{A.1.5}$$

And the gradient of the Equation A.1.4 with respect to $\mathbf{W}_{(1)}$ is:

$$\frac{\partial E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)}{\partial \mathbf{W}_{(1)}} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \sum_{m=1}^{M} \left[\left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right) \boldsymbol{\varphi}'\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(n)}\right) \mathbf{W}_{(2)}^{T} \boldsymbol{\varphi}'\left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(n)}\right) \mathbf{x}^{(n)}\right]_{m} \tag{A.1.6}$$

Rewriting the Equation A.1.6 to Equation A.1.7, one notice that the factors multiplying the vector $\mathbf{z}^{(n)}$ in the Equation A.1.5 are also present in Equation A.1.7. This factors are called local gradients or error signals of layer inputs with respect to layers outputs. For example, the local gradient of the output-layer is $\boldsymbol{\delta}_{y}$.

$$\frac{\partial E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)}{\partial \mathbf{W}_{(1)}} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \sum_{m=1}^{M} \left[\underbrace{\left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right) \boldsymbol{\varphi}'\left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(n)}\right)}_{\boldsymbol{\delta}_{y^{(n)}}} \mathbf{W}_{(2)}^{T} \boldsymbol{\varphi}'\left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(n)}\right) \mathbf{x}^{(n)}\right]_{m} \tag{A.1.7}$$

The equations (A.1.5 and A.1.7) are described with respect to local gradients:

$$\frac{\partial E\left(\mathbf{W}_{(1)}, \mathbf{W}_{(2)}\right)}{\partial \mathbf{W}_{(2)}} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \boldsymbol{\delta}_{y}^{(n)} \mathbf{z}^{(n)} \tag{A.1.8}$$

$$\frac{\partial E(\mathbf{W}_{(1)}, \mathbf{W}_{(2)})}{\partial \mathbf{W}_{(1)}} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \sum_{m=1}^{M} \left[\boldsymbol{\delta}_{y_{m}^{(n)}} \mathbf{W}_{(2)_{m}}\right] \boldsymbol{\varphi}'\left(\mathbf{W}_{(1)} \cdot \mathbf{x}^{(n)}\right) \mathbf{x}^{(n)} = - \sum_{(\mathbf{x},\mathbf{y})^{(n)}_{n}{}^{N} \in \mathbf{D}} \boldsymbol{\delta}_{\mathbf{z}^{(n)}} \mathbf{x}^{(n)}$$

(A.1.9)

with,

$$\delta_{\hat{\mathbf{y}}^{(n)}} = \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right) \boldsymbol{\varphi}' \left(\mathbf{W}_{(2)}^{T} \mathbf{z}^{(n)}\right) \tag{A.1.10}$$

$$\delta_{\mathbf{z}^{(n)}} = \sum_{m=1}^{M} \left[\delta_{y_m^{(n)}} \mathbf{W}_{(2)_m}^{T}\right] \boldsymbol{\varphi}' \left(\mathbf{W}_{(1)}^{T} \mathbf{x}^{(n)}\right) \tag{A.1.11}$$

The local gradients are computed by using the derivatives of the activation functions (see Equation 2.1.1). Since the derivatives of the activation functions are expressed in terms of their outputs (the neurons outputs), the local gradients are also expressed in terms of the neurons outputs. Thus, the order of the Backpropagation is relevant: First, the neurons outputs are computed, and then the gradients. In fact, the Backpropagation propagates backwards the local gradients or error signals—The gradient of a hidden layer depends on the gradient of the next layer.

A.1.2 *Example local gradients with Sigmoid*

If one uses the sigmoid as an activation function (Equation 2.1.6), one will have the same expression as Equation 2.1.21 for each of the neurons. By organizing the expressions in matrices, the local gradients will become [Roj96]:

$$\delta_{\hat{\mathbf{y}}^{(n)}} = \operatorname{diag}\left(\hat{\mathbf{y}}^{(n)}\right) \left[I_{[M,M]} - \operatorname{diag}\left(\hat{\mathbf{y}}^{(n)}\right)\right] \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right)^{T} \tag{A.1.12}$$

$$\delta_{\hat{\mathbf{z}}^{(n)}} = \operatorname{diag}\left(\mathbf{z}^{(n)}\right) \left[I_{[K,K]} - \operatorname{diag}\left(\mathbf{z}^{(n)}\right)\right] \mathbf{W}_{(2)} \delta_{\hat{\mathbf{y}}^{(n)}} \tag{A.1.13}$$

with,

$$\operatorname{diag}(\mathbf{a}) = \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_N \end{bmatrix} \tag{A.1.14}$$

And the weight-updates

$$W_{(2)_{i+1}} = W_{(2)_i} - \gamma \delta_{\hat{\mathbf{y}}^{(n)}} \mathbf{z}^{(n)} \tag{A.1.15}$$

$$W_{(1)_{i+1}} = W_{(1)_i} - \gamma \delta_{\mathbf{z}^{(n)}} \mathbf{x}^{(n)} \tag{A.1.16}$$

A.2 LINEAR AND ANGULAR VELOCITY AND ACCELERATION OF A POINT

A frame is a coordinate system attached to a body or point in space and describes a coordinate system relative to another. It contains the orientation and position information of the body or point in space. The orientation of the unit vectors defining the principal axes of the coordinate system is given by a $[3 \times 3]$ rotation matrix ${}^{\{O\}}_{\{J\}}\mathbf{R} \in \mathbb{R}$, and its position with respect to an origin frame is given by a vector $\mathbf{p}_{B\,ORG} \in \mathbb{R}$ [Crao5]. Figure A.2.1 presents three frames {U}, {A}, {B} and {C}, where {A} and {A} are described relative to frame {A} or origin, and {A} is given relative to frame {A}.

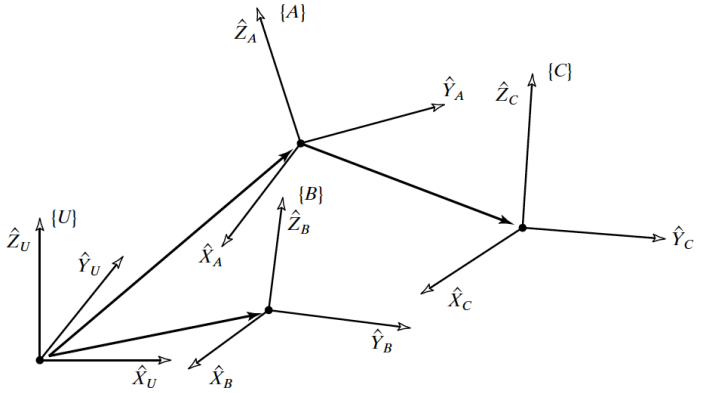


Figure A.2.1: Example of three frames, {U}, {A}, {B} and {C}. Image taken from [Crao5].

For a point Q, as Figure A.2.2a shows, the velocity of a point Q with respect to the frame {B} is given by

$${}^{B}v_Q = \frac{d}{dt}{}^{B}\mathbf{q} = \lim_{\Delta t \to 0} \frac{{}^{B}\mathbf{q}(t + \Delta t) - {}^{B}\mathbf{q}(t)}{\Delta t}. \tag{A.2.1}$$

The velocity vector ${}^{B}v_{Q}$ can be represented in terms of a reference frame $\{A\}$ by

$$ {}_{A}\mathbf{v}_{Q} = {}^{\{A\}}\mathbf{v}_{\{B\}\,ORG} + {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{B}v_{Q}, \tag{A.2.2} $$

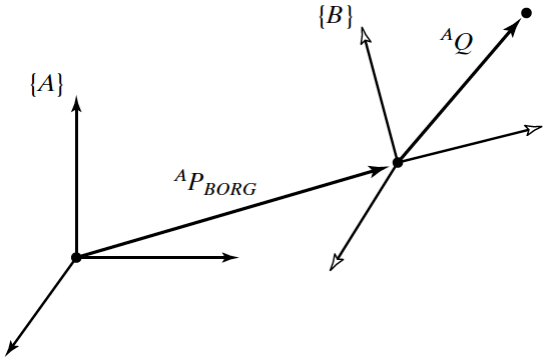
with ${}^{\{A\}}\mathbf{v}_{\{B\}\,ORG}$ being the velocity of the frame $\{B\}$.
The angular velocity is computed by derivating the rotation matrix $\mathbf{R}$, following

$$ \dot{\mathbf{R}} = \lim_{\Delta t \to 0} \frac{\mathbf{R}(t+\Delta t) - \mathbf{R}(t)}{\Delta t}. \tag{A.2.3} $$

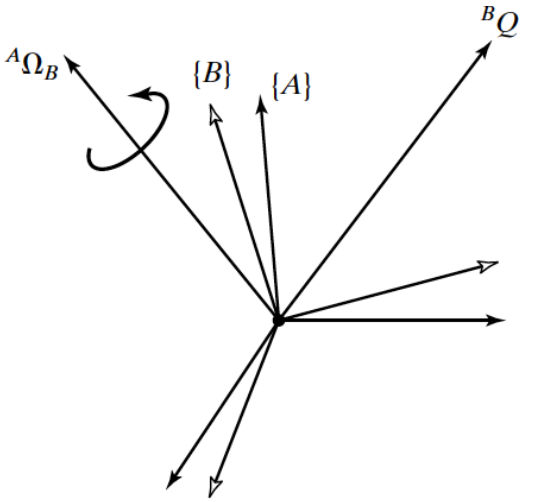
There linear velocity of the point Q with respect to $\{A\}$ is

$$ {}^{\{A\}}\mathbf{v}_{Q} = {}^{\{A\}}\mathbf{v}_{\{B\}\,ORG} + {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{v}_{Q} + {}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q}, \tag{A.2.4} $$

with ${}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{v}_{Q}$ the linear velocity of the point Q in frame $\{B\}$ with respect to $\{A\}$, ${}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q}$ being the linear velocity due to the angular velocity, and $\times$ the cross product.



(a) Point Q and frame $\{B\}$ with respect to $\{A\}$. Image taken from [Cra05].



(b) Point Q fixed in frame $\{B\}$ rotating with respect to $\{A\}$ with an angular velocity ${}^{A}_{B}\boldsymbol{\Omega}$. Image taken from [Cra05].

Derivating the linear velocity Equation A.2.4 with respect to time and considering not coincident frames A and B, the linear acceleration of the point Q with respect to {A} is

$$ {}^{\{A\}}\dot{\mathbf{v}}_Q = \frac{d}{dt}\left({}^{\{A\}}\mathbf{v}_{\{B\}\,ORG}\right) + \frac{d}{dt}\left({}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{v}_Q\right) + {}^{\{A\}}\dot{\boldsymbol{\Omega}}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q} + {}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times \frac{d}{dt}\left({}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q}\right), \tag{A.2.5} $$

obtaining,

$$ {}^{\{A\}}\dot{\mathbf{v}}_Q = {}^{\{A\}}\dot{\mathbf{v}}_{\{B\}\,ORG} + {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\dot{\mathbf{v}}_Q + 2\,{}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{v}_Q + {}^{\{A\}}\dot{\boldsymbol{\Omega}}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q} + {}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times \left({}^{\{A\}}\boldsymbol{\Omega}_{\{B\}} \times {}^{\{A\}}_{\{B\}}\mathbf{R}\,{}^{\{B\}}\mathbf{q}\right), \tag{A.2.6} $$

with ${}^{\{A\}}\dot{\mathbf{v}}_{\{B\}\,ORG}$, ${}^{\{A\}}\boldsymbol{\Omega}_{\{B\}}$ and ${}^{\{A\}}\dot{\boldsymbol{\Omega}}_{\{B\}}$ the linear velocity, angular velocity, and angular acceleration of {B} with respect to {A}.

A.3 TRAINING PROCEDURE

The network parameters are updated by minimizing the TCCE, Equation 2.1.18 when performing classification using the *Softmax* activation function, e.g., $TCNN^{softmax}_{fuse}$; or minimising the TBCE when predicting attributes using the *Sg* activation function, e.g., $TCNN^{attribute}_{fuse}$. Besides, the training approach uses the SBGD with the RMSProp update rule, Equation 2.1.15, as in [OR16]. A sliding window approach extracts sequence segments that are fed to the network. These segments are assigned the most frequent activity label[1]. We used the following hyperparameters: RMS decay of 0.95, base learning rates of $\gamma = [10^{-2}, 10^{-3}, 10^{-4}10^{-5}10^{-6}]$, and a batch size of B. Dropout is applied to all the fuse layers, e.g., FC layers, except to the classification layer. Networks are trained from scratch for a fixed number of epochs. Orthonormal initialization is utilized, following [MRGF+18]. A validation set is used for parameter search and early stopping.

As a preprocessing step, input sequences are normalized per channel to the range [0.1], following the Opportunity challenge [CSC+13b]. This normalization considers that the convolutional filters are shared among channels. Additionally, a Gaussian noise of zero-mean and of 0.01 standard deviation is added, following Algorithm 3.

1 This considers an annotation per sample.

SOBD from V-HAR Datasets

The joint poses sequences of $\mathcal{D}_{\mathtt{Source}}$ are either up-sampled or down-sampled to match the frequency of the $\mathcal{D}_{\mathtt{Target}}$. The J-HMDB, CAD-60, and NTU RGB+D are up-sampled by factors of $[4, 3, 2]$ for $\mathcal{D}_{\mathtt{Target}}$=Pamap2 as target, and by factors of $[1, 1, 0.5]$ for $\mathcal{D}_{\mathtt{Target}}$=Opp, respectively.

A sliding window approach with 1sec., W depending on the sampling rate, and a stride of Str is used for segmenting sequences of human poses; specifically, the following parameters were used: a window size of ($W = 25$) and a stride of $Str = 12$ for J-HMDB$_{pose}$, a window size of ($W = 30$) and a stride of $Str = 12$ for the CAD-60$_{pose}$, and a window size of ($W = 30$) and a stride of $Str = 3$ for the NTU$_{pose}$.

Pose channels and SOBDs are normalized to zero-mean and unit deviation. Following the pre-processing protocol from LARa-M Subsection 6.1.1, we normalize the joint-poses with respect to the torso for J-HMDB and CAD-60, and middle of the spine for NTU RGB+D.

A.4 RESULTS PER DATASET

A.4.1 *LARa*

Following Subsection 5.1.1, M-HAR is addressed on all the three subsets of the LARa dataset: LARa-M, LARa-Mb, and LARa-MM. LARa-M and LARa-Mb will be used as a source domain for transfer learning, i.e., from human pose to inertial measurements. LARa-M will be used as target domains for Transfer Learning for HAR. This takes advantage of the *TCNN* and *IMU-TCNN*, as they process sequences per channel with late fusion, the zero-mean and unit-deviation normalization of the channels, and the semantic attribute representation.

Considering that the target domains in Subsection 6.1.1 have different recording rates, two additional subsets are created from LARa-M, LARa-Mb and LARa-MM. LARa-M is sub-sampled from 200Hz to 100Hz and 30Hz, and LARa-M and LARa-MM is sub-sampled from 100Hz to 30Hz. Subsets will be denoted with the sampling rate.

A sliding-window approach is used for segmenting sequences with the following parameters: window size of $W = 200$ (1sec.) and step size of $Str = 25$ samples (87.5% overlapping) for LARa-M, $W = 100$ (1sec.) and step size of $Str = 12$ samples (88% overlapping) for LARa-M100, and $W = 24$(720ms) and step size of $Str = 12$ samples (48% LARa-M30—this window size is selected according to the Opp in Subsection 6.1.1. In general, the size of the input sequences $[W, H, C]$ is $[200, 126, 1]$ for LARa-M, $[100, 126, 1]$ for LARa-M100 and $[25, 126, 1]$ for LARa-M30. Similarly, the size of the input sequences is $[100, 30, 1]$ for LARa-Mb and $[25, 30, 1]$ for LARa-Mb30. Similarly, the size of the input sequences is $[100, 27, 1]$ for LARa-MM and $[25, 27, 1]$ for LARa-MM30. The Attr-IMU-tCNN and the networks in Subsection 6.3.1 are trained for each of the datasets for $B = 64$ epochs with a training batch of 200 samples.

A similar segmentation window approach to the LARa-SOBD is deployed. In addition, a sliding-window approach segments sequences with the following parameters: window size of $W = 200$ (1sec.) and step size of $Str = 25$ samples (87.5% overlapping) for LARa-SOBD, $W = 100$ (1sec.) and step size of $Str = 12$ samples (88% overlapping) for LARa-SOBD100, and $W = 24$(720ms) and step size of $Str = 12$ samples (48% LARa-SOBD30—this window size is selected according to the Opp in Subsection 6.1.1. In general, the size of the input sequences $[W, H, C]$ is $[200, 126, 1]$ for LARa-SOBD, $[100, 126, 1]$ for LARa-SOBD100 and $[25, 126, 1]$ for LARa-SOBD30.

Baseline

Table A.4.2 shows the classification performance, in terms of the wF1 [%], on the testing set from the LARa datasets: LARa-M, LARa-M30, LARa-Mb, LARa-Mb30 and LARa-MM. The number of channels S from the sets affects the M-HAR performance. LARa-Mb includes recordings from the legs compared to the LARa-MM. Activities directly related to walking are negatively affected. The less quantity of measurements also affects the *IMU-TCNN*$^{softmax}_{LSTM}$ and *IMU-TCNN*$^{softmax}_{FCN}$.

Subjects in the LARa dataset do not have instructions on how to perform the activities, so there is no script with a sequence of movements that the subjects carry to complete a task. The fuser layers LSTM and FCN present significantly lesser performance than the FC. LSTMs in M-HAR are good at learning distinctive temporal relations in the sequence, especially for cases with relatively ordered and repetitive movements in sequences. However, when movements are not following a particular order or script, i.e., movement variation is high, they cannot cope with this variation. The FCN computes a deep representation and produces a classification per sample; however, it does not observe the global representation of the sequence. The FCN depends on a large receptive field of the convolutional filters.

From Table A.4.3, the [1]SpectralPool and [1-2]SpectralPool reduce the number of parameters by 55.78% and 79.51% respectively. The spectral polling improves performance on the LARa-Mb. The spectral pooling does not affect the performance of the networks on the LARa-MM negatively.

Table A.4.1 shows the learnt attributes using the EA-*TCNN*$^{attribute}_{FC}$ on the LARa-MM. The a_{12} is shared among all the activities so that this attribute can be removed for final usage. The EA also controls the size of the representation. The activities *Standing* and *Walking* share more attributes than with respect to the other activities.

There are sample-wise attribute annotations of the LARa-MM. Table A.4.4 presents the performance of the *IMU-TCNN*$^{attribute}_{LSTM}$ and *IMU-TCNN*$^{attribute}_{FCN}$ on the LARa sets. The attribute representations improve classification on the OBD datasets, especially when using TPP fuse layers.

Table A.4.5 shows the performance of the LARa-SOBD and the set LARa-SOBD30. The TPP fuser affects negatively the performance.

Transfer Learning for LARa

For observing the effects of TL, three additional versions of the LARa-MM are created. These versions contain different proportions of the datasets. Specifically, LARa-MM25, LARa-MM50 and LARa-MM75 are created, respectively, with $\% = [20, 50, 75]$ of the subjects from the training set; respectively—the validation and testing sets remain equal.

Table A.4.1: Attribute representation $A \in \mathbb{B}^{19}$ of the LARa-MM dataset found using the EA-$TCNN_{\text{FC}}^{\text{attribute}}$.

Activity	Attributes																		
	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
Standing	1	1	0	0	1	1	1	1	1	1	0	0	1	0	1	0	1	1	1
Walking	1	1	0	0	1	0	1	1	0	1	1	1	1	1	0	0	1	1	1
Moving Cart	0	1	1	0	1	0	0	0	1	0	1	1	1	0	1	1	1	1	0
Handling Upwards	0	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1	0	1	1
Handling Centred	0	0	1	1	0	1	1	0	1	1	1	0	1	1	1	0	1	0	1
Handling Downwards	1	1	0	1	0	1	0	1	1	1	1	1	1	1	0	1	0	0	1
Synchronisation	1	0	0	0	1	1	0	0	0	1	0	1	1	0	0	1	1	0	1

Table A.4.6 presents the performances on the three versions of the LARa-MMs using pretrained architectures with LARa-M, LARa-MM and LARa-SOBD. In general, the performance improves according to the proportion of the dataset. Learnt features from human poses and their derivatives can be deployed on inertial data. This suggests that filters learnt short temporal-relations from the sequence inputs per channel independently of the rather related domain; this considering that time sequences of human poses and inertial measurements are physically related; that the number of channels differ; and that source and target are belonging to the same problem, i.e., activity classes.

Table A.4.7 shows the performance of the transfer learning on the LARa-Mb as the target scenario. In this case, the performance of the three networks $tCNN_{\text{JHMDB}}^{\text{LARa-Mb}}$, $tCNN_{\text{CAD60}}^{\text{LARa-Mb}}$, and $tCNN_{\text{NTU}}^{\text{LARa-Mb}}$ for both data sources, pose annotations and SOBD, and different transferable layers remains similar to the $tCNN^{\text{LARa-Mb}}$. The activities in the LARa-Mb dataset are performed in a warehouse environment, whereas the source datasets consider ADLs. The performance improves when considering the 75%, 50%, 30% or 10% of the LARa-Mb.

Table A.4.2: The best testing *w*F1 [%] from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the LARa-M, LARa-M30, LARa-Mb, LARa-Mb30, LARa-MM. The mean and SD from the *w*F1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same dataset and fuse layer.

	$TCNN_{fuse}^{softmax}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
LARa-M100	**75.80 ± 0.15**	74.92 ± 0.33	74.99 ± 0.20	68.88 ± 0.55	**69.45 ± 0.74**	68.75 ± 0.20	39.51 ± 0.27	**40.09 ± 0.62**	44.30 ± 0.69	**55.93 ± 0.62**	53.19 ± 1.62	52.77 ± 1.10
LARa-M30	**75.41 ± 0.35**	**75.68 ± 0.33**	**75.73 ± 0.22**	**74.13 ± 0.70**	**74.40 ± 0.55**	**74.11 ± 0.18**	37.94 ± 0.09	38.23 ± 0.11	38.09 ± 0.17	53.74 ± 0.76	54.34 ± 0.50	54.01 ± 0.43
LARa-Mb	73.80 ± 0.73	74.26 ± 0.68	**74.89 ± 0.48**	**74.86 ± 0.57**	**74.47 ± 0.44**	**74.80 ± 0.42**	42.46 ± 0.86	**45.22 ± 0.65**	44.78 ± 0.92	**61.38 ± 0.62**	**60.00 ± 0.91**	57.66 ± 1.83
LARa-Mb30	**74.59 ± 0.88**	**74.33 ± 0.71**	**74.42 ± 0.76**	**74.50 ± 0.98**	74.11 ± 0.20	**74.71 ± 0.62**	48.46 ± 0.12	**49.98 ± 0.68**	**49.95 ± 0.28**	65.13 ± 0.97	**66.29 ± 0.72**	65.76 ± 0.57
LARa-MM	**71.88 ± 0.34**	70.75 ± 0.50	**72.29 ± 0.67**	65.99 ± 0.90	**66.89 ± 1.12**	66.08 ± 2.09	41.71 ± 0.04	44.78 ± 1.56	**47.17 ± 1.00**	**62.53 ± 0.27**	56.72 ± 0.91	57.57 ± 3.06

	$IMU\text{-}TCNN_{fuse}^{softmax}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
LARa-M100	**76.27 ± 0.16**	**76.05 ± 0.22**	**76.13 ± 0.19**	**70.43 ± 1.03**	68.64 ± 0.69	69.23 ± 0.76	47.84 ± 1.43	**51.83 ± 1.60**	50.56 ± 1.41	**63.41 ± 0.24**	60.46 ± 1.23	60.10 ± 0.52
LARa-M30	**76.16 ± 0.14**	**76.50 ± 0.64**	**76.41 ± 0.25**	**75.09 ± 0.47**	**75.26 ± 0.71**	**75.15 ± 0.33**	44.33 ± 0.55	43.95 ± 1.09	**45.43 ± 1.47**	**58.71 ± 0.74**	**58.61 ± 0.90**	**58.79 ± 0.78**
LARa-Mb	74.59 ± 0.96	**75.27 ± 0.55**	**75.12 ± 0.37**	73.77 ± 0.58	74.34 ± 0.55	**74.93 ± 0.89**	57.51 ± 0.51	**58.74 ± 0.99**	**58.76 ± 0.37**	69.66 ± 1.05	70.70 ± 0.73	**71.29 ± 0.25**
LARa-Mb30	74.09 ± 0.30	**75.23 ± 0.50**	**75.36 ± 0.12**	**74.88 ± 0.43**	74.09 ± 0.69	**74.44 ± 0.87**	68.04 ± 0.23	**68.62 ± 0.38**	**68.84 ± 0.49**	**72.63 ± 0.49**	73.10 ± 0.29	**72.31 ± 0.26**
LARa-MM	71.21 ± 0.58	**71.98 ± 0.36**	**71.83 ± 0.31**	**66.84 ± 1.44**	65.03 ± 1.86	65.38 ± 1.26	51.82 ± 3.19	**55.68 ± 0.45**	**56.12 ± 0.47**	**68.89 ± 0.13**	68.11 ± 0.96	67.45 ± 0.64

Table A.4.3: The testing *w*F1 [%] computed from solving M-HAR using the *TCNN-[1-2]SpectralPool*$_{fuse}^{softmax}$ and the *IMU-TCNN-[1-2]SpectralPool*$_{fuse}^{softmax}$ on the OBD datasets. The mean and SD from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same pooling layer.

Dataset	$TCNN_{FC}^{softmax}$			$TCNN_{TPP}^{softmax}$			$IMU\text{-}TCNN_{FC}^{softmax}$			$IMU\text{-}TCNN_{TPP}^{softmax}$		
	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool
LARa-M100	**75.80 ± 0.15**	75.19 ± 0.42	75.15 ± 0.25	53.43 ± 2.07	**54.92 ± 0.69**	51.81 ± 2.95	**76.27 ± 0.16**	75.84 ± 0.37	**76.05 ± 0.24**	59.59 ± 1.06	**61.04 ± 0.63**	59.54 ± 0.36
LARa-Mb	73.80 ± 0.73	**74.81 ± 0.65**	**74.85 ± 0.77**	55.82 ± 1.36	**58.80 ± 1.22**	**57.28 ± 0.86**	74.59 ± 0.96	**75.45 ± 0.63**	**75.25 ± 0.36**	69.66 ± 1.05	**70.99 ± 0.85**	**70.99 ± 0.88**
LARa-MM	**71.88 ± 0.34**	**71.97 ± 0.30**	**71.68 ± 0.38**	**60.61 ± 0.88**	56.70 ± 0.94	56.15 ± 1.15	**71.21 ± 0.58**	**71.73 ± 0.18**	71.13 ± 0.68	**67.23 ± 0.71**	**67.54 ± 0.84**	66.85 ± 0.44

Table A.4.4: The wF1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the LARa-MM datasets. The mean and SD from the wF1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to classifier.

Dataset	$TCNN^{softmax}_{fuse}$		$TCNN^{attribute}_{fuse}$		$IMU\text{-}TCNN^{softmax}_{fuse}$		$IMU\text{-}TCNN^{attribute}_{fuse}$	
	FC	TPP	FC	TPP	FC	TPP	FC	TPP
LARa-M100	**75.80 ± 0.15**	53.43 ± 2.07	74.13 ± 0.90	53.41 ± 0.34	**76.27 ± 0.16**	59.59 ± 1.06	75.70 ± 0.39	61.00 ± 0.07
LARa-M30	**75.41 ± 0.35**	53.74 ± 0.76	**75.03 ± 0.24**	53.73 ± 0.27	**76.16 ± 0.14**	58.71 ± 0.74	**76.06 ± 0.22**	61.22 ± 0.04
LARa-Mb	73.80 ± 0.73	55.82 ± 1.36	**75.49 ± 0.40**	60.75 ± 0.73	74.59 ± 0.96	69.66 ± 1.05	**76.11 ± 0.15**	71.67 ± 0.27
LARa-Mb30	74.59 ± 0.88	65.13 ± 0.97	**75.66 ± 0.53**	65.45 ± 0.72	74.09 ± 0.30	72.63 ± 0.49	**75.22 ± 0.22**	72.83 ± 0.13
LARa-MM	**71.88 ± 0.34**	60.61 ± 0.88	**72.39 ± 1.01**	63.16 ± 0.55	71.21 ± 0.58	67.23 ± 0.71	**73.04 ± 0.97**	68.80 ± 0.55

Table A.4.5: The wF1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the LARa-SOBD. The mean and SD from the wF1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same fuse layer.

Dataset	*TCNN*						*IMU-TCNN*					
	FC			TPP			FC			TPP		
	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
LARa-SOBD100	**63.87 ± 0.51**	**64.32 ± 0.48**	**64.05 ± 0.11**	45.88 ± 1.43	47.33 ± 1.23	**47.96 ± 0.56**	**64.42 ± 0.42**	**64.89 ± 0.23**	**64.90 ± 0.33**	54.81 ± 0.39	**55.17 ± 1.42**	**55.47 ± 1.32**
LARa-SOBD30	**62.26 ± 0.14**	61.43 ± 1.20	**62.69 ± 0.34**	42.99 ± 0.51	**45.58 ± 0.46**	**45.94 ± 0.93**	56.20 ± 0.65	**57.76 ± 0.53**	**57.44 ± 0.24**	47.08 ± 0.26	**48.84 ± 0.10**	**48.61 ± 0.55**

Table A.4.6: Mean wF1 [%] of the $TCNN^{softmax}_{fuse}\uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{softmax}_{fuse}\uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $c_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{Source}$. Subsequently, the N_{conv} corresponding to highest wF1 [%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		$TCNN^{softmax}_{FC}\uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$							$IMU\text{-}TCNN^{softmax}_{FC}\uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$					
		Baseline	LARa-M		LARa-SOBD		LARa-Mb		Baseline	LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
N_{conv} Tr. Layers	c_1	71.88 ± 0.34	71.67 ± 0.63	62.75 ± 0.55	63.83 ± 2.57	64.32 ± 0.26	**73.22 ± 0.53**	64.45 ± 0.69	71.21 ± 0.58	**73.09 ± 0.46**	68.85 ± 0.80	63.59 ± 0.87	70.04 ± 0.54	72.11 ± 1.41	69.53 ± 0.20
	$c_{1,2}$		72.02 ± 0.58	64.12 ± 0.31	65.12 ± 2.46	64.14 ± 0.64	72.52 ± 1.13	64.59 ± 0.93		72.34 ± 0.85	69.19 ± 0.54	63.25 ± 0.31	69.16 ± 0.29	72.45 ± 1.19	67.29 ± 0.53
	c_{1-3}		72.29 ± 0.53	64.47 ± 0.78	65.00 ± 1.63	64.43 ± 0.63	**73.26 ± 0.41**	64.02 ± 0.33		72.42 ± 1.06	69.04 ± 0.55	63.85 ± 1.59	69.21 ± 0.41	72.54 ± 1.05	67.37 ± 0.57
	c_{1-4}		**72.80 ± 0.85**	62.88 ± 0.65	67.08 ± 1.71	64.88 ± 0.52	72.60 ± 0.63	63.50 ± 0.50		72.11 ± 0.94	67.84 ± 0.58	63.28 ± 0.50	69.23 ± 1.07	71.09 ± 1.34	68.60 ± 0.65
	c_{1-4}		71.15 ± 0.89	61.31 ± 0.83	63.74 ± 2.03	64.37 ± 0.17	71.34 ± 0.79	42.87 ± 0.06		64.52 ± 0.78	53.81 ± 0.31	63.85 ± 1.59	54.57 ± 0.11	71.26 ± 0.93	63.12 ± 0.27
$\%D_{tr}$	75	43.69 ± 1.32	69.64 ± 0.54	61.31 ± 0.83	**70.49 ± 0.65**	63.52 ± 0.39	69.31 ± 0.46	62.25 ± 0.44	44.36 ± 2.44	**70.99 ± 1.87**	51.08 ± 0.15	70.41 ± 0.75	68.76 ± 0.53	70.39 ± 1.18	69.12 ± 0.06
	50	42.11 ± 0.82	**68.42 ± 0.62**	58.87 ± 0.97	67.28 ± 1.61	59.26 ± 1.94	66.90 ± 1.05	58.93 ± 0.67	43.32 ± 1.33	**68.05 ± 0.29**	51.27 ± 0.25	62.93 ± 0.54	66.47 ± 0.48	66.91 ± 1.33	66.43 ± 0.40
	25	41.66 ± 2.36	**65.95 ± 2.24**	53.19 ± 1.23	**65.97 ± 1.61**	52.70 ± 0.93	61.21 ± 1.08	53.39 ± 0.67	42.54 ± 1.06	59.78 ± 1.96	49.28 ± 0.21	61.07 ± 0.49	58.12 ± 1.76	**62.22 ± 1.78**	59.95 ± 1.68

Table A.4.7: Mean wF1[%] of the $TCNN^{\text{LARa-Mb}}_{\mathcal{D}_{\text{Source}}}$ and $IMU\text{-}TCNN^{\text{LARa-Mb}}_{\mathcal{D}_{\text{Source}}}$ using the joint poses and the synthetic data. The $N_{conv} = c_4$ corresponding to highest wF1[%] is fixed, and $[10, 30, 50, 75]$% of the $\mathcal{D}_{\text{Target}}$ are deployed for fine-tuning. SD wF1[%] lies around 0.01. The mean from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion.

Dataset		$TCNN^{\text{softmax}}_{\text{FC}} \uparrow^{\text{LARa-Mb}}_{\mathcal{D}_{\text{Source}}}$							$IMU\text{-}TCNN^{\text{softmax}}_{\text{FC}} \uparrow^{\text{LARa-Mb}}_{\mathcal{D}_{\text{Source}}}$						
		JHMDB		CAD60		NTU RGB+D		**Baseline**	JHMDB		CAD60		NTU RGB+D		**Baseline**
		SOBD	**Pose**	SOBD	**Pose**	**Synth**	**Pose**	**%wF1**	SOBD	**Pose**	SOBD	**Pose**	SOBD	**Pose**	**%wF1**
%$D_{tr.}$	80	**65.26**	60.93	62.90	62.90	62.61	62.95	63.83	62.93	63.92	63.67	57.25	62.08	63.54	64.86
	60	**59.37**	54.74	51.70	52.73	53.42	**57.67**	53.14	60.09	59.99	55.49	54.70	59.95	59.95	59.37
	20	53.80	50.09	45.75	52.20	51.12	50.84	51.19	56.10	57.79	46.12	51.10	53.53	54.96	53.40
	10	47.41	46.81	**51.36**	48.95	**51.57**	43.32	47.06	51.67	48.38	50.21	47.64	49.62	49.86	44.41

A.4.2 *Opportunity Locomotion*

Table A.4.8: The two scripted ADL runs from the Opp.

ADL-Run	Drill-Run
Start	Open and Close the fridge
Groom	Open and Close the dishwasher
Relax	Open and Close 3 drawers (at different heights)
Prepare Coffee	Open and Close door 1
Drink Coffee	Open and Close door 2
Prepare sandwich	Turn on and off the lights
Eat sandwich	Clean table
Cleanup	Drink (standing)
Break	Drink (sitting)

Baseline

Table A.4.13 presents the baseline of the Opp_{loc} and Opp_{ges}. The sensor measurements contained in each human limb LA, LL, RA, RL, and NT are set to the five corresponding branches of the Attr-IMU-tCNN and *IMU-TCNN*; Figure 6.1.2b shows the sensor correspondences. Contrary to [RSA15, OR16], Table A.4.13 shows that using max-pooling layers does not negatively influence the networks' performances, even for short sequence inputs.

Table A.4.9 and Table A.4.10 show the learned attributes using the EA-Attr-IMU-tCNN$^{\mathcal{D}_{Target}}_{LARa-M}$ and the attribute representation of the LARa-M on the Opp_{loc} and Opp_{ges} respectively. The LARa attributes represent the activities from Opp. Even though the Opp does not define the activities and their particularities of the recordings, the EA-Attr-IMU-tCNN finds attributes that are semantically related to the activities. For example, the *Walk* activity is represented by *Gait Cycle*, *Step*, *Upwards*, *Downwards*, *Torso Rotation*, *Left Hand*, *Tool*, *Cart*, *None*. The *Gait Cycle* and *Step* describe when the subject carries out a displacement with more than three steps and step small movements with the feet. Table A.4.14 shows the performance when using the learnt attribute representation from EA-*TCNN*$^{attribute}_{FC}$.

Transfer Learning for Opportunity

LOCOMOTION Table A.4.15 shows the performance of the architecture on the Opp_{loc}. The architectures are trained from scratch, or pretrained on LARa-M30, LARa-Mb30

Table A.4.9: Attribute representation $A \in \mathbb{B}^{19}$ of the Opp_{loc} dataset found using the EA-Attr-IMU-tCNN$_{LARa-M}$.

Activity	Attributes																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
Null	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Stand	0	1	1	0	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0
Walk	1	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0	0	1	0
Sit	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1
Lie	1	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1	1	1	1

and LARa-SOBD30. Table A.4.15 presents also the performance on the Opp_{loc} under different proportions of the training set, $[10, 20, 40, 60, 80]\%$. The architectures' performances improve when pretraining with LARa datasets. Interestingly, the *IMU-TCNN*$^{attribute}_{fuse} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$ presents a poorer performance compared to *TCNN*$^{attribute}_{fuse} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$. This outcome might be explained by combining the larger amount of architecture parameters with the shorter segmented windows, the subsampling of the source domains, and the re-organization of the measurements per limb following the *IMU-TCNN* with five convolutional blocks.

Table A.4.11 presents the precision and recall of the architectures on the Opp_{loc} per activity class. In general, the performance of the activity classes improves when using the *TCNN*$_{LARa-M30}$ with respect to the *TCNN* that is trained from scratch. These activity classes are similar to LARa dataset, except the *Sit* class. The *Lie* improves, although this activity is far from being present in the intralogistics where LARa is recorded. This outcome contrasts with the class *Lie* results on the Pamap2.

Table A.4.16 shows the classification performance of the transfer learning on the Opp_{loc} dataset. The *TCNN*$^{Opp_{loc}}$ is considered as baseline. Only the *TCNN*$^{softmax}_{FC} \uparrow^{Opp_{loc}}_{J-HMDB_{Pose}}$ using pose annotations and *TCNN*$^{softmax}_{FC} \uparrow^{Opp_{loc}}_{NTU_{Pose}}$ improve the performance. In the case of having $[10, 30, 50]\%$ of the $\mathcal{D}_{Target}$, the pre-trained networks on J-HMDB and CAD60 significantly improved the performance.

Table A.4.10: Attribute representation $A \in \mathbb{B}^{19}$ of the Opp_{ges} dataset found using the EA-Attr-IMU-tCNN$_{LARa-M}$.

Activity	Attributes																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
Null	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	0	0	1
Open Door 1	0	1	0	0	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0
Open Door 2	1	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1
Close Door 1	1	0	0	1	0	1	1	0	0	0	0	1	0	1	0	1	1	0	1
Close Door 2	0	0	1	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0
Open Fridge	1	0	1	1	0	1	0	1	1	1	0	1	1	1	1	0	0	0	1
Close Fridge	1	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1
Open Dishwasher	1	0	1	1	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1
Close Dishwasher	1	0	1	1	0	0	1	0	1	1	1	0	1	0	0	1	1	1	1
Open Drawer 1	1	0	0	1	0	0	1	0	1	1	0	1	1	0	0	0	1	1	1
Close Drawer 1	0	1	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
Open Drawer 2	1	1	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Close Drawer 2	0	0	0	0	1	1	0	0	1	1	0	0	0	0	1	1	0	0	0
Open Drawer 3	0	0	1	1	0	0	0	1	1	1	1	0	1	0	0	1	1	1	0
Close Drawer 3	1	1	0	1	0	1	0	1	0	0	1	1	1	1	0	1	0	1	1
Clean Table	0	1	1	0	1	1	0	1	1	1	1	0	0	1	0	0	1	0	0
Drink from Cup	0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	1	1	0	0
Toggle Switch	0	1	0	1	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1

Table A.4.11: Precision and Recall [%] per class activity for the Opp_{loc} dataset using the *TCNN*, trained from scratch and using the LARa-M30 as data source. Numbers in bold are selected according to harmonic mean, see Section 6.2.

Activity	*TCNN*		$TCNN_{FC}^{attribute} \uparrow_{LARa-M30}^{Opp_{loc}}$	
	Precision	Recall	Precision	Recall
None	88.89 ± 2.7	55.53 ± 6.7	$\mathbf{89.40 \pm 2.3}$	$\mathbf{72.48 \pm 2.6}$
Stand	73.73 ± 2.2	91.94 ± 2.8	$\mathbf{81.08 \pm 2.5}$	$\mathbf{91.54 \pm 2.9}$
Walk	75.59 ± 1.0	74.99 ± 1.8	$\mathbf{81.27 \pm 1.9}$	$\mathbf{78.77 \pm 3.5}$
Sit	95.98 ± 2.1	95.42 ± 0.4	$\mathbf{96.21 \pm 0.9}$	$\mathbf{97.35 \pm 0.3}$
Lie	94.63 ± 0.3	98.92 ± 0.6	$\mathbf{94.73 \pm 0.3}$	$\mathbf{99.71 \pm 0.3}$

GESTURES Table A.4.17 shows the performance of the architecture on the Opp_{ges}. Besides, it also presents the performance on the two additional versions of gestures under different proportions of the training set, $[10, 20, 40, 60, 80]\%$.

The *TCNN* performance improves when pretraining with LARa-M30. However, the performances of the *IMU-TCNN* are poorer comparing the *TCNN*.

Table A.4.12 presents the precision and recall of the architectures on the Opp_{ges} dataset per activity class. The Opp_{ges} dataset mostly includes manual activity classes. These activities are strongly related to the activities in the intralogistics from LARa, e.g., order-picking activities. The performance of most of the activities in Opp_{ges} improves with respect to a network that is trained from scratch.

Table A.4.12: Precision and Recall [%] per class activity for the and Opp_{ges} dataset using the *TCNN*, trained from scratch and using the LARa-M30 as data source. DW stays for Dishwasher. Numbers in bold are selected according to harmonic mean between the precision and recall.

Activity	*TCNN*		$TCNN_{\text{FC}}^{\text{attribute}} \uparrow_{\text{LARa-Mb30}}^{\text{Opp}_{loc}}$	
	Precision	Recall	Precision	Recall
None	92.94 ± 0.3	96.2 ± 0.7	$\mathbf{93.67 \pm 0.3}$	$\mathbf{95.81 \pm 0.7}$
Open Door 1	60.65 ± 5.0	56.51 ± 3.1	$\mathbf{63.90 \pm 6.3}$	$\mathbf{56.51 \pm 0.4}$
Open Door 2	79.75 ± 4.2	53.24 ± 11.6	$\mathbf{80.74 \pm 5.7}$	$\mathbf{67.43 \pm 6.0}$
Close Door 1	$\mathbf{64.87 \pm 9.1}$	$\mathbf{69.69 \pm 3.9}$	61.20 ± 5.1	68.98 ± 1.0
Close Door 2	66.05 ± 6.3	89.75 ± 5.9	$\mathbf{73.68 \pm 3.0}$	$\mathbf{91.00 \pm 3.0}$
Open Fridge	80.68 ± 34.0	47.31 ± 5.5	$\mathbf{83.86 \pm 1.6}$	$\mathbf{47.48 \pm 2.8}$
Close Fridge	72.07 ± 1.2	68.17 ± 6.4	$\mathbf{73.80 \pm 5.8}$	$\mathbf{74.79 \pm 5.3}$
Open DW	$\mathbf{41.05 \pm 2.9}$	$\mathbf{48.09 \pm 6.9}$	37.42 ± 3.1	51.10 ± 0.4
Close DW	28.51 ± 2.6	49.08 ± 8.8	$\mathbf{41.86 \pm 8.1}$	$\mathbf{44.47 \pm 2.1}$
Open Drawer 1	$\mathbf{55.48 \pm 3.9}$	$\mathbf{35.60 \pm 5.3}$	50.77 ± 9.0	34.68 ± 0.9
Close Drawer 1	57.14 ± 4.6	19.95 ± 4.6	$\mathbf{48.04 \pm 0.7}$	$\mathbf{34.17 \pm 17.2}$
Open Drawer 2	$\mathbf{60.09 \pm 12.1}$	$\mathbf{23.89 \pm 6.5}$	59.69 ± 15.0	20.21 ± 2.1
Close Drawer 2	56.77 ± 16.2	25.58 ± 10.7	$\mathbf{58.56 \pm 10.2}$	$\mathbf{33.97 \pm 3.6}$
Open Drawer 3	$\mathbf{57.32 \pm 8.7}$	$\mathbf{55.89 \pm 13.1}$	46.21 ± 2.1	60.03 ± 17.2
Close Drawer 3	47.81 ± 11.1	62.22 ± 9.9	$\mathbf{49.26 \pm 8.3}$	$\mathbf{61.81 \pm 11.6}$
Clean Table	94.58 ± 2.1	39.37 ± 2.6	$\mathbf{88.61 \pm 1.3}$	$\mathbf{45.64 \pm 0.2}$
Drink f. Cup	67.69 ± 12.8	51.40 ± 2.8	$\mathbf{60.31 \pm 4.8}$	$\mathbf{59.14 \pm 1.2}$
Toggle Switch	78.59 ± 4.4	22.40 ± 0.6	$\mathbf{77.46 \pm 1.7}$	$\mathbf{30.38 \pm 1.2}$

Table A.4.18 presents the performance of the three pre-trained architectures on the Opp_{ges}. In comparison with the Opp_{loc}, pre-training the *TCNN* with the three source datasets with both the pose annotations or the SOBDs influences the performance on the $\mathcal{D}_{\text{Source}}$ significantly, especially when transferring the first convolutional layer. The performance in all scenarios improves significantly when considering $[30, 50]\%$ of the training material from the $\mathcal{D}_{\text{Target}}$ and pre-training with the three $\mathcal{D}_{\text{Source}}$ using pose annotations or the SOBDs.

Table A.4.13: The best validation *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the Opp. The mean and SD from the *w*F1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same fuse layer.

	$TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
Opp_{ges}	**89.05 ± 0.29**	**89.23 ± 0.31**	**89.02 ± 0.36**	**89.14 ± 0.33**	**89.09 ± 0.29**	**89.06 ± 0.24**	**75.90 ± 0.09**	**76.02 ± 0.04**	**76.04 ± 0.03**	76.35 ± 0.45	**76.72 ± 1.73**	**77.09 ± 2.53**
Opp_{loc}	85.77 ± 1.11	**86.83 ± 0.18**	85.34 ± 0.75	86.52 ± 0.24	85.41 ± 0.49	86.10 ± 0.89	53.24 ± 0.85	63.82 ± 1.86	65.69 ± 1.78	67.70 ± 1.87	10.84 ± 7.67	31.09 ± 13.39
	$IMU\text{-}TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
Opp_{ges}	89.82 ± 0.23	89.84 ± 0.10	89.94 ± 0.20	88.48 ± 0.19	88.55 ± 0.36	88.78 ± 0.32	88.48 ± 0.19	78.56 ± 0.13	78.68 ± 0.29	86.49 ± 0.04	86.43 ± 0.13	86.28 ± 0.25
Opp_{loc}	86.02 ± 0.36	85.34 ± 0.45	86.15 ± 0.22	84.28 ± 0.52	84.17 ± 1.31	84.23 ± 1.44	66.34 ± 0.99	71.44 ± 0.41	71.25 ± 1.97	76.91 ± 0.60	71.77 ± 4.54	73.60 ± 2.39

Table A.4.14: The *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the OBD datasets. The mean and SD from the *w*F1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to classifier.

	$TCNN^{attribute}_{fuse}$				$IMU\text{-}TCNN^{attribute}_{fuse}$			
Dataset	FC	LSTM	FCN	TPP	FC	LSTM	FCN	TPP
Opp_{ges}	86.49 ± 0.85	88.14 ± 0.29	75.84 ± 0.02	76.66 ± 0.27	88.22 ± 0.30	88.00 ± 0.60	75.83 ± 0.01	84.58 ± 0.38
Opp_{loc}	85.84 ± 1.15	80.69 ± 0.90	43.77 ± 2.69	66.18 ± 1.23	83.54 ± 0.68	81.83 ± 0.46	39.15 ± 2.50	74.30 ± 0.94

Table A.4.15: Mean wF1[%] of the $TCNN^{softmax}_{FC} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $c_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{Source}$. Subsequently, the N_{conv} corresponding to highest wF1[%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		Baseline	$TCNN^{softmax}_{FC} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$						Baseline	$IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$					
			LARa-M		LARa-SOBD		LARa-Mb			LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
N_{conv} Tr. Layers	c_1	85.77 ± 1.11	88.56 ± 0.22	74.00 ± 1.24	88.19 ± 0.21	69.91 ± 1.89	88.31 ± 0.30	75.75 ± 1.09	86.02 ± 0.36	87.14 ± 0.38	82.61 ± 0.62	86.88 ± 0.09	82.03 ± 0.83	87.63 ± 0.08	82.68 ± 0.58
	$c_{1,2}$		88.45 ± 0.08	75.03 ± 0.46	88.41 ± 0.29	75.56 ± 0.06	88.34 ± 0.26	74.97 ± 0.73		87.76 ± 0.21	83.26 ± 0.85	87.66 ± 0.23	83.02 ± 0.47	87.59 ± 0.39	82.59 ± 0.40
	c_{1-3}		88.21 ± 0.28	71.82 ± 2.65	88.88 ± 0.08	75.16 ± 1.41	87.92 ± 0.31	75.47 ± 0.38		87.87 ± 0.46	82.89 ± 0.43	87.53 ± 0.41	82.78 ± 0.78	87.55 ± 0.20	83.15 ± 0.56
	c_{1-4}		88.73 ± 0.35	73.57 ± 2.66	88.24 ± 0.08	73.94 ± 1.27	88.19 ± 0.17	74.78 ± 1.01		88.17 ± 0.48	83.12 ± 0.64	87.69 ± 0.36	83.39 ± 0.76	87.21 ± 0.18	82.43 ± 0.28
	c_{1-4}		89.09 ± 0.04	40.47 ± 0.20	87.98 ± 0.19	33.85 ± 0.18	87.85 ± 0.15	44.80 ± 0.33		87.86 ± 0.11	69.82 ± 0.07	87.02 ± 0.33	74.97 ± 0.53	84.05 ± 0.43	70.20 ± 0.10
$\%D_{tr.}$	80	81.85 ± 1.29	81.62 ± 0.62	66.26 ± 1.48	82.33 ± 0.37	64.15 ± 8.37	81.32 ± 0.75	67.68 ± 1.11	70.37 ± 5.65	74.41 ± 2.13	76.11 ± 0.76	79.51 ± 0.22	77.58 ± 0.05	78.58 ± 1.20	75.61 ± 0.63
	60	75.66 ± 1.65	77.58 ± 1.18	58.63 ± 2.71	75.79 ± 0.52	66.28 ± 0.94	74.26 ± 1.49	65.31 ± 0.97	48.66 ± 23.28	71.18 ± 1.30	69.48 ± 0.86	72.28 ± 0.45	70.62 ± 0.46	73.18 ± 0.89	70.01 ± 1.19
	40	63.57 ± 20.23	72.85 ± 1.38	54.65 ± 2.64	73.36 ± 0.99	64.03 ± 0.90	73.56 ± 2.66	60.47 ± 2.17	13.26 ± 5.70	69.78 ± 1.31	64.80 ± 0.60	69.79 ± 1.03	68.16 ± 0.78	72.44 ± 1.05	67.39 ± 1.03
	20	63.85 ± 8.91	67.77 ± 0.87	48.62 ± 4.38	67.99 ± 1.74	40.76 ± 0.95	66.23 ± 4.44	48.36 ± 6.00	8.63 ± 3.24	60.87 ± 3.28	60.86 ± 1.26	68.54 ± 0.60	64.75 ± 0.42	66.54 ± 2.18	61.15 ± 0.97
	10	52.60 ± 0.1	56.40 ± 2.69	34.62 ± 4.45	55.77 ± 1.59	33.32 ± 0.53	39.93 ± 5.29	29.54 ± 2.04	21.36 ± 12.48	35.25 ± 3.67	54.71 ± 0.80	52.82 ± 1.30	52.06 ± 1.38	32.70 ± 3.60	44.39 ± 5.38

Table A.4.16: Mean wF1[%] of the $TCNN^{Opp_{loc}}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{Opp_{loc}}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The $N_{conv} = c_4$ corresponding to highest wF1[%] is fixed, and $[10, 20, 60, 80]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. SD wF1[%] lies around 0.01. The mean from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion.

Dataset		$TCNN^{attribute}_{fuse} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$							$IMU\text{-}TCNN^{attribute}_{fuse} \uparrow^{Opp_{loc}}_{\mathcal{D}_{Source}}$						
		JHMDB		CAD60		NTU RGB+D		*Baseline*	JHMDB		CAD60		NTU RGB+D		*Baseline*
		Synth	*Pose*	*Synth*	*Pose*	*Synth*	*Pose*	%$\mathbf{wF1}$	*Synth*	*Pose*	*Synth*	*Pose*	*Synth*	*Pose*	%$\mathbf{wF1}$
$\%D_{tr.}$	75	85.90	**87.33**	86.01	85.67	**86.67**	86.26	81.85	84.64	83.59	**85.83**	**85.89**	84.14	84.34	70.37
	60	85.18	**85.60**	**85.62**	**85.46**	**85.62**	85.03	75.66	**83.83**	**83.15**	**83.86**	**85.73**	**83.15**	**83.25**	48.66
	20	**84.23**	84.05	**84.32**	83.74	83.78	81.46	63.85	**81.87**	**82.71**	**82.02**	**83.09**	**81.52**	**81.80**	8.63
	10	**55.16**	**54.77**	**54.90**	53.12	**55.33**	53.61	52.60	44.09	47.74	48.98	48.43	44.00	46.14	21.36

Table A.4.17: Mean wF1 [%] of the *TCNN*$^{\text{softmax}}_{\text{FC}}$ $\uparrow^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$ and *IMU-TCNN*$^{\text{softmax}}_{\text{FC}}$ $\uparrow^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $\text{c}_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{\text{Source}}$. Subsequently, the N_{conv} corresponding to highest wF1 [%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{\text{Target}}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		*TCNN*$^{\text{softmax}}_{\text{FC}}$ $\uparrow^{\text{Opp}_{\text{loc}}}_{\mathcal{D}_{\text{Source}}}$							*IMU-TCNN*$^{\text{softmax}}_{\text{FC}}$ $\uparrow^{\text{Opp}_{\text{loc}}}_{\mathcal{D}_{\text{Source}}}$						
		Baseline	LARa-M		LARa-SOBD		LARa-Mb		Baseline	LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
Tr. Layers	c_1	89.05 ± 0.29	87.84 ± 0.67	78.02 ± 1.70	86.60 ± 0.33	77.10 ± 0.70	88.34 ± 0.40	79.05 ± 1.53	89.82 ± 0.23	89.58 ± 0.15	87.07 ± 0.16	89.02 ± 0.09	83.75 ± 0.27	90.10 ± 0.12	87.09 ± 0.25
	$\text{c}_{1,2}$		87.82 ± 0.61	87.96 ± 0.46	87.37 ± 0.37	77.31 ± 0.30	88.67 ± 0.35	79.28 ± 0.86		89.92 ± 0.16	87.30 ± 0.21	89.34 ± 0.21	84.13 ± 0.30	89.80 ± 0.51	86.83 ± 0.12
	c_{1-3}		88.85 ± 0.40	78.16 ± 2.10	87.94 ± 0.32	77.01 ± 0.59	88.08 ± 0.32	79.20 ± 2.17		90.26 ± 0.04	87.16 ± 0.30	89.89 ± 0.16	87.35 ± 0.13	90.28 ± 0.17	87.02 ± 0.08
	c_{1-4}		88.68 ± 0.55	80.69 ± 2.03	88.36 ± 0.23	80.71 ± 0.62	89.86 ± 0.48	80.68 ± 0.42		90.26 ± 0.28	86.98 ± 0.18	89.68 ± 0.37	83.36 ± 0.29	90.16 ± 0.20	87.17 ± 0.24
	c_{1-4}		88.03 ± 0.25	75.83 ± 0.00	86.00 ± 1.34	75.83 ± 0.00	88.57 ± 0.34	75.82 ± 0.01		90.16 ± 0.12	78.54 ± 0.09	90.23 ± 0.14	80.36 ± 0.02	87.66 ± 0.18	78.18 ± 0.14
%$\text{D}_{\text{tr.}}$	80	85.17 ± 0.87	87.20 ± 0.36	80.27 ± 0.73	87.37 ± 0.37	81.25 ± 0.27	87.27 ± 0.21	80.17 ± 0.23	86.67 ± 0.79	87.13 ± 0.26	85.09 ± 0.08	86.74 ± 0.77	82.35 ± 4.39	87.37 ± 0.17	85.35 ± 0.43
	60	82.56 ± 1.23	82.99 ± 0.34	77.98 ± 0.28	83.25 ± 0.49	78.60 ± 0.38	84.15 ± 0.17	77.95 ± 0.92	83.88 ± 0.63	83.90 ± 0.36	82.89 ± 0.31	84.07 ± 0.37	82.91 ± 0.04	84.35 ± 0.17	83.07 ± 0.45
	40	79.97 ± 0.53	80.66 ± 0.39	76.95 ± 1.01	79.72 ± 1.12	78.30 ± 0.56	81.85 ± 0.35	78.36 ± 0.38	79.82 ± 0.37	78.98 ± 0.13	80.47 ± 0.26	79.40 ± 0.52	80.87 ± 0.39	79.89 ± 0.45	81.06 ± 0.03
	20	80.52 ± 0.35	80.48 ± 0.70	76.09 ± 0.33	79.23 ± 0.11	76.93 ± 0.60	81.07 ± 0.09	76.36 ± 0.30	78.91 ± 0.39	78.61 ± 0.30	77.82 ± 0.11	78.82 ± 0.10	79.21 ± 0.30	80.10 ± 0.25	78.98 ± 0.11
	10	74.54 ± 1.45	75.50 ± 1.19	75.79 ± 0.03	77.11 ± 0.18	75.91 ± 0.11	76.51 ± 1.11	75.84 ± 0.05	75.83 ± 0.00	75.95 ± 0.56	75.87 ± 0.05	76.45 ± 0.38	75.83 ± 0.00	77.54 ± 0.77	76.29 ± 0.34

Table A.4.18: Mean wF1 [%] of the *TCNN*$^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$ and *IMU-TCNN*$^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$ using the joint poses and the synthetic data. The $\text{N}_{\text{conv}} = \text{c}_4$ corresponding to highest wF1 [%] is fixed, and $[10, 20, 60, 80]$% of the $\mathcal{D}_{\text{Target}}$ are deployed for fine-tuning. SD wF1 [%] lies around 0.01. The mean from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion.

Dataset		*TCNN*$^{\text{attribute}}_{\text{fuse}}$ $\uparrow^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$							*IMU-TCNN*$^{\text{attribute}}_{\text{fuse}}$ $\uparrow^{\text{Opp}_{\text{ges}}}_{\mathcal{D}_{\text{Source}}}$						
		JHMDB		CAD60		NTU RGB+D		**Baseline**	JHMDB		CAD60		NTU RGB+D		**Baseline**
		Synth	**Pose**	**Synth**	**Pose**	**Synth**	**Pose**	**%wF1**	**Synth**	**Pose**	**Synth**	**Pose**	**Synth**	**Pose**	**%wF1**
%$\text{D}_{\text{tr.}}$	80	87.33	87.41	**87.44**	87.12	**88.14**	**88.28**	85.17	87.23	86.83	87.58	86.22	87.23	87.45	86.67
	60	**87.06**	**87.04**	**87.01**	**86.77**	**86.96**	**86.77**	82.56	86.23	86.49	86.49	85.57	86.78	86.63	83.88
	20	**85.22**	**85.06**	**84.62**	**84.93**	84.80	84.60	80.52	84.39	82.67	84.37	83.05	84.29	83.74	78.91
	10	76.47	75.99	76.54	76.11	75.67	75.67	74.54	76.62	75.92	76.47	76.15	75.67	75.98	75.83

A.4.3 *Pamap2*

Baseline

Table A.4.19: Precision and Recall [%] per class activity for the Pamap2 dataset using the *TCNN*, trained from scratch and using the LARa-SOBD as data source. Numbers in bold are selected according to harmonic mean, see Section 6.2.

Activity	tCNN		$TCNN^{attribute}_{FC}$	$\uparrow^{Opp_{loc}}_{LARa-SOBD}$
	Precision	Recall	Precision	Recall
Rope Jump.	16.7 ± 28.9	8.33 ± 14.4	$\mathbf{33.33 \pm 57.7}$	$\mathbf{31.05 \pm 53.8}$
Lying	$\mathbf{99.5 \pm 0.6}$	$\mathbf{97.32 \pm 1.3}$	98.57 ± 1.3	98.09 ± 1.7
Sitting	94.5 ± 1.1	92.09 ± 2.5	$\mathbf{97.70 \pm 1.8}$	$\mathbf{93.30 \pm 3.1}$
Standing	70.8 ± 2.1	47.33 ± 22.5	$\mathbf{87.11 \pm 5.2}$	$\mathbf{58.86 \pm 35.8}$
Walking	96.8 ± 2.2	97.26 ± 1.1	$\mathbf{96.37 \pm 2.3}$	$\mathbf{98.49 \pm 0.8}$
Running	$\mathbf{100.0 \pm 0.0}$	$\mathbf{95.46 \pm 3.9}$	99.91 ± 0.2	94.77 ± 2.0
Cycling	98.1 ± 2.4	95.17 ± 4.5	$\mathbf{99.30 \pm 0.6}$	$\mathbf{95.44 \pm 1.9}$
Nordic Walk	96.7 ± 2.1	96.44 ± 1.7	$\mathbf{98.84 \pm 0.9}$	$\mathbf{98.09 \pm 0.4}$
Asc. Stairs	80.7 ± 15.2	77.70 ± 3.2	$\mathbf{92.03 \pm 2.4}$	$\mathbf{88.33 \pm 7.4}$
Desc. Stairs	82.6 ± 10.5	90.02 ± 4.0	$\mathbf{82.02 \pm 9.5}$	$\mathbf{91.92 \pm 2.1}$
VC*	90.9 ± 3.7	86.05 ± 7.2	$\mathbf{92.62 \pm 1.8}$	$\mathbf{93.81 \pm 2.3}$
Ironing	69.7 ± 1.9	89.55 ± 17.8	$\mathbf{80.24 \pm 14.7}$	$\mathbf{99.54 \pm 0.7}$

The sensor measurements contained in each human limb LA, LL, RA, RL, and NT are set to the five corresponding branches of the Attr-IMU-tCNN and *IMU-TCNN*; Figure 6.1.2b shows the sensor correspondences. Pamap2 contains recordings only from the dominant wrist and its corresponding side's ankle. Thus, the recordings of the dominant's side wrist and ankle recordings where used for both, the LA, RA and LL, RL branches respectively.

Table A.4.21 presents the testing performance of the baseline networks $TCNN^{softmax}_{fuse}$ and $IMU\text{-}TCNN^{softmax}_{fuse}$ for Pamap2 for the four fusion alternatives [FC, FCN, LSTM, TPP] and the max-pooling. The FCN fuser does not show promising results. The *IMU-TCNN* shows the better performance for all configurations.

Using max-Pooling does not affect negatively the performance; $[1-2]$MaxPool improves significantly the performance when $IMU\text{-}TCNN^{softmax}_{LSTM}$. Table A.4.22 shows the performance when using SpectralPooling. This pooling does not negatively affect the performance when using *TCNN*, even showing a slight increase. Table A.4.20 presents the semantic attributes learnt with EA-Attr-IMU-tCNN. The activities that are not shared with LARa set the attribute *Error*, corresponding to the assigned semantic meaning by the creators of the LARa. The locomotion activities contain the attribute *Gait Cycle* or *Step*,

and they do not contain *Standing Still*. Table A.4.23 shows the performance of usage of attributes on the Pamap2.

Table A.4.20: Attribute representation $A \in \mathbb{B}^{19}$ of the Pamap2 dataset found using the evolutionary algorithm and the $TCNN_{FC}^{attribute}$.

Activity	**Attributes**																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
rope jumping	0	0	0	0	0	1	1	1	1	1	1	1	0	1	0	1	1	1	0
lying	1	0	1	1	1	1	1	0	0	1	0	1	0	0	0	1	0	0	0
sitting	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	1
standing	0	0	0	1	1	1	0	0	1	0	0	1	1	0	1	0	1	1	0
walking	1	1	0	1	1	1	0	0	1	1	1	0	1	0	0	1	1	1	1
running	1	0	0	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1	1
cycling	1	0	0	0	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1
nordic watching	0	0	0	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	0
ascending stairs	0	1	0	0	1	1	1	1	1	0	1	1	0	0	1	1	1	1	1
descending stairs	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	1	0
vacuum cleaning	0	0	1	1	1	1	0	1	0	0	1	0	1	1	0	1	0	1	0
ironing	0	1	1	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0

Transfer Learning for Pamap2

Table A.4.24 shows the performance of the architecture on the Pamap2. The architectures are trained from scratch and using the LARa-SOBD, LARa-M, LARa-Mb. Table A.4.24 presents also the performance on the Pamap2 under different proportions of the training set, $[10, 20, 40, 60, 80]\%$. The validation and testing sets remain equal.

Pretraining both architectures using the different LARa datasets shows a positive effect on their performances. Transferring the learnt filters from the three datasets versions improves the performance of the architectures. Interestingly, pretraining the architectures and finetuning them using 50% of the Pamap2 training sets yields the best performances. Table A.4.19 presents the performance of the architectures on the Pamap2 per activity class. The performance per activity classes improves when transferring across different datasets, i.e., different amount of devices and activity annotations. Interestingly, the performance of activity class that is strange in intralogistic scenarios, lying did not improve on the test datasets. This implies that convolutional filters are rather generic, considering. Results

differ from [OMR16] as probably they considered longer extracted windows for learning longer temporal-relations. Those temporal relations are rather scenario-specific.

Table A.4.19 shows the performance of the networks $TCNN^{\mathcal{D}_{\text{Target}}}$ and $IMU\text{-}TCNN^{\mathcal{D}_{\text{Target}}}$ on the Pamap2 dataset as target. The $TCNN^{\text{softmax}}_{\text{FC}} \uparrow^{\text{Pamap2}}_{\mathcal{D}_{\text{Source}}} TCNN^{\text{Pamap2}}_{\text{D}_s}$ is pre-trained using the $\mathcal{D}_{\text{Source}}$: J-HMDB, CAD60, and NTU datasets. The performance on the Pamap2 dataset significantly increases when pre-training with the J-HMDB. Considering a limited amount of dataset for training, i.e., $[10, 30, 50, 75]\%$, the synthetic datasets from the three $\mathcal{D}_{\text{Source}}$ show a positive influence.

Table A.4.25 shows the precision and recall for each activity class of the Pamap2 and the shared ones in the J-HMDB datasets. The $TCNN^{\text{Pamap2}}_{\text{J-HMDB}}$ using the network with the c1 pretrained on the synthetic on-body devices of the J-HMDB and finetuned on the Pamap2. For comparing the performance of the networks, the harmonic mean (HM) of precision and recall are computed for each activity—the highest HMs are highlighted in bold. The activities *Climb Stairs*, *Sit*, *Run*, *Walk*, and *Stand* are common between the Pamap2 and J-HMDB. The performance of these shared activities increase. Activities that are only in Pamap2, but are semantically near to those in JHMDB, also showed a boost in performance, e.g., $\textit{Climb Stairs}_{\text{J-HMDB}}$-$\textit{Desc. Stairs}_{\text{Pamap2}}$, $\textit{Stand}_{\text{J-HMDB}}$-$\textit{Ironing}_{\text{Pamap2}}$, as person performs the activity standing and the activity is picking centred.

Table A.4.21: The best validation *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the Pamap2 dataset. The mean and SD from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same fuse layer.

	$TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
Pamap2	88.24 ± 0.97	88.43 ± 1.52	88.23 ± 0.99	86.81 ± 0.49	87.07 ± 0.54	86.91 ± 0.44	43.37 ± 3.94	41.82 ± 3.79	47.39 ± 4.63	70.93 ± 0.37	66.94 ± 1.86	67.57 ± 2.13
	$IMU\text{-}TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
Pamap2	88.86 ± 1.74	88.26 ± 1.24	88.25 ± 0.80	89.48 ± 1.50	89.48 ± 2.64	91.54 ± 2.20	65.30 ± 1.04	61.68 ± 1.57	61.17 ± 0.79	81.76 ± 0.21	79.10 ± 1.63	78.02 ± 2.44

Table A.4.22: The testing *w*F1 [%] computed from solving M-HAR using the $TCNN^{softmax}_{TPP}$ and the $IMU\text{-}TCNN^{softmax}_{TPP}$ on the Pamap2 datasets. The mean and SD from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same pooling layer.

Dataset	$TCNN^{softmax}_{FC}$			$TCNN^{softmax}_{TPP}$			$IMU\text{-}TCNN^{softmax}_{FC}$			$IMU\text{-}TCNN^{softmax}_{TPP}$		
	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool
Pamap2	88.24 ± 0.97	88.41 ± 1.35	88.77 ± 1.20	67.42 ± 1.48	70.32 ± 2.19	68.86 ± 1.04	88.86 ± 1.74	87.74 ± 0.47	87.65 ± 0.09	80.08 ± 1.53	79.37 ± 2.73	76.99 ± 1.68

Table A.4.23: The *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the Pamap2 datasets. The mean and SD from the *w*F1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to classifier.

Dataset	$TCNN^{attribute}_{fuse}$				$IMU\text{-}TCNN^{attribute}_{fuse}$			
	FC	LSTM	FCN	TPP	FC	LSTM	FCN	TPP
Pamap2	86.12 ± 0.81	86.21 ± 0.70	65.18 ± 1.70	65.18 ± 1.70	86.95 ± 0.50	88.82 ± 2.04	50.26 ± 1.71	77.60 ± 1.75

Table A.4.24: Mean wF1[%] of the $TCNN^{softmax}_{FC} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $c_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{Source}$. Subsequently, the N_{conv} corresponding to highest wF1[%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		Baseline	$TCNN^{softmax}_{FC} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$						Baseline	$IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$					
			LARa-M		LARa-SOBD		LARa-Mb			LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
Tr. Layers	c_1	88.24 ± 0.97	88.76 ± 0.89	67.44 ± 3.15	90.06 ± 0.61	62.98 ± 4.23	89.87 ± 1.58	70.02 ± 3.23	88.86 ± 1.74	88.60 ± 0.13	84.45 ± 1.06	87.91 ± 0.41	81.58 ± 1.18	88.84 ± 0.32	84.61 ± 1.13
	$c_{1,2}$		89.37 ± 1.78	70.87 ± 3.21	89.71 ± 0.93	65.90 ± 4.41	88.51 ± 0.58	71.31 ± 5.05		89.76 ± 2.54	83.55 ± 1.61	88.45 ± 0.17	85.54 ± 0.79	89.11 ± 0.20	80.53 ± 1.76
	c_{1-3}		89.99 ± 1.33	71.49 ± 4.33	88.57 ± 0.82	62.83 ± 5.71	88.63 ± 0.94	69.31 ± 2.99		89.13 ± 0.91	84.38 ± 1.68	59.33 ± 40.59	87.77 ± 1.69	88.87 ± 0.18	80.74 ± 1.46
	c_{1-4}		91.49 ± 0.28	74.87 ± 6.15	87.80 ± 0.84	77.61 ± 1.91	90.37 ± 0.86	73.98 ± 3.71		89.44 ± 0.78	86.35 ± 2.52	87.36 ± 0.72	88.88 ± 0.74	88.77 ± 0.26	83.65 ± 2.32
	c_{1-4}		90.81 ± 0.57	47.50 ± 0.89	88.50 ± 0.20	5.90 ± 0.13	92.21 ± 0.31	72.34 ± 0.38		88.91 ± 0.52	75.98 ± 0.45	87.79 ± 0.32	78.41 ± 0.32	88.05 ± 0.35	79.83 ± 1.60
$\%D_{tr.}$	80	80.18 ± 9.62	91.01 ± 0.35	59.27 ± 4.39	90.91 ± 1.17	77.61 ± 1.91	89.41 ± 0.68	67.00 ± 5.02	87.10 ± 0.82	90.31 ± 0.44	80.78 ± 0.62	87.75 ± 0.61	85.26 ± 1.00	88.69 ± 0.72	85.31 ± 0.81
	60	81.55 ± 2.66	89.48 ± 1.74	50.32 ± 1.94	91.21 ± 0.47	73.35 ± 3.55	89.62 ± 0.83	66.68 ± 7.22	86.68 ± 1.55	89.66 ± 1.10	73.81 ± 3.11	86.39 ± 1.42	83.24 ± 1.18	85.23 ± 0.54	77.30 ± 2.34
	40	72.51 ± 11.99	90.74 ± 0.46	46.54 ± 5.49	90.95 ± 0.60	74.39 ± 7.55	90.02 ± 0.58	67.32 ± 4.15	87.41 ± 0.99	89.79 ± 0.89	74.61 ± 4.16	86.90 ± 0.68	85.10 ± 1.65	85.80 ± 1.40	76.66 ± 0.74
	20	25.78 ± 8.13	63.79 ± 3.33	15.35 ± 8.54	57.94 ± 1.81	42.76 ± 2.00	50.99 ± 2.31	13.82 ± 4.95	54.62 ± 5.21	50.19 ± 5.38	48.18 ± 5.83	56.91 ± 3.82	54.60 ± 0.36	55.43 ± 5.22	48.48 ± 0.62
	10	24.11 ± 9.32	67.32 ± 5.98	12.99 ± 3.34	60.78 ± 1.47	42.24 ± 3.05	50.25 ± 2.02	18.50 ± 7.60	52.43 ± 14.01	54.94 ± 3.62	46.24 ± 3.54	55.24 ± 2.78	55.12 ± 3.54	56.73 ± 1.38	46.74 ± 0.80

Table A.4.25: Mean wF1[%] of the $TCNN^{Pamap2}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{Pamap2}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The $N_{conv} = c_4$ corresponding to highest wF1[%] is fixed, and $[10, 20, 60, 80]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. SD wF1[%] lies around 0.01. The mean from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion.

Dataset		$TCNN^{attribute}_{fuse} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$							$IMU\text{-}TCNN^{attribute}_{fuse} \uparrow^{Pamap2}_{\mathcal{D}_{Source}}$						
		JHMDB		CAD60		NTU RGB+D		**Baseline**	JHMDB		CAD60		NTU RGB+D		**Baseline**
		Synth	**Pose**	**Synth**	**Pose**	**Synth**	**Pose**	**%wF1**	**Synth**	**Pose**	**Synth**	**Pose**	**Synth**	**Pose**	**%wF1**
$\%D_{tr.}$	80	**89.64**	**89.60**	88.16	80.18	**88.73**	88.02	80.18	86.25	86.23	86.14	85.84	88.41	86.52	87.10
	60	**89.63**	86.93	85.76	**89.03**	85.95	85.15	81.55	**85.38**	**85.65**	83.63	83.49	**85.93**	**84.91**	86.68
	20	**58.70**	**49.00**	**51.98**	**50.27**	**81.98**	**70.71**	25.78	**67.71**	**70.01**	**64.43**	60.29	**66.96**	**73.68**	54.62
	10	**50.97**	**46.75**	**47.26**	**45.93**	**43.93**	**51.36**	24.11	40.20	40.20	40.20	40.20	**51.30**	**59.59**	52.43

A.4.4 *Order Picking*

Baseline

A 3-fold validation is followed as the dataset is scarce, with only three subjects per warehouse, in which one subject is taken as the testing set and the other two as the training set. The *IMU-TCNN*$^{softmax}_{fuse}$ contains three branches for each of the three OBD from the dataset, similarly to LARa-MM. The permutation test used $[12567, 17436, 16799]$ samples for each of the three persons as the test set. Table A.4.28 presents the performance of the baseline per subject and per warehouse A or B from the OPD. The architectures using the TPP fuser on the OPD_B show the best performance. This outcome is not the case for the OPD_A. The max and the spectral pooling do not negatively affect the performance on both warehouses, as Table A.4.28 and Table A.4.29 show. The *IMU-TCNN* architectures obtain significantly better performances for both warehouses. Computing temporal dependencies and obtaining a feature representation per limb show favourable behaviour. The FCN fuser shows a negative performance with respect to the other layers, similar to the performance on the other target datasets.

Table A.4.26 and Table A.4.27 present the learnt attributes using the EA-aimutcnn from LARa. The EA sets the *Error* attribute on the activities that are not appearing in LARa; this attribute was intended for such purpose. The *None* is found for the *Unknown* activity on the OPD_A. Interestingly, $1 - None$ represents also the *Unknown* activity on the OPD_B. The attributes *Gait Cycle* is part of the representation of *Walk* activity on the OPD_A as proposed by the LARa authors. Contrary, the *Step* attribute represents this activity on the OPD_B. Semantically, the *Step* refers to when the subject uses one of his feet as part of the main movement of either standing with minor steps or when handling with an additional lounge-like movement.

Transfer Learning for Order Picking

Nevertheless, the architectures easily overfit to the training data, so deploying a feature extractor learnt from the related LARa dataset helps reduce the effect of the overfitting; thus, Transfer Learning for HAR shows a positive influence on the performance when freezing the transferred convolutional layers. The LARa-SOBD becomes an interestingly good source for the OPD dataset.

Table A.4.26: Attribute representation $A \in \mathbb{B}^{19}$ of the OPD_A dataset found using the evolutionary algorithm and the $TCNN_{\text{FC}}^{\text{attribute}}$.

Activity	**Attributes**																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
UNKNOWN	1	0	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0
FLIP	1	0	0	1	1	0	1	0	0	0	0	1	1	0	1	0	1	0	1
WALK	1	1	0	0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	0
SEARCH	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	1	1
PICK	0	0	1	1	0	1	1	0	0	0	1	1	1	1	0	1	1	0	0
INFO	0	0	0	0	0	1	0	1	1	1	0	0	1	1	0	1	0	0	0
CARRY	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	1
ACK	1	0	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0	1

Table A.4.27: Attribute representation $A \in \mathbb{B}^{19}$ of the OPD_B dataset found using the evolutionary algorithm and the $TCNN_{\text{FC}}^{\text{attribute}}$.

Activity	**Attributes**																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
UNKNOWN	0	0	1	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0
FLIP	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	1	1
WALK	0	1	0	1	0	1	1	0	0	0	0	1	0	0	1	1	0	1	1
SEARCH	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0
PICK	1	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	1	0
SCAN	0	0	1	1	0	1	0	1	1	1	0	1	1	0	0	0	0	1	1
INFO	1	0	1	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1

Table A.4.28: The *w*F1 computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the two sets of OPD. The mean and std from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same fuse layer.

	$TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
OPD_A P_1	70.94 ± 1.20	70.67 ± 0.79	70.94 ± 1.01	63.89 ± 2.78	68.49 ± 3.12	70.09 ± 1.49	59.46 ± 0.57	60.61 ± 0.66	60.75 ± 0.32	58.24 ± 1.17	59.68 ± 1.06	60.61 ± 0.63
OPD_A P_2	60.17 ± 1.63	59.61 ± 2.47	61.52 ± 2.10	62.08 ± 1.34	63.56 ± 2.05	61.37 ± 1.74	57.86 ± 0.77	58.41 ± 1.16	58.70 ± 0.52	55.43 ± 1.88	56.57 ± 1.45	57.60 ± 1.47
OPD_A P_3	67.22 ± 1.99	67.22 ± 0.99	66.96 ± 1.59	61.04 ± 1.40	62.49 ± 1.34	63.84 ± 1.49	57.34 ± 0.66	57.89 ± 0.84	57.44 ± 0.73	52.45 ± 1.18	54.26 ± 0.71	54.50 ± 1.39
OPD_A	66.11 ± 5.47	65.83 ± 5.66	66.47 ± 4.73	62.34 ± 1.44	64.85 ± 3.20	65.10 ± 4.50	58.22 ± 1.11	58.97 ± 1.44	58.96 ± 1.67	55.38 ± 2.90	56.84 ± 2.72	57.57 ± 3.06
OPD_B P_1	72.17 ± 2.45	71.09 ± 4.28	68.56 ± 2.07	71.17 ± 3.59	72.02 ± 1.41	72.02 ± 1.47	72.40 ± 0.43	72.57 ± 0.24	72.78 ± 0.88	71.17 ± 2.02	73.55 ± 2.70	71.03 ± 1.85
OPD_B P_2	70.42 ± 0.30	70.33 ± 0.57	70.87 ± 0.60	68.10 ± 1.01	68.47 ± 0.45	72.02 ± 1.47	60.33 ± 0.43	60.21 ± 0.82	60.24 ± 0.81	61.29 ± 1.40	62.95 ± 0.58	61.32 ± 0.67
OPD_B P_3	81.20 ± 0.62	82.15 ± 0.95	82.22 ± 0.48	81.06 ± 0.91	81.23 ± 0.21	81.68 ± 0.91	76.98 ± 0.41	76.83 ± 0.65	77.06 ± 0.19	78.16 ± 1.01	77.81 ± 0.71	76.86 ± 1.35
OPD_B	74.60 ± 5.79	74.52 ± 6.61	73.88 ± 7.31s	73.44 ± 6.77	73.91 ± 6.58	73.72 ± 7.25	69.90 ± 8.60	69.87 ± 8.63	70.03 ± 8.74	70.21 ± 1.48	71.43 ± 7.65	69.74 ± 7.85

	$IMU\text{-}TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
OPD_A P_1	70.09 ± 1.90	70.89 ± 0.45	70.92 ± 0.56	67.10 ± 1.23	68.17 ± 1.32	70.17 ± 1.35	62.00 ± 0.43	63.26 ± 0.68	63.55 ± 0.41	63.92 ± 1.12	63.19 ± 0.53	62.90 ± 1.31
OPD_A P_2	61.12 ± 2.26	60.60 ± 0.78	60.29 ± 1.73	65.51 ± 1.67	64.86 ± 1.93	65.81 ± 2.41	61.41 ± 0.72	61.26 ± 0.56	61.74 ± 0.58	57.27 ± 0.90	59.52 ± 1.06	60.29 ± 0.49
OPD_A P_3	60.70 ± 0.96	64.47 ± 1.28	66.42 ± 3.49	62.35 ± 1.40	62.61 ± 1.29	63.28 ± 0.95	57.97 ± 0.31	57.41 ± 0.44	57.79 ± 0.30	57.52 ± 0.78	57.75 ± 0.83	57.64 ± 1.36
OPD_A	65.07 ± 4.58	65.32 ± 5.20	65.29 ± 5.34	64.44 ± 3.34	65.22 ± 2.79	64.66 ± 2.95	60.46 ± 2.18	60.64 ± 2.98	61.03 ± 2.94	59.57 ± 3.77	60.15 ± 2.78	60.28 ± 2.63
OPD_B P_1	63.34 ± 7.88	61.99 ± 8.98	66.69 ± 1.84	65.28 ± 2.43	66.63 ± 1.77	65.79 ± 2.23	71.90 ± 1.45	73.69 ± 1.24	71.02 ± 2.31	76.55 ± 1.24	76.08 ± 1.01	76.13 ± 0.83
OPD_B P_2	70.01 ± 0.29	70.95 ± 0.51	41.71 ± 1.98	69.26 ± 0.43	69.93 ± 0.45	69.63 ± 0.41	62.11 ± 0.33	62.90 ± 0.50	62.72 ± 0.72	66.50 ± 0.92	64.54 ± 0.46	64.74 ± 0.83
OPD_B P_3	83.64 ± 0.44	84.32 ± 0.60	59.90 ± 3.65	84.45 ± 0.60	84.07 ± 0.63	84.61 ± 0.40	79.03 ± 0.45	79.10 ± 0.39	79.04 ± 0.33	82.18 ± 0.31	82.14 ± 0.58	81.65 ± 0.53
OPD_B	72.33 ± 10.35	72.42 ± 11.24	73.31 ± 9.48	73.00 ± 10.12	73.54 ± 9.26	73.34 ± 9.94	71.01 ± 8.49	71.90 ± 8.25	70.93 ± 8.16	75.08 ± 0.82	74.25 ± 8.94	74.17 ± 8.62

Table A.4.29: The testing *w*F1 [%] computed from solving M-HAR using the $TCNN^{softmax}_{TPP}$ and the $IMU\text{-}TCNN^{softmax}_{TPP}$ on the OBD datasets. The mean and SD from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same pooling layer.

Dataset	$TCNN^{softmax}_{FC}$			$TCNN^{softmax}_{TPP}$			$IMU\text{-}TCNN^{softmax}_{FC}$			$IMU\text{-}TCNN^{softmax}_{TPP}$		
	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool
OPD_A	66.11 ± 5.47	65.28 ± 5.36	65.72 ± 4.71	55.38 ± 2.90	57.17 ± 2.30	57.22 ± 2.51	65.07 ± 4.58	66.34 ± 4.35	65.91 ± 3.89	59.57 ± 3.77	60.33 ± 3.26	59.95 ± 3.61
OPD_B	74.60 ± 5.79	74.94 ± 0.93	74.66 ± 1.27	70.21 ± 1.48	70.31 ± 1.17	70.11 ± 1.17	72.33 ± 10.35	72.25 ± 2.10	73.19 ± 2.27	75.08 ± 0.82	73.59 ± 0.75	74.48 ± 0.61

Table A.4.30: The wF1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the OPD datasets. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to classifier.

Dataset	$TCNN^{softmax}_{fuse}$		$TCNN^{attribute}_{fuse}$		$IMU\text{-}TCNN^{softmax}_{fuse}$		$IMU\text{-}TCNN^{attribute}_{fuse}$	
	FC	TPP	FC	TPP	FC	TPP	FC	TPP
OPD_A	66.11 ± 5.47	55.38 ± 2.90	63.79 ± 6.51	53.19 ± 1.93	65.07 ± 4.58	59.57 ± 3.77	61.86 ± 4.35	57.61 ± 3.59
OPD_B	74.60 ± 5.79	70.21 ± 1.48	73.63 ± 5.84	67.93 ± 9.34	72.33 ± 10.35	75.08 ± 0.82	74.28 ± 6.70	74.35 ± 9.07

Table A.4.31: Mean wF1[%] of the $TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $c_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{Source}$. Subsequently, the N_{conv} corresponding to highest wF1[%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		Baseline	$TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$						Baseline	$IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$					
			LARa-M		LARa-SOBD		LARa-Mb			LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
N_{conv} Tr. Layers	c_1	66.11 ± 5.47	67.85 ± 3.56	56.03 ± 2.25	65.80 ± 5.16	54.55 ± 1.25	68.29 ± 2.50	54.15 ± 1.49	65.07 ± 4.58	66.62 ± 3.78	59.92 ± 2.88	65.68 ± 3.95	59.19 ± 2.86	67.40 ± 3.86	59.80 ± 2.77
	$c_{1,2}$		66.37 ± 5.34	51.72 ± 4.33	65.52 ± 4.87	55.54 ± 1.25	67.64 ± 3.55	45.88 ± 1.80		65.32 ± 4.47	61.82 ± 3.01	66.35 ± 4.59	59.49 ± 3.34	68.46 ± 4.55	60.09 ± 3.17
	c_{1-3}		66.84 ± 3.93	52.57 ± 3.19	66.65 ± 4.22	56.26 ± 2.09	67.58 ± 3.90	48.55 ± 9.26		66.55 ± 3.51	62.68 ± 3.78	66.69 ± 4.11	59.29 ± 3.20	68.87 ± 4.26	60.41 ± 3.38
	c_{1-4}		66.81 ± 2.82	55.02 ± 4.89	67.36 ± 3.14	56.32 ± 3.89	68.59 ± 2.80	55.45 ± 3.71		64.17 ± 2.63	61.79 ± 2.86	66.29 ± 4.38	60.57 ± 3.34	68.48 ± 4.01	62.00 ± 3.81
	c_{1-4}		69.78 ± 2.88	14.24 ± 3.34	70.56 ± 4.14	41.10 ± 4.42	69.66 ± 3.24	30.53 ± 6.59		67.60 ± 3.13	58.92 ± 1.84	66.89 ± 3.25	62.34 ± 3.71	67.14 ± 5.21	60.50 ± 5.34

Dataset		Baseline	$TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$						Baseline	$IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{OPD}_{\mathcal{D}_{Source}}$					
			LARa-M		LARa-SOBD		LARa-Mb			LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
N_{conv} Tr. Layers	c_1	74.60 ± 5.79	71.24 ± 9.65	63.12 ± 9.35	73.40 ± 7.23	66.16 ± 11.14	72.67 ± 7.99	60.59 ± 5.75	72.33 ± 10.35	73.75 ± 9.12	74.33 ± 8.20	74.12 ± 9.49	72.87 ± 8.50	73.96 ± 9.84	73.07 ± 9.89
	$c_{1,2}$		74.22 ± 6.59	68.19 ± 8.11	74.20 ± 6.78	67.46 ± 6.26	74.05 ± 7.75	59.52 ± 3.44		73.61 ± 9.49	75.12 ± 7.31	74.43 ± 9.02	75.20 ± 7.23	73.75 ± 10.13	74.54 ± 7.75
	c_{1-3}		73.71 ± 6.84	66.67 ± 8.01	74.41 ± 6.44	69.19 ± 8.79	73.31 ± 7.91	63.03 ± 11.36		75.05 ± 8.60	73.67 ± 6.40	74.14 ± 9.45	75.44 ± 6.75	75.32 ± 8.54	73.96 ± 6.17
	c_{1-4}		73.02 ± 8.24	65.37 ± 8.16	74.72 ± 6.36	69.83 ± 7.72	73.58 ± 7.82	63.52 ± 8.59		75.14 ± 8.20	74.56 ± 7.23	72.80 ± 11.13	75.49 ± 7.54	74.75 ± 9.44	73.26 ± 7.64
	c_{1-4}		73.96 ± 8.55	22.71 ± 10.66	74.41 ± 7.30	28.38 ± 9.77	74.97 ± 6.27	18.30 ± 9.93		75.52 ± 7.66	71.57 ± 8.69	73.36 ± 9.10	71.38 ± 11.33	71.08 ± 12.20	67.08 ± 10.83

A.4.5 *MotionMiners*

Baseline

Table A.4.33 presents the performance of the baseline for MM. In general, the *IMU-tCNN* with [FC, LSTM] fusers perform relatively equally. The FCN layer does not perform as well as the other two fusers. It indicates that the tCNN layers are good extractors. Layers allowing a holistic aggregation along the time of the features are required for the classification of the sequence.

Table A.4.32 shows the learnt attributes using the EA-Attr-IMU-tCNN trained with the LARa dataset. The EA allows finding the *Error* only for the *Background* activity. However, the attributes related to displacement show a variate outcome. The attributes *Gait Cycle* and *Step* represent the *Standing* activity, and the *Standing Still* and *Step* attributes represent the *Walking* activity. Both the *Standing* and *Walking* share the attribute *Step*, which was intended to short movements of the leg, either at standing or at handling items; hence, there is a sort of relation to the definition of the attribute given by LARa. However, this outcome also indicates that the annotation given to *Walking* from MM also considers short leg movements that are not necessarily a gait cycle. Nevertheless, the attributes related to order picking or handling are not found on the *Walking* activity, e.g., *Right Hand*, *Left Hand*, *Bulky Unit*, and *Handy Unit*. The attributes associated with handling activities in the LARa represent the *Handling* activity from MM, namely *Upwards*, *Centred*, *Downwards*, and the hand attributes*Right Hand*, and *Left Hand*.

Table A.4.32: Attribute representation $A \in \mathbb{B}^{19}$ of the MM dataset found using the evolutionary algorithm and the $TCNN_{FC}^{attribute}$.

Activity	**Attributes**																		
	Gait Cycle	Step	Standing Still	Upwards	Centred	Downwards	No Intentional Motion	Torso Rotation	Right Hand	Left Hand	No Arms	Bulky Unit	Handy Unit	Tool	Cart	Computer	No Item	None	Error
Background	1	1	0	0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
Walking	0	1	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0
Standing	1	1	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	1	0
Handling	0	0	1	1	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0
Driving	0	1	0	1	1	1	0	1	0	0	0	1	0	0	1	1	0	0	0
Sitting	1	0	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0

Transfer Learning for MotionMiners

For observing the effects of TL, five different versions of the LARa-MM are created. These versions contain different proportions of the datasets, specifically, with $\% = [10, 20, 40, 60, 80]$ of the recordings from the training set; respectively—the validation and testing sets remain equal.

Table A.4.36 displays the performance for M-HAR on the different proportions of the MM, and when transferring the convolutional layers. Interestingly, the *IMU-TCNN* perform relatively well when facing the situation when a proportion of the dataset is available as annotated, compared to the textitTCNN. This outcome empirically supports that the limb-oriented structure of the *IMU-TCNN* computes a descriptive representation of the sequence according to the limb movements related to the movement. All the source datasets provide relatively good material for pretraining deep architectures for improving M-HAR on MM. The *IMU-TCNN* using the TPP layer shows the best performance.

Table A.4.33: The *w*F1 computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the MM. The mean and std from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same fuse layer.

	$TCNN^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
MM	82.89 ± 0.58	82.87 ± 0.37	83.04 ± 0.43	82.87 ± 0.63	82.41 ± 0.16	81.93 ± 1.10	$52.84 \pm 1, .2$	51.60 ± 2.69	51.31 ± 2.55	66.44 ± 1.80	68.06 ± 0.38	68.58 ± 0.56
	IMU-TCNN$^{softmax}_{fuse}$											
Dataset	FC			LSTM			FCN			TPP		
	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]NoPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool	[0]MaxPool	[1]MaxPool	[1-2]MaxPool
MM	82.75 ± 0.46	82.64 ± 0.84	83.13 ± 0.61	80.22 ± 2.05	79.96 ± 1.67	80.70 ± 0.59	77.81 ± 0.54	78.54 ± 0.32	78.99 ± 0.13	82.72 ± 0.31	82.79 ± 0.55	82.67 ± 0.22

Table A.4.34: The testing *w*F1 [%] computed from solving M-HAR using the $TCNN^{softmax}_{TPP}$ and the *IMU-TCNN*$^{softmax}_{TPP}$ on the OBD datasets. The mean and SD from the *w*F1 are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the same pooling layer.

Dataset	$TCNN^{softmax}_{FC}$			$TCNN^{softmax}_{TPP}$			*IMU-TCNN*$^{softmax}_{FC}$			*IMU-TCNN*$^{softmax}_{TPP}$		
	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool	[0]NoPool	[1]SpectralPool	[1-2]SpectralPool
MM	82.89 ± 0.58	83.07 ± 0.64	82.84 ± 0.39	66.44 ± 1.80	67.88 ± 0.71	67.72 ± 0.87	82.75 ± 0.46	81.70 ± 1.39	83.23 ± 0.58	82.72 ± 0.31	82.58 ± 0.49	81.68 ± 0.86

Table A.4.35: The *w*F1 [%] computed from solving M-HAR using the *TCNN* and the *IMU-TCNN* on the MM datasets. The mean and SD from the *w*F1 [%] are given as training procedure is repeated 5×. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to classifier.

Dataset	$TCNN^{softmax}_{fuse}$		$TCNN^{attribute}_{fuse}$		*IMU-TCNN*$^{softmax}_{fuse}$		*IMU-TCNN*$^{attribute}_{fuse}$	
	FC	TPP	FC	TPP	FC	TPP	FC	TPP
MM	82.89 ± 0.58	66.44 ± 1.80	83.33 ± 0.29	72.50 ± 4.57	82.75 ± 0.46	82.72 ± 0.31	83.35 ± 0.86	84.76 ± 0.34

Table A.4.36: Mean wF1[%] of the $TCNN^{softmax}_{fuse} \uparrow^{MM}_{\mathcal{D}_{Source}}$ and $IMU\text{-}TCNN^{softmax}_{fuse} \uparrow^{MM}_{\mathcal{D}_{Source}}$ using the joint poses and the synthetic data. The N_{conv} changes from c_1 to $c_{1,2,3,4}$ keeping 100% of the $\mathcal{D}_{Source}$. Subsequently, the N_{conv} corresponding to highest wF1[%] is fixed and $[10, 30, 50, 75]$% of the $\mathcal{D}_{Target}$ are deployed for fine-tuning. The mean and SD from the wF1 [%] are given as training procedure is repeated $5\times$. Values in bold are not statistically significant to the highest; hence, they represent the best performance. The comparison are carried out with respect to the data proportion or the transposed layers.

Dataset		Baseline	$TCNN^{softmax}_{FC} \uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$						Baseline	$IMU\text{-}TCNN^{softmax}_{FC} \uparrow^{LARa-MM}_{\mathcal{D}_{Source}}$					
			LARa-M		LARa-SOBD		LARa-Mb			LARa-M		LARa-SOBD		LARa-Mb	
			FC	TPP	FC	TPP	FC	TPP		FC	TPP	FC	TPP	FC	TPP
Tr. Layers	c_1	82.89 ± 0.58	82.40 ± 0.48	78.63 ± 0.85	81.90 ± 0.46	77.97 ± 1.58	81.88 ± 0.51	79.55 ± 1.22	82.75 ± 0.46	83.57 ± 1.36	85.61 ± 0.53	84.69 ± 0.97	85.80 ± 0.26	83.06 ± 0.28	85.92 ± 0.33
	$c_{1,2}$		82.25 ± 0.60	79.51 ± 0.97	81.76 ± 0.78	79.19 ± 0.89	83.20 ± 0.27	79.95 ± 0.65		81.61 ± 1.54	85.59 ± 0.28	83.96 ± 0.45	85.46 ± 0.38	83.34 ± 0.90	85.58 ± 0.44
	c_{1-3}		83.65 ± 0.52	79.22 ± 0.98	82.01 ± 0.58	79.02 0.90	83.54 ± 0.20	80.66 ± 0.33		83.76 ± 1.90	84.45 ± 0.27	83.17 ± 1.58	85.43 ± 0.28	82.65 ± 0.90	84.98 ± 0.77
	c_{1-4}		84.62 ± 0.35	70.53 ± 3.10	81.43 ± 0.65	78.93 ± 0.87	85.04 ± 0.67	79.85 ± 0.84		81.87 ± 0.96	83.32 ± 0.30	83.16 ± 0.88	84.83 ± 0.49	85.79 ± 0.44	86.22 ± 0.50
	c_{1-4}		85.63 ± 0.61	53.93 ± 0.20	82.88 ± 0.69	60.84 ± 3.22	85.57 ± 0.24	59.54 ± 0.10		72.02 ± 0.85	76.19 ± 0.23	73.75 ± 1.36	81.34 ± 0.09	84.43 ± 0.84	83.38 ± 0.25
$\%D_{Tr.}$	80	51.70 ± 7.59	85.14 ± 0.40	72.28 ± 2.20	82.49 ± 0.84	78.30 ± 0.98	84.19 ± 0.84	78.16 ± 0.56	77.30 ± 2.28	82.61 ± 1.13	83.85 ± 0.70	83.80 ± 1.00	85.00 ± 0.20	84.49 ± 0.88	85.51 ± 0.68
	60	50.03 ± 3.24	86.11 ± 0.21	70.10 ± 3.09	82.95 ± 0.81	77.89 ± 1.51	84.31 ± 0.60	77.58 ± 0.83	71.32 ± 9.02	80.69 ± 1.80	84.52 ± 0.65	83.09 ± 0.58	85.02 ± 0.88	82.82 ± 1.24	85.86 ± 0.66
	40	49.15 ± 7.09	83.92 ± 0.63	59.33 ± 1.76	80.45 ± 0.78	74.27 ± 0.60	83.50 ± 0.69	72.06 ± 2.61	74.51 ± 6.32	80.78 ± 0.71	80.52 ± 2.07	81.91 ± 0.54	82.98 ± 0.53	83.23 ± 0.91	84.19 ± 0.81
	20	46.15 ± 3.07	82.22 ± 0.82	58.38 ± 1.89	79.65 ± 2.43	70.57 ± 1.20	82.38 ± 1.06	69.94 ± 3.57	76.32 ± 1.07	77.47 ± 2.09	80.59 ± 0.76	79.21 ± 0.99	82.02 ± 0.41	81.34 ± 0.79	83.61 ± 0.70
	10	51.82 ± 6.07	81.59 ± 0.44	55.75 ± 1.59	79.12 ± 1.01	64.53 ± 3.36	81.21 ± 1.78	65.98 ± 3.61	68.71 ± 10.90	76.99 ± 1.78	79.73 ± 0.85	78.79 ± 0.68	81.64 ± 0.69	80.71 ± 0.64	82.92 ± 1.12

BIBLIOGRAPHY

[AB10] Altun, Kerem ; Barshan, Billur: Human Activity Recognition Using Inertial/Magnetic Sensor Units. In: *Human Behavior Understanding* Bd. 6219, Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–14715–9, S. 38–51

[ABT10] Altun, Kerem ; Barshan, Billur ; Tunçel, Orkun: Comparative study on classifying human activities with miniature inertial and magnetic sensors. In: *Pattern Recognition* 43 (2010), Nr. 10, 3605–3620. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2010.04.019. – DOI https://doi.org/10.1016/j.patcog.2010.04.019. – ISSN 0031–3203

[AEB06] Aharon, Michal ; Elad, Michael ; Bruckstein, Alfred: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. In: *IEEE Transactions on Signal Processing* 54 (2006), Nr. 11, S. 4311–4322. http://dx.doi.org/10.1109/TSP.2006.881199. – DOI 10.1109/TSP.2006.881199

[AGFV14] Almazán, Jon ; Gordo, Albert ; Fornés, Alicia ; Valveny, Ernest: Word Spotting and Recognition with Embedded Attributes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), Nr. 12, S. 2552–2566. http://dx.doi.org/10.1109/TPAMI.2014.2339814. – DOI 10.1109/TPAMI.2014.2339814

[AGO+12] Anguita, Davide ; Ghio, Alessandro ; Oneto, Luca ; Parra, Xavier ; Reyes-Ortiz, Jorge L.: Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In: *Ambient Assisted Living and Home Care* Bd. 7657, Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–35395–6, S. 216–223

[AI15] Aly, Heba ; Ismail, Mohamed A.: UbiMonitor: Intelligent Fusion of Body-Worn Sensors for Real-Time Human Activity Recognition. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. Salamanca, Spain : Association for Computing Machinery, 2015. – ISBN 978–1–4503–3196–8, 563–568

[APH+21] Abdar, Moloud ; Pourpanah, Farhad ; Hussain, Sadiq ; Rezazadegan, Dana ; Liu, Li ; Ghavamzadeh, Mohammad ; Fieguth, Paul ; Cao, Xiaochun ; Khosravi, Abbas ; Acharya, U. R. ; Makarenkov, Vladimir ; Nahavandi,

Saeid: A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges. In: *Inf. Fusion* 76 (2021), Dezember, Nr. C, 243–297. http://dx.doi.org/10.1016/j.inffus.2021.05.008. – DOI 10.1016/j.inffus.2021.05.008. – ISSN 1566–2535

[AR17] Alam, Mohammad Arif U. ; Roy, Nirmalya: Unseen Activity Recognitions: A Hierarchical Active Transfer Learning Approach. In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, S. 436–446

[BBS14] Bulling, Andreas ; Blanke, Ulf ; Schiele, Bernt: A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors. In: *ACM Computing Surveys* 46 (2014), Nr. 3. http://dx.doi.org/10.1145/2499621. – DOI 10.1145/2499621. – ISSN 0360–0300

[BHMVL21] Bock, Marius ; Hölzemann, Alexander ; Moeller, Michael ; Van Laerhoven, Kristof: Improving Deep Learning for HAR with Shallow LSTMs. In: *Proceedings of the 2021 ACM International Symposium on Wearable Computers*. New York, NY, USA : Association for Computing Machinery, 2021 (ISWC '21). – ISBN 978–1–4503–8462–9, 7–12. – event-place: Virtual, USA

[BPT14] Bayat, Akram ; Pomplun, Marc ; Tran, Duc A.: A Study on Human Activity Recognition Using Accelerometer Data from Smartphones. In: *Procedia Computer Science* 34 (2014), 450–457. http://dx.doi.org/https://doi.org/10.1016/j.procs.2014.07.009. – DOI https://doi.org/10.1016/j.procs.2014.07.009. – ISSN 1877–0509

[BV21] Buffelli, Davide ; Vandin, Fabio: Attention-Based Deep Learning Framework for Human Activity Recognition With User Adaptation. In: *IEEE Sensors Journal* 21 (2021), Nr. 12, S. 13474–13483. http://dx.doi.org/10.1109/JSEN.2021.3067690. – DOI 10.1109/JSEN.2021.3067690

[BYRN11] Baeza-Yates, Ricardo ; Ribeiro-Neto, Berthier: *Modern Information Retrieval: The Concepts and Technology behind Search*. 2nd. USA : Addison-Wesley Publishing Company, 2011. – ISBN 978–0–321–41691–9

[CFK13] Cook, Diane ; Feuz, Kyle D. ; Krishnan, Narayanan C.: Transfer learning for activity recognition: A survey. In: *Knowledge and information systems* 36 (2013), Nr. 3, S. 537–556. http://dx.doi.org/https://doi.org/10.1007/s10115-013-0665-3. – DOI https://doi.org/10.1007/s10115–013–0665–3

[CFP+16] Chahuara, Pedro ; Fleury, Anthony ; Portet, François ; Vacher, Michel ; Hunter, Gordon ; Kymäläinen, Tiina ; Herrera-Acuña, Raúl: On-Line

Human Activity Recognition from Audio and Home Automation Sensors: Comparison of Sequential and Non-Sequential Models in Realistic Smart Homes1. In: *J. Ambient Intell. Smart Environ.* 8 (2016), Januar, Nr. 4, 399–422. http://dx.doi.org/10.3233/AIS-160386. – DOI 10.3233/AIS–160386. – ISSN 1876–1364

[CGD+13] Cheng, Heng-Tze ; Griss, Martin ; Davis, Paul ; Li, Jianguo ; You, Di: Towards zero-shot learning for human activity recognition using semantic attribute sequence model. In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Association for Computing Machinery, 2013. – ISBN 978–1–4503–1770–2, S. 355–358

[CGS18] Chikhaoui, Belkacem ; Gouineau, Frank ; Sotir, Martin: A CNN Based Transfer Learning Model for Automatic Activity Recognition from Accelerometer Sensors. In: *International Conference on Machine Learning and Data Mining in Pattern Recognition*, Springer, 2018, 302–315

[CLCG18] Chen, Zhenghua ; Le, Zhang ; Cao, Zhiguang ; Guo, Jing: Distilling the Knowledge From Handcrafted Features for Human Activity Recognition. In: *IEEE Transactions on Industrial Informatics* 14 (2018), Nr. 10, 4334–4342. http://dx.doi.org/10.1109/TII.2018.2789925. – DOI 10.1109/TII.2018.2789925. – ISSN 1551–3203, 1941–0050

[CLW+21] Chen, Wei ; Liu, Yu ; Wang, Weiping ; Bakker, Erwin ; Georgiou, Theodoros ; Fieguth, Paul ; Liu, Li ; Lew, Michael S.: Deep Image Retrieval: A Survey. (2021). https://arxiv.org/abs/2101.11282

[CPR11] Casale, Pierluigi ; Pujol, Oriol ; Radeva, Petia: Human Activity Recognition from Accelerometer Data Using a Wearable Device. In: *Pattern Recognition and Image Analysis*, Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–21257–4, S. 289–296

[Cra05] Craig, John J.: *Introduction to robotics: Mechanics and control*. Bd. 3. Pearson Educacion, 2005. – ISBN 0–13–123629–6

[CSC+13a] Chavarriaga, Ricardo ; Sagha, Hesam ; Calatroni, Alberto ; Digumarti, Sundara T. ; Tröster, Gerhard ; Millán, José del R. ; Roggen, Daniel: *OPPORTUNITY Activity Recognition Dataset*. "https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition". Version: 2013. – The OPPORTUNITY Dataset for Human Activity Recognition from Wearable, Object, and Ambient Sensors is a dataset devised to benchmark human

activity recognition algorithms (classification, automatic data segmentation, sensor fusion, feature extraction, etc).

[CSC+13b] Chavarriaga, Ricardo ; Sagha, Hesam ; Calatroni, Alberto ; Digumarti, Sundara T. ; Tröster, Gerhard ; Millán, José del R. ; Roggen, Daniel: The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. In: *Pattern Recognition Letters* 34 (2013), Nr. 15, S. 2033–2042. http://dx.doi.org/https://doi.org/10.1016/j.patrec.2012.12.014. – DOI https://doi.org/10.1016/j.patrec.2012.12.014. – ISSN 0167–8655

[CSG+13] Cheng, Heng-Tze ; Sun, Feng-Tso ; Griss, Martin ; Davis, Paul ; Li, Jianguo ; You, Di: NuActiv: Recognizing Unseen New Activities Using Semantic Attribute-Based Learning. In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. Taipei, Taiwan : Association for Computing Machinery, 2013 (MobiSys '13). – ISBN 978–1–4503–1672–9, S. 361–374

[CSR+21] Civitarese, Gabriele ; Sztyler, Timo ; Riboni, Daniele ; Bettini, Claudio ; Stuckenschmidt, Heiner: POLARIS: Probabilistic and Ontological Activity Recognition in Smart-Homes. In: *IEEE Transactions on Knowledge and Data Engineering* 33 (2021), Nr. 1, S. 209–223. http://dx.doi.org/10.1109/TKDE.2019.2930050. – DOI 10.1109/TKDE.2019.2930050

[CSWS17] Cao, Zhe ; Simon, Tomas ; Wei, Shih-En ; Sheikh, Yaser: Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, S. 1302–1310

[CVN+20] Cruciani, Federico ; Vafeiadis, Anastasios ; Nugent, Chris ; Cleland, Ian ; McCullagh, Paul ; Votis, Konstantinos ; Giakoumis, Dimitrios ; Tzovaras, Dimitrios ; Chen, Liming ; Hamzaoui, Raouf: Feature learning for Human Activity Recognition using Convolutional Neural Networks. In: *CCF Transactions on Pervasive Computing and Interaction* 2 (2020), Nr. 1, S. 18–32. http://dx.doi.org/10.1007/s42486-020-00026-2. – DOI 10.1007/s42486–020–00026–2

[CWRS18] Choutas, Vasileios ; Weinzaepfel, Philippe ; Revaud, Jérôme ; Schmid, Cordelia: PoTion: Pose MoTion Representation for Action Recognition. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA : IEEE, 2018, S. 7024–7033

[CX15] Chen, Yuqing ; Xue, Yang: A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer. In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. Hong Kong, China : IEEE, 2015. – ISBN 978–1–4799–8697–2, 1488–1492

[CZY+21] Chen, Kaixuan ; Zhang, Dalin ; Yao, Lina ; Guo, Bin ; Yu, Zhiwen ; Liu, Yunhao: Deep Learning for Sensor-Based Human Activity Recognition: Overview, Challenges, and Opportunities. In: *ACM Comput. Surv.* 54 (2021), Mai, Nr. 4. http://dx.doi.org/10.1145/3447744. – DOI 10.1145/3447744. – ISSN 0360–0300. – Place: New York, NY, USA Publisher: Association for Computing Machinery

[DBLG14] Duffner, Stefan ; Berlemont, Samuel ; Lefebvre, Grégoire ; Garcia, Christophe: 3D gesture classification with convolutional neural networks. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy : IEEE, 2014, S. 5432–5436

[DLGY12] Deng, Liqun ; Leung, Howard ; Gu, Naijie ; Yang, Yang: Generalized Model-Based Human Motion Recognition with Body Partition Index Maps. In: *Computer Graphics Forum* 31 (2012), Nr. 1, S. 202–215. http://dx.doi.org/10.1111/j.1467-8659.2011.02095.x. – DOI 10.1111/j.1467–8659.2011.02095.x. – ISSN 0167–7055

[DLKP22] Dirgová Luptáková, Iveta ; Kubovčík, Martin ; Pospíchal, Jiří: Wearable Sensor-Based Human Activity Recognition with Transformer Model. In: *Sensors* 22 (2022), Nr. 5. http://dx.doi.org/10.3390/s22051911. – DOI 10.3390/s22051911. – ISSN 1424–8220

[DPBR20] Demrozi, Florenc ; Pravadelli, Graziano ; Bihorac, Azra ; Rashidi, Parisa: Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey. In: *IEEE Access* 8 (2020), S. 210816–210836. http://dx.doi.org/10.1109/ACCESS.2020.3037715. – DOI 10.1109/ACCESS.2020.3037715

[DTR18] Dalmazzo, David ; Tassani, Simone ; Ramírez, Rafael: A Machine Learning Approach to Violin Bow Technique Classification: A Comparison Between IMU and MOCAP Systems. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. Berlin, Germany : Association for Computing Machinery, 2018. – ISBN 978–1–4503–6487–4

[DXTL10] Duan, Lixin ; Xu, Dong ; Tsang, Ivor W. ; Luo, Jiebo: Visual event recognition in videos by learning from web data. In: *2010 IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition. San Francisco, CA, USA : IEEE, 2010, S. 1959–1966

[EH20] Engelen, Jesper E. ; Hoos, Holger H.: A survey on semi-supervised learning. In: *Machine Learning* 109 (2020), Februar, Nr. 2, 373–440. http://dx.doi.org/10.1007/s10994-019-05855-6. – DOI 10.1007/s10994–019–05855–6. – ISSN 1573–0565

[FAt16] Feldhorst, Sascha ; Aniol, Sandra ; ten Hompel, Michael: Human Activity Recognition in der Kommissionierung – Charakterisierung des Kommissionierprozesses als Ausgangsbasis für die Methodenentwicklung. In: *Logistics Journal : Proceedings* (2016), Nr. 10. http://www.logistics-journal.de/proceedings/2016/fachkolloquium2016/4473. – ISSN 1860–7977

[FEHF09] Farhadi, Ali ; Endres, Ian ; Hoiem, Derek ; Forsyth, David: Describing objects by their attributes. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, S. 1778–1785

[FK16] Fallmann, Sarah ; Kropf, Johannes: Human Activity Pattern Recognition based on Continuous Data from a Body Worn Sensor placed on the Hand Wrist using Hidden Markov Models. In: *Simulation Notes Europe SNE* 26 (2016), Nr. 1, 9–16. http://dx.doi.org/10.11128/sne.26.tn.10322. – DOI 10.11128/sne.26.tn.10322. – ISSN 23059974, 23060271

[FM82] Fukushima, Kunihiko ; Miyake, Sei: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. In: *Pattern Recognition* 15 (1982), Nr. 6, S. 455–469. http://dx.doi.org/https://doi.org/10.1016/0031-3203(82)90024-3. – DOI https://doi.org/10.1016/0031–3203(82)90024–3

[FMHF16] Feldhorst, Sascha ; Masoudenijad, Mojtaba ; Hompel, Michael ten ; Fink, Gernot A.: Motion classification for analyzing the order picking process using mobile sensors. In: *Proc. Int. Conf. Pattern Recognition Applications and Methods*. Rome, Italy, 2016 (ICPRAM 2016). – ISBN 978–989–758–173–1, 706–713

[FPZ16] Feichtenhofer, Christoph ; Pinz, Axel ; Zisserman, Andrew: Convolutional Two-Stream Network Fusion for Video Action Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 1933–1941

[GBB11] Glorot, Xavier ; Bordes, Antoine ; Bengio, Yoshua: Deep Sparse Rectifier Neural Networks. In: *Proceedings of the Fourteenth International Conference*

on Artificial Intelligence and Statistics Bd. 15, PMLR, 2011 (Proceedings of Machine Learning Research), 315–323

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016 http://www.deeplearningbook.org. – ISBN 0–262–03561–8

[GCBCJG14] Garcia-Ceja, Enrique ; Brena, Ramon F. ; Carrasco-Jimenez, Jose C. ; Garrido, Leonardo: Long-Term Activity Recognition from Wristwatch Accelerometer Data. In: *Sensors* 14 (2014), Nr. 12, 22500–22524. http://dx.doi.org/10.3390/s141222500. – DOI 10.3390/s141222500. – ISSN 1424–8220

[GD14] Gupta, Piyush ; Dallas, Tim: Feature Selection and Activity Recognition System Using a Single Triaxial Accelerometer. In: *IEEE Transactions on Biomedical Engineering* 61 (2014), Nr. 6, 1780–1786. http://dx.doi.org/10.1109/TBME.2014.2307069. – DOI 10.1109/TBME.2014.2307069

[GMH13] Graves, Alex ; Mohamed, Abdel-rahman ; Hinton, Geoffrey: Speech recognition with deep recurrent neural networks. In: *International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada : IEEE, 2013, S. 6645–6649

[GSF18] Gurjar, Neha ; Sudholt, Sebastian ; Fink, Gernot A.: Learning Deep Representations for Word Spotting under Weak Supervision. In: *13th IAPR International Workshop on Document Analysis Systems (DAS)*. Vienna, Austria : IEEE, 2018, S. 7–12

[GW15] Guo, Ming ; Wang, Zhelong: A feature extraction method for human action recognition using body-worn inertial sensors. In: *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. Calabria, Italy : IEEE, 2015. – ISBN 978–1–4799–2002–0, S. 576–581

[GY17] Ge, Weifeng ; Yu, Yizhou: Borrowing Treasures from the Wealthy: Deep Transfer Learning through Selective Joint Fine-Tuning. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA : IEEE, 2017, S. 10–19

[HHP16] Hammerla, Nils ; Halloran, Shane ; Plötz, Thomas: Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligenc* Bd. 8. New York, New York, USA : AAAI Press, 2016 (IJCAI'16). – ISBN 978–1–57735–770–4, S. 1533–1540

[HKA+18] Huang, Yinghao ; Kaufmann, Manuel ; Aksan, Emre ; Black, Michael J. ; Hilliges, Otmar ; Pons-Moll, Gerard: Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time. In: *ACM Trans. Graph.* 37 (2018), Nr. 6. http://dx.doi.org/10.1145/3272127.3275108. – DOI 10.1145/3272127.3275108. – ISSN 0730–0301

[HM19] Haines, Duane E. ; Mihailoff, Gregory A.: *Principios de Neurociencia: Aplicaciones Básicas Y Clínicas*. Elsevier Health Sciences, 2019 https://books.google.de/books?id=m5GFDwAAQBAJ. – ISBN 978–84–9113–500–5

[HMS16] Hoettinger, Hannes ; Mally, Franziska ; Sabo, Anton: Activity Recognition in Surfing - A Comparative Study between Hidden Markov Model and Support Vector Machine. In: *11th Conference of the International Sports Engineering Association (ISEA 2016)* Bd. 147. Delft, Netherlands : Elsevier Procedia, Dezember 2016, S. 912–917

[HMSB18] Haescher, Marian ; Matthies, Denys J. ; Srinivasan, Karthik ; Bieber, Gerald: Mobile Assisted Living: Smartwatch-Based Fall Risk Assessment for Elderly People. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. New York, NY, USA : Association for Computing Machinery, 2018 (iWOAR '18). – ISBN 978–1–4503–6487–4. – event-place: Berlin, Germany

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), November, Nr. 8, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735. – ISSN 0899–7667

[HSU12] Hachiya, Hirotaka ; Sugiyama, Masashi ; Ueda, Naonori: Importance-Weighted Least-Squares Probabilistic Classifier for Covariate Shift Adaptation with Application to Human Activity Recognition. In: *Neurocomputing* 80 (2012), Nr. C, 93–101. http://www.ms.k.u-tokyo.ac.jp/sugi/2012/IWLSPC.pdf. – ISSN 0925–2312

[HVL18] Hölzemann, Alexander ; Van Laerhoven, Kristof: Using Wrist-Worn Activity Recognition for Basketball Game Analysis. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. New York, NY, USA : Association for Computing Machinery, 2018 (iWOAR '18). – ISBN 978–1–4503–6487–4

[HY11] Hu, Derek H. ; Yang, Qiang: Transfer Learning for Activity Recognition via Sensor Mapping. In: *Proceedings of the Twenty-Second International Joint*

Conference on Artificial Intelligence Bd. 3. Barcelona, Catalonia, Spain : AAAI Press, 2011 (IJCAI'11). – ISBN 978–1–57735–515–1, S. 1962–1967

[HZRS15] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (2015), Nr. 9, S. 1904–1916. http://dx.doi.org/10.1109/TPAMI.2015.2389824. – DOI 10.1109/TPAMI.2015.2389824. – ISSN 1939–3539

[IS15] Ioffe, Sergey ; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, JMLR.org, 2015 (ICML'15), S. 448–456. – event-place: Lille, France

[JGZ+13] Jhuang, Hueihan ; Gall, Juergen ; Zuffi, Silvia ; Schmid, Cordelia ; Black, Michael J.: Towards Understanding Action Recognition. In: *2013 IEEE International Conference on Computer Vision*. Sydney, NSW, Australia : IEEE, 2013, S. 3192–3199

[JK19] Jeong, Chi Y. ; Kim, Mooseop: An Energy-Efficient Method for Human Activity Recognition with Segment-Level Change Detection and Deep Learning. In: *Sensors* 19 (2019), Nr. 17. http://dx.doi.org/10.3390/s19173688. – DOI 10.3390/s19173688. – ISSN 1424–8220

[JMA22] Javidani, Ali ; Mahmoudi-Aznaveh, Ahmad: Learning representative temporal features for action recognition. In: *Multimedia Tools and Applications* 81 (2022), Januar, Nr. 3, 3145–3163. http://dx.doi.org/10.1007/s11042-021-11022-8. – DOI 10.1007/s11042–021–11022–8. – ISSN 1573–7721

[KCKL22] Kim, Yeon-Wook ; Cho, Woo-Hyeong ; Kim, Kyu-Sung ; Lee, Sangmin: Inertial-Measurement-Unit-Based Novel Human Activity Recognition Algorithm Using Conformer. In: *Sensors* 22 (2022), Nr. 10. http://dx.doi.org/10.3390/s22103932. – DOI 10.3390/s22103932. – ISSN 1424–8220

[KEK08] Kasteren, Tim van ; Englebienne, Gwenn ; Krose, Ben: Recognizing Activities in Multiple Contexts using Transfer Learning. In: *AAAI fall symposium: AI in eldercare: new solutions to old problems*, 2008, 142–149

[KEK10] Kasteren, Tim van ; Englebienne, Gwenn ; Kröse, B.J.A.: Transferring Knowledge of Activity Recognition across Sensor Networks. In: *Pervasive Computing* Bd. 6030, Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–12654–3, S. 283–300

[KHC10] Kim, Eunju ; Helal, Sumi ; Cook, Diane: Human Activity Recognition and Pattern Discovery. In: *IEEE Pervasive Computing* 9 (2010), Nr. 1, S. 48–53. http://dx.doi.org/10.1109/MPRV.2010.7. – DOI 10.1109/MPRV.2010.7

[KJG+11] Kuehne, H. ; Jhuang, H. ; Garrote, E. ; Poggio, T. ; Serre, T.: HMDB: A large video database for human motion recognition. In: *2011 International Conference on Computer Vision*. Barcelona, Spain : IEEE, 2011, S. 2556–2563

[KKB14] Kwon, Yongjin ; Kang, Kyuchang ; Bae, Changseok: Unsupervised learning for human activity recognition using smartphone sensors. In: *Expert Systems with Applications* 41 (2014), Nr. 14, 6067–6074. http://dx.doi.org/https://doi.org/10.1016/j.eswa.2014.04.037. – DOI https://doi.org/10.1016/j.eswa.2014.04.037. – ISSN 0957–4174

[KLLK10] Khan, A. M. ; Lee, Y.-K. ; Lee, S. Y. ; Kim, T.-S.: Human Activity Recognition via an Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis. In: *2010 5th International Conference on Future Information Technology*. Busan, Korea (South) : IEEE, 2010, S. 1–6

[KNY+14] Krüger, Frank ; Nyolt, Martin ; Yordanova, Kristina ; Hein, Albert ; Kirste, Thomas: Computational state space models for activity and intention recognition. A feasibility study. In: *PloS one* 9 (2014), Nr. 11, S. e109381. http://dx.doi.org/10.1371/journal.pone.0109381. – DOI 10.1371/journal.pone.0109381. – Publisher: Public Library of Science San Francisco, USA

[Kon05] Konar, Amit: *Computational intelligence: principles, techniques and applications*. Berlin, Heidelberg : Springer Science & Business Media, 2005 https://link.springer.com/book/10.1007/b138935. – ISBN 978–3–540–20898–3

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems* Bd. 25, Curran Associates, Inc., 2012

[KSH17] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Communications of the ACM* 60 (2017), Nr. 6, S. 84–90. http://dx.doi.org/10.1145/3065386. – DOI 10.1145/3065386. – ISSN 0001–0782

[KSR+21a] Kirchhof, Michael ; Schmid, Lena ; Reining, Christopher ; Hompel, Michael ten ; Markus, Pauly: pRSL: Interpretable Multi-label Stacking by Learning Probabilistic Rules. In: *Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021.*, PMLR, 2021

[KSR+21b] Kirchhof, Michael ; Schmid, Lena ; Reining, Christopher ; Hompel, Michael ten ; Pauly, Markus: Chances of Interpretable Transfer Learning for Human Activity Recognition in Warehousing. In: *Computational Logistics*, Springer International Publishing, 2021. – ISBN 978–3–030–87672–2, S. 163–177

[KWM11] Kwapisz, Jennifer R. ; Weiss, Gary M. ; Moore, Samuel A.: Activity Recognition Using Cell Phone Accelerometers. In: *ACM SIGKDD Explorations Newsletter* 12 (2011), Nr. 2, 74–82. http://dx.doi.org/10.1145/1964897.1964918. – DOI 10.1145/1964897.1964918. – ISSN 1931–0145

[KY18] Khalid, Muhammad U. ; Yu, Jie: Multi-Modal Three-Stream Network for Action Recognition. In: *24th International Conference on Pattern Recognition (ICPR)*. Beijing, China : IEEE, 2018, S. 3210–3215

[LBBH98] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324. http://dx.doi.org/10.1109/5.726791. – DOI 10.1109/5.726791

[LBOM12] LeCun, Yann A. ; Bottou, Léon ; Orr, Genevieve B. ; Müller, Klaus-Robert: Efficient BackProp. Version: 2012. "https://doi.org/10.1007/978-3-642-35289-8_3". In: Montavon, Grégoire (Hrsg.) ; Orr, Geneviève B. (Hrsg.) ; Müller, Klaus-Robert (Hrsg.): *Neural Networks: Tricks of the Trade: Second Edition*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–35289–8, 9–48

[LC11] Lee, Young-Seol ; Cho, Sung-Bae: Activity Recognition Using Hierarchical Hidden Markov Models on a Smartphone with 3D Accelerometer. In: *International Conference on Hybrid Artificial Intelligence Systems*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–21219–2, S. 460–467

[LDS89] LeCun, Yann ; Denker, John ; Solla, Sara: Optimal Brain Damage. In: *Advances in Neural Information Processing Systems* Bd. 2, 1989, S. 598–605

[LKF10] LeCun, Yann ; Kavukcuoglu, Koray ; Farabet, Clement: Convolutional networks and applications in vision. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, S. 253–256

[LLSL16] Liu, Xi ; Liu, Lei ; Simske, Steven J. ; Liu, Jerry: Human Daily Activity Recognition for Healthcare Using Wearable and Visual Sensing Data. In: *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, 2016, S. 24–31

[LMB14] Loper, Matthew ; Mahmood, Naureen ; Black, Michael J.: MoSh: Motion and Shape Capture from Sparse Markers}. In: *ACM Transactions on Graphics* 33 (2014), Nr. 6, 1–13. http://dx.doi.org/10.1145/2661229.2661273. – DOI 10.1145/2661229.2661273. – ISSN 0730–0301

[LMR+15] Loper, Matthew ; Mahmood, Naureen ; Romero, Javier ; Pons-Moll, Gerard ; Black, Michael J.: SMPL: A Skinned Multi-Person Linear Model. In: *ACM Transactions on Graphics* 34 (2015), Nr. 6, 1–16. http://dx.doi.org/10.1145/2816795.2818013. – DOI 10.1145/2816795.2818013

[LNH09] Lampert, Christoph H. ; Nickisch, Hannes ; Harmeling, Stefan: Learning to detect unseen object classes by between-class attribute transfer. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, S. 951–958

[LNH14] Lampert, Christoph H. ; Nickisch, Hannes ; Harmeling, Stefan: Attribute-Based Classification for Zero-Shot Visual Object Categorization. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), Nr. 3, S. 453–465. http://dx.doi.org/10.1109/TPAMI.2013.140. – DOI 10.1109/T-PAMI.2013.140

[LPLP12] Lara, Oscar D. ; Pérez, Alfredo J. ; Labrador, Miguel A. ; Posada, José D.: Centinela: A human activity recognition system based on acceleration and vital sign data. In: *Pervasive and Mobile Computing* 8 (2012), Nr. 5, S. 717–729. http://dx.doi.org/https://doi.org/10.1016/j.pmcj.2011.06.004. – DOI https://doi.org/10.1016/j.pmcj.2011.06.004. – ISSN 1574–1192

[LPT18] Luvizon, Diogo C. ; Picard, David ; Tabia, Hedi: 2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT, USA, 2018, S. 5137–5146

[LSB+18] Lüdtke, Stefan ; Schröder, Max ; Bader, Sebastian ; Kersting, Kristian ; Kirste, Thomas: Lifted Filtering via Exchangeable Decomposition. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, Juli 2018, 5067–5073

[LSP06] Lazebnik, Svetlana ; Schmid, Cordelia ; Ponce, Jean: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* Bd. 2. New York, NY, USA : IEEE, 2006, S. 2169–2178

[LYA09] Long, Xi ; Yin, Bin ; Aarts, Ronald M.: Single-accelerometer-based daily physical activity classification. In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Minneapolis, MN, USA : IEEE, 2009, S. 6107–6110

[LYC17] Lee, Song-Mi ; Yoon, Sang M. ; Cho, Heeryon: Human activity recognition from accelerometer data using Convolutional Neural Network. In: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. Jeju, Korea (South) : IEEE, 2017. – ISBN 978–1–5090–3015–6, S. 131–134

[Mah20] Mahmud, Saif: Human Activity Recognition from Wearable Sensor Data Using Self-Attention. In: *24th European Conf. on Artificial Intelligence, ECAI 2020* 325 (2020), S. 1332 – 1339. http://dx.doi.org/10.3233/FAIA200236. – DOI 10.3233/FAIA200236

[MHB+16] Margarito, Jenny ; Helaoui, Rim ; Bianchi, Anna M. ; Sartor, Francesco ; Bonomi, Alberto G.: User-Independent Recognition of Sports Activities from a Single Wrist-worn Accelerometer: A Template Matching Based Approach. In: *IEEE Transactions on Biomedical Engineering* 63 (2016), Nr. 4, S. 788–796. http://dx.doi.org/10.1109/TBME.2015.2471094. – DOI 10.1109/TBME.2015.2471094. – ISSN 0018–9294, 1558–2531

[MMD17] Majumder, Sumit ; Mondal, Tapas ; Deen, M. J.: Wearable Sensors for Remote Health Monitoring. In: *Sensors* 17 (2017), Nr. 1. http://dx.doi.org/10.3390/s17010130. – DOI 10.3390/s17010130. – ISSN 1424–8220

[MRBPM17] Marcard, Timo von ; Rosenhahn, Bodo ; Black, Michael ; Pons-Moll, Gerard: Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs. In: *Proceedings of the 38th Annual Conference of the European Association for Computer Graphics (Eurographics)* 36 (2017), Nr. 2, 349–360. https://doi.org/10.1111/cgf.13131. – ISSN 0167–7055

[MSR+17] Münzner, Sebastian ; Schmidt, Philip ; Reiss, Attila ; Hanselmann, Michael ; Stiefelhagen, Rainer ; Dürichen, Robert: CNN-Based Sensor Fusion Techniques for Multimodal Human Activity Recognition. In: *Proceedings of the 2017 ACM International Symposium on Wearable Computers*. New York, NY, USA : Association for Computing Machinery, 2017. – ISBN 978–1–4503–5188–1, S. 158–165

[NDCT17] Nguyen-Dinh, Long-Van ; Calatroni, Alberto ; Tröster, Gerhard: Robust Online Gesture Recognition with Crowdsourced Annotations. Version: 2017. http://link.springer.com/10.1007/978-3-319-57021-1_

18. In: Escalera, Sergio (Hrsg.) ; Guyon, Isabelle (Hrsg.) ; Athitsos, Vassilis (Hrsg.): *Gesture Recognition*. Cham : Springer International Publishing, 2017. – ISBN 978–3–319–57020–4 978–3–319–57021–1, 503–537

[NMRRF23] Nair, Nilah R. ; Moya Rueda, Fernando ; Reining, Christopher ; Fink, Gernot A.: Multi-Channel Time-Series Person and Soft-Biometric Identification. In: Rousseau, Jean-Jacques (Hrsg.) ; Kapralos, Bill (Hrsg.): *Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges*. Cham : Springer Nature Switzerland, 2023. – ISBN 978–3–031–37660–3, S. 256–272

[NTJ18] Nair, Nitin ; Thomas, Chinchu ; Jayagopi, Dinesh B.: Human Activity Recognition Using Temporal Convolutional Network. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. Berlin, Germany : Association for Computing Machinery, 2018. – ISBN 978–1–4503–6487–4

[NYD16] Newell, Alejandro ; Yang, Kaiyu ; Deng, Jia: Stacked Hourglass Networks for Human Pose Estimation. In: *Computer Vision – ECCV 2016*, Springer International Publishing, 2016. – ISBN 978–3–319–46484–8, S. 483–499

[OET+14] Ordóñez, Francisco J. ; Englebienne, Gwenn ; Toledo, Paula de ; Kasteren, Tim van ; Sanchis, Araceli ; Kröse, Ben: In-Home Activity Recognition: Bayesian Inference for Hidden Markov Models}. In: *IEEE Pervasive Computing* 13 (2014), Nr. 3, S. 67–75. http://dx.doi.org/10.1109/MPRV.2014.52. – DOI 10.1109/MPRV.2014.52

[OG09] Ojala, Markus ; Garriga, Gemma C.: Permutation Tests for Studying Classifier Performance. In: *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*. USA : IEEE Computer Society, 2009 (ICDM '09). – ISBN 978–0–7695–3895–2, 908–913

[OMR16] Ordóñez Morales, Francisco J. ; Roggen, Daniel: Deep Convolutional Feature Transfer across Mobile Activity Recognition Domains, Sensor Modalities and Locations. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. Heidelberg, Germany : Association for Computing Machinery, 2016. – ISBN 978–1–4503–4460–9, S. 92–99

[OR16] Ordóñez, Francisco J. ; Roggen, Daniel: Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. In: *Sensors* 16 (2016), Nr. 1, S. 115. http://dx.doi.org/10.3390/s16010115. – DOI 10.3390/s16010115. – ISSN 1424–8220

[PVZ15] Parkhi, Omkar M. ; Vedaldi, Andrea ; Zisserman, Andrew: Deep Face Recognition. In: *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, 2015. – ISBN 1–901725–53–7, 41.1–41.12

[PY10] Pan, Sinno J. ; Yang, Qiang: A Survey on Transfer Learning. In: *Transactions on Knowledge and Data Engineering* Bd. 22, IEEE, 2010 (10), S. 1345–1359

[RC15] Ronao, Charissa A. ; Cho, Sung-Bae: Deep Convolutional Neural Networks for Human Activity Recognition with Smartphone Sensors. In: *International Conference on Neural Information Processing*, Springer, 2015. – ISBN 978–3–319–26561–2, S. 46–53

[RC16] Ronao, Charissa A. ; Cho, Sung-Bae: Human activity recognition with smartphone sensors using deep learning neural networks. In: *Expert Systems with Applications* 59 (2016), 235–244. http://dx.doi.org/https://doi.org/10.1016/j.eswa.2016.04.032. – DOI https://doi.org/10.1016/j.eswa.2016.04.032. – ISSN 0957–4174

[RC17] Ronao, Charissa A. ; Cho, Sung-Bae: Recognizing human activities from smartphone sensors using hierarchical continuous hidden Markov models. In: *International Journal of Distributed Sensor Networks* 13 (2017), Nr. 1. http://dx.doi.org/10.1177/1550147716683687. – DOI 10.1177/1550147716683687

[RCR+10] Roggen, Daniel ; Calatroni, Alberto ; Rossi, Mirco ; Holleczek, Thomas ; Förster, Kilian ; Tröster, Gerhard ; Lukowicz, Paul ; Bannach, David ; Pirkl, Gerald ; Ferscha, Alois ; Doppler, Jakob ; Holzmann, Clemens ; Kurz, Marc ; Holl, Gerald ; Chavarriaga, Ricardo ; Sagha, Hesam ; Bayati, Hamidreza ; Creatura, Marco ; Millàn, José del R.: Collecting complex activity datasets in highly rich networked sensor environments. In: *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, 2010, S. 233–240

[RDS+15] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision* 115 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[Rei12] Reiss, Attila: *PAMAP2 Physical Activity Monitoring Data Set*. https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring. Version: 2012. – The PAMAP2 Physical Activity Monitoring

dataset contains data of 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units and a heart rate monitor.

[Rei21] Reining, Christopher S.: *Attributebasierte Erkennung menschlicher Aktivitäten in industriellen Prozessen am Beispiel der Logistik*. Praxiswissen Service, 2021. – ISBN 978–3–86975–163–4

[RFCT13] Roggen, Daniel ; Förster, Kilian ; Calatroni, Alberto ; Tröster, Gerhard: The adARC pattern analysis architecture for adaptive human activity recognition systems. In: *Journal of Ambient Intelligence and Humanized Computing* 4 (2013), Nr. 2, S. 169–186. http://dx.doi.org/https://doi.org/10.1007/s12652-011-0064-0. – DOI https://doi.org/10.1007/s12652–011–0064–0. – ISSN 1868–5145

[RHGS15] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: *Advances in Neural Information Processing Systems* Bd. 28, 2015

[Roj96] Rojas, Raúl: *Neural Networks: A Systematic Introduction*. Berlin : Springer-Verlag, 1996 https://page.mi.fu-berlin.de/rojas/neural/. – ISBN 978–3–642–61068–4

[RRMF18] Rusakov, Eugen ; Rothacker, Leonard ; Mo, Hyunho ; Fink, Gernot A.: A Probabilistic Retrieval Model for Word Spotting Based on Direct Attribute Prediction. In: *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Niagara Falls, NY, USA : IEEE, 2018, S. 38–43

[RRR18] Ramasamy Ramamurthy, Sreenivasan ; Roy, Nirmalya: Recent trends in machine learning for human activity recognition-A survey. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8 (2018), März, S. e1254. http://dx.doi.org/https://doi.org/10.1002/widm.1254. – DOI https://doi.org/10.1002/widm.1254

[RS12a] Reiss, Attila ; Stricker, Didier: Creating and Benchmarking a New Dataset for Physical Activity Monitoring. In: *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*. New York, NY, USA : Association for Computing Machinery, 2012 (PETRA '12). – ISBN 978–1–4503–1300–1. – event-place: Heraklion, Crete, Greece

[RS12b] Reiss, Attila ; Stricker, Didier: Introducing a New Benchmarked Dataset for Activity Monitoring. In: *2012 16th International Symposium on Wearable Computers*, 2012, S. 108–109

[RSA15] Rippel, Oren ; Snoek, Jasper ; Adams, Ryan P.: Spectral Representations for Convolutional Neural Networks. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada : MIT Press, 2015 (NIPS'15), 2449–2457

[RSMF21] Rusakov, Eugen ; Somel, Turna ; Müller, Gerfrid G. W. ; Fink, Gernot A.: Embedded Attributes for Cuneiform Sign Spotting. In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference*. Lausanne, Switzerland : Springer-Verlag, 2021. – ISBN 978–3–030–86330–2, S. 291–305

[RVBZH17] Ramachandran Venkatapathy, Aswin K. ; Bayhan, Haci ; Zeidler, Felix ; Hompel, Michael ten: Human machine synergies in intra-logistics: Creating a hybrid network for research and technologies. In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. Prague, Czech Republic : IEEE, 2017, S. 1065–1068

[SAEM19] Slim, Salwa O. ; Atia, Ayman ; Elfattah, Marwa M. A. ; M.Mostafa, Mostafa-Sami: Survey on Human Activity Recognition based on Acceleration Data. In: *International Journal of Advanced Computer Science and Applications* 10 (2019), Nr. 3. http://dx.doi.org/10.14569/IJACSA.2019.0100311. – DOI 10.14569/IJACSA.2019.0100311. – Publisher: The Science and Information Organization

[SCS18] Sztyler, Timo ; Civitarese, Gabriele ; Stuckenschmidt, Heiner: Modeling and Reasoning with ProbLog: An Application in Recognizing Complex Activities. In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2018, S. 259–264

[SF16] Sudholt, Sebastian ; Fink, Gernot A.: PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. In: *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China : IEEE, 2016. – ISBN 978–1–5090–0981–7, S. 277–282

[SF18] Sudholt, Sebastian ; Fink, Gernot A.: Attribute CNNs for word spotting in handwritten documents. In: *International Journal on Document Analysis and Recognition (IJDAR)* 21 (2018), September, Nr. 3, 199–218. http://dx.doi.org/10.1007/s10032-018-0295-0. – DOI 10.1007/s10032–018–0295–0

[SFLP17] Schmidt Feris, Rogerio ; Lampert, Christoph ; Parikh, Devi: *Visual Attributes*. 1. Springer International Publishing, 2017 (Advances in Computer Vision and Pattern Recognition). – ISBN 978–3–319–50077–5

[SIVA17] Szegedy, Christian ; Ioffe, Sergey ; Vanhoucke, Vincent ; Alemi, Alexander A.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. San Francisco, California, USA : AAAI Press, 2017, 4278–4284

[SK21] Shavit, Yoli ; Klein, Itzik: Boosting Inertial-Based Human Activity Recognition With Transformers. In: *IEEE Access* 9 (2021), S. 53540–53547. http://dx.doi.org/10.1109/ACCESS.2021.3070646. – DOI 10.1109/ACCESS.2021.3070646

[SKSS14] Sung, J ; Koppula, H ; Selman, B ; Saxena, A: *Cornell activity datasets: CAD-60 & CAD-120*. https://www.re3data.org/repository/r3d100012216. Version: 2014

[SLNW16] Shahroudy, Amir ; Liu, Jun ; Ng, Tian ; Wang, Gang: NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA : IEEE, Juni 2016, S. 1010–1019

[SR12] Siirtola, Pekka ; Röning, Juha: Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data. In: *International Journal of Interactive Multimedia and Artificial Intelligence* 1 (2012), Nr. Special Issue on Distributed Computing and Artificial Intelligence, 38–45. http://dx.doi.org/10.9781/ijimai.2012.155. – DOI 10.9781/ijimai.2012.155. – ISSN 1989–1660

[SSH13] Shoaib, Muhammad ; Scholten, Hans ; Havinga, P.J.M.: Towards Physical Activity Recognition Using Smartphone Sensors. In: *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. Italy : IEEE, 2013. – ISBN 978–1–4799–2482–0 978–1–4799–2481–3, S. 80–87

[STBO17] Scheurer, Sebastian ; Tedesco, Salvatore ; Brown, Kenneth N. ; O'Flynn, Brendan: Human activity recognition for emergency first responders via body-worn inertial sensors. In: *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. Eindhoven, Netherlands, 2017. – ISBN 978–1–5090–6244–7, S. 5–8

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Two-Stream Convolutional Networks for Action Recognition in Videos. In: *Proceedings of the 27th Interna-*

tional Conference on Neural Information Processing Systems Bd. 1. Montreal, Canada : MIT Press, 2014, 568–576

[TDF+18] Twomey, Niall ; Diethe, Tom ; Fafoutis, Xenofon ; Elsts, Atis ; McConville, Ryan ; Flach, Peter ; Craddock, Ian: A Comprehensive Study of Activity Recognition Using Accelerometers. In: *Informatics* 5 (2018), Nr. 2, 27. http://dx.doi.org/10.3390/informatics5020027. – DOI 10.3390/informatics5020027. – ISSN 2227–9709

[TLLY18] Tao, Wenjin ; Lai, Ze-Hao ; Leu, Ming C. ; Yin, Zhaozheng: Worker Activity Recognition in Smart Manufacturing Using IMU and sEMG Signals with Convolutional Neural Networks. In: *Procedia Manufacturing* 26 (2018), 1159–1166. http://dx.doi.org/https://doi.org/10.1016/j.promfg.2018.07.152. – DOI https://doi.org/10.1016/j.promfg.2018.07.152. – ISSN 2351–9789

[VEA+15] Vinciarelli, Alessandro ; Esposito, Anna ; André, Elisabeth ; Bonin, Francesca ; Chetouani, Mohamed ; Cohn, Jeffrey F. ; Cristani, Marco ; Fuhrmann, Ferdinand ; Gilmartin, Elmer ; Hammal, Zakia ; Heylen, Dirk ; Kaiser, Rene ; Koutsombogera, Maria ; Potamianos, Alexandros ; Renals, Steve ; Riccardi, Giuseppe ; Salah, Albert A.: Open Challenges in Modelling, Analysis and Synthesis of Human Behaviour in Human–Human and Human–Machine Interactions. In: *Cognitive Computation* 7 (2015), Nr. 4, S. 397–413. http://dx.doi.org/https://doi.org/10.1007/s12559-015-9326-z. – DOI https://doi.org/10.1007/s12559–015–9326–z. – ISSN 1866–9964

[VFD+17] Vital, Jessica P. M. ; Faria, Diego R. ; Dias, Gonçalo ; Couceiro, Micael S. ; Coutinho, Fernanda ; Ferreira, Nuno M. F.: Combining discriminative spatiotemporal features for daily life activity recognition using wearable motion sensing suit. In: *Pattern Analysis and Applications}* 20 (2017), 1179–1194. http://dx.doi.org/10.1007/s10044-016-0558-7. – DOI 10.1007/s10044–016–0558–7. – ISSN 1433–7541, 1433–755X

[WF22] Wolf, Fabian ; Fink, Gernot A.: Self-Training of Handwritten Word Recognition for Synthetic-to-Real Adaptation. In: *Proc. Int. Conf. on Pattern Recognition*. Montreal, Canada, 2022

[WGT+11] Wang, Liang ; Gu, Tao ; Tao, Xianping ; Chen, Hanhua ; Lu, Jian: Recognizing multi-user activities using wearable sensors in a smart home. In: *Pervasive and Mobile Computing* 7 (2011), Nr. 3, 287–

298. http://dx.doi.org/https://doi.org/10.1016/j.pmcj.2010.11.008. – DOI https://doi.org/10.1016/j.pmcj.2010.11.008. – ISSN 1574–1192

[WGWH18] Wolff, Johann P. ; Grützmacher, Florian ; Wellnitz, Arne ; Haubelt, Christian: Activity Recognition Using Head Worn Inertial Sensors. In: *Proceedings of the 5th International Workshop on Sensor-Based Activity Recognition and Interaction*. Berlin, Germany : Association for Computing Machinery, 2018. – ISBN 978–1–4503–6487–4

[XHF+18] Xi, Rui ; Hou, Mengshu ; Fu, Mingsheng ; Qu, Hong ; Liu, Daibo: Deep Dilated Convolution on Multimodality Time Series for Human Activity Recognition. In: *International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro, Brazil : IEEE, 2018, S. 1–8

[XLSA19] Xian, Yongqin ; Lampert, Christoph H. ; Schiele, Bernt ; Akata, Zeynep: Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41 (2019), Nr. 9, S. 2251–2265. http://dx.doi.org/10.1109/TPAMI.2018.2857768. – DOI 10.1109/TPAMI.2018.2857768. – ISSN 1939–3539

[YLSR18] Yao, Rui ; Lin, Guosheng ; Shi, Qinfeng ; Ranasinghe, Damith C.: Efficient Dense Labelling of Human Activity Sequences from Wearables using Fully Convolutional Networks. In: *Pattern Recognition* 78 (2018), S. 252–266. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2017.12.024. – DOI https://doi.org/10.1016/j.patcog.2017.12.024. – ISSN 0031–3203

[YNS+15] Yang, Jian B. ; Nguyen, Minh N. ; San, Phyo P. ; Li, Xiao L. ; Krishnaswamy, Shonali: Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. Buenos Aires, Argentina : AAAI Press, 2015 (IJCAI'15). – ISBN 978–1–57735–738–4, 3995–4001

[YPS+18] Yordanova, Kristina ; Paiement, Adeline ; Schröder, Max ; Tonkin, Emma ; Woznowski, Przemyslaw ; Olsson, Carl M. ; Rafferty, Joseph ; Sztyler, Timo: Challenges in Annotation of useR Data for UbiquitOUs Systems: Results from the 1st ARDUOUS Workshop. In: *arXiv:1803.05843 [cs]* (2018)

[ZCS19] Zhu, Qingchang ; Chen, Zhenghua ; Soh, Yeng C.: A Novel Semisupervised Deep Learning Method for Human Activity Recognition. In: *IEEE Transactions on Industrial Informatics* 15 (2019), Nr. 7, 3821–3830. http://dx.doi.org/10.1109/TII.2018.2889315. – DOI 10.1109/TII.2018.2889315

[Zha12] Zhang, Zhengyou: Microsoft Kinect Sensor and Its Effect. In: *IEEE MultiMedia* 19 (2012), Nr. 2, S. 4–10. http://dx.doi.org/10.1109/MMUL.2012.24. – DOI 10.1109/MMUL.2012.24

[ZHY09] Zheng, Vincent W. ; Hu, Derek H. ; Yang, Qiang: Cross-Domain Activity Recognition. In: *Proceedings of the 11th International Conference on Ubiquitous Computing*. Orlando, Florida, USA : Association for Computing Machinery, 2009 (UbiComp '09). – ISBN 978–1–60558–431–7, 61–70

[ZJC17] Zheng, Jingjing ; Jiang, Zhuolin ; Chellappa, Rama: Submodular Attribute Selection for Visual Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2017), Nr. 11, S. 2242–2255. http://dx.doi.org/10.1109/TPAMI.2016.2636827. – DOI 10.1109/TPAMI.2016.2636827

[ZLT17] Zhou, Wengang ; Li, Houqiang ; Tian, Qi: Recent Advance in Content-based Image Retrieval: A Literature Survey. (2017). https://arxiv.org/abs/1706.06064

[ZNY+14] Zeng, Ming ; Nguyen, Le T. ; Yu, Bo ; Mengshoel, Ole J. ; Zhu, Jiang ; Wu, Pang ; Zhang, Joy: Convolutional Neural Networks for human activity recognition using mobile sensors. In: *6th International Conference on Mobile Computing, Applications and Services*, IEEE, 2014, S. 197–205

[ZO18] Zhao, Junqi ; Obonyo, Esther: Towards a Data-Driven Approach to Injury Prevention in Construction. In: *Advanced Computing Strategies for Engineering*. Cham : Springer International Publishing, 2018. – ISBN 978–3–319–91635–4, S. 385–411

[ZSMP15] Zainudin, M.N.Shah ; Sulaiman, Md N. ; Mustapha, Norwati ; Perumal, Thinagaran: Activity recognition based on accelerometer sensor using combinational classifiers. In: *Activity recognition based on accelerometer sensor using combinational classifiers*. Melaka, Malaysia : IEEE, 2015. – ISBN 978–1–4673–9434–5, S. 68–73

[ZXZ+17] Zhang, Zhendong ; Xu, Dongfang ; Zhou, Zhihao ; Mai, Jingeng ; He, Zhongkai ; Wang, Qining: IMU-based underwater sensing system for swimming stroke classification and motion analysis. In: *2017 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 2017, S. 268–272

ACRONYMS

Adaptable Attr-IMU-tCNN Adaptable Attribute IMU-Temporal Convolutional Neural Network.
ADL Activities of Daily Living.
ANN Artificial Neural Network.
AP Average Precision.
Attr-based M-HAR Attribute-based Multi-channel Time-Series HAR.
Attr-based Transfer Learning for M-HAR Attribute-based Transfer Learning for Multi-channel Time-Series HAR.
Attr-IMU-tCNN Attribute IMU-Temporal Convolutional Neural Network.
Attr-tCNN Attribute Temporal Convolutional Neural Network.

B-LSTM Bidirectional-LSTM.
BagFR Bag of Features Representation.
BCE Binary Cross-Entropy.
BGD Batch Gradient Descent.
BPM Business Process Model.
BPTT BackPropagation Through Time.

CAD-60 Cornell Activity Dataset.
CNN Convolutional Neural Network.
CRF Conditional Random Field.
CSSM Computational State-Space Model.

DAP Direct Attribute Prediction.
DBMM Dynamic Bayesian Mixture Model.
DDA Data-Driven Attribute.
DeepConvLSTM Deep Convolutional LSTM.
DFT Discrete Fourier Transform.
DIP Deep Inertial Poser.
DL Deep Learning.
DNN Deep Neural Network.
DT Decision Tree.
DTW Dynamic Time Warping.

EA Evolutionary Algorithm.
ED Euclidean Distance.

FC Fully Connected Layer.
FCN Fully Convolutional Layer.
FLW Lehrstuhl für Förder- und Lagerwesen.
FN False Negative.
FP False Positive.

GD Gradient Descent.
GDAP Generalized Direct Attribute Prediction.
GMM Gaussian Mixture Model.

HAR Human Activity Recognition.
HARetr Human Activity Retrieval.
HLA Human-Labeled Attribute.
HMDB Human Motion DataBase.
HMM Hidden Markov Model.

i.i.d. Independent and Identically Distributed.
IAP Indirect Attribute Prediction.
IDFT Inverse Discrete Fourier Transform.
IMU-tCNN IMU Temporal Convolutional Neural Network.
IS Transfer Learning Informed Supervised Transfer Learning.
ITL Inductive Transfer Learning.
IU Transfer Learning Informed Unsupervised Transfer Learning.

J-HMDB Joints Human Motion DataBase.
JB Joint Boosting.

KDA Kernel Discriminant Analysis.
KNN K-Nearest Neighbour.

LA Left Arm.
LARa Logistic Activity Recognition Challenge.
LARa-M LARa-MoCap.
LARa-Mb LARa-Mbientlab.
LARa-MM LARa-MotionMiners.
LARa-SOBD LARa-Synthetic OBD.
LDA Linear Discriminant Analysis.

LL Left Leg.
LR Logistic Regression.
LSM Least Squares Method.
LSTM Long short-term Memory.
LTCOV Leave-Two-Class-Out Cross Validation.

M-HAR multi-channel time-series Human Activity Recognition.
marker-based MoCap marker-based Motion Capturing System.
MLP Multi-Layer Perceptron.
MM MotionMiners Dataset.
multi-channel Time-Series multi-channel Time-Series.
multi-channel time-series Space multi-channel time-series Space.

NB Naïve Bayes.
NN Nearest Neighbour.
NNA Nearest Neighbour Approximation.
NT Neck or Torso.
NTU RGB+D A large Scale RGB+Depth Dataset for 3D HAR from the Nanyang Technological University.

OBD On-body Device.
OPD Order-Picking Dataset.
Opp Opportunity Challenge Dataset.

PAF Part Affinity Field.
Pamap2 Physical Activity Monitoring Data Set.
PCA Principal Component Analysis.
PDDL Planning Domain Definition Language.
PDF Probability Density Function.
PHOC Pyramidal Histogram of Characters.
PHOCNet Pyramidal Histogram of Characters Network.
pose-based HAR pose-based Human Activity Recognition.
PRM Probabilistic Retrieval Model.
pRSL Probabilistic Rule Stacking Learner.

QbA Query-by-Attribute.
QbAR Query-by-Attribute Representation.
QbC Query-by-Class.
QbE Query-by-Example.

QbS Query-by-String.
QbX Query-by-eXpression.
QDA Quadratic Discriminant Analysis.
QPP Quadratic Programming Problem.

RA Right Arm.
ReLU Rectified Linear Unit.
RF Random Forest.
RFE Recursive Feature Elimination.
RL Right Leg.
RNN Recurrent Neural Network.
RP Random Projection.

SBFS Sequential Backward Feature Selection.
SBGD Stochastic Batch Gradient Descent.
SD Standard Deviation.
sEMG Surface Electromyography Sensors.
Sensor-based HAR Sensor-based Human Activity Recognition.
SFFS Sequential Forward Feature Selection.
SGD Stochastic Gradient Descent.
SLP Single-Layer Perceptron.
SMGD Stochastic Minibatch Gradient Descent.
SMPL Skinned Multi-Person Linear Model.
SOBD Synthetic On-body Device.
SPP Spatial Pyramid Pooling.
SPP Layer SPP Layer.
STIP Spatio-Temporal Interest Points.
SVM Support Vector Machine.

TBCE Total Binary Cross-Entropy.
TCCE Total Categorical Cross-Entropy.
tCNN Temporal Convolutional Neural Network.
TL Transfer Learning.
TM Template Maching.
TN True Negative.
TP True Positive.
TPP Temporal Pyramid Pooling.
Transfer Learning for HAR Transfer Learning for Human Activity Recognition.
Transfer Learning for M-HAR Transfer Learning for Multi-channel Time-Series HAR.

TSSE Total Sum Squared Error.
TTL Transductive Transfer Learning.

US Transfer Learning Uninformed Supervised Transfer Learning.
UU Transfer Learning Uninformed Unsupervised Transfer Learning.

video-based HAR video-based Human Activity Recognition.

Zero-Shot HAR Zero-Shot Human Activity Recognition.
Zero-Shot M-HAR Zero-Shot Multi-channel Time-Series HAR.