POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Deep learning and domain adaptation acceleration-based techniques for human activity recognition in a hospital setting

Laurea Magistrale in Biomedical Engineering - Ingegneria Biomedica

Author: Lorenzo Principi

Advisor: Prof. Enrico Gianluca Caiani

Academic year: 2020-2021

1. Introduction

This study was based on a previous project focusing on the development of a Deep Neural Network (DNN) to recognize activities carried out by inpatients [1]. Such setting entailed a data collection on 20 healthy subject simulating patient-like activities and following a specific protocol. For this reason, the reference study in [1] has been defined "Simulated Hospital Study" (SHS).
The aim of this work is to validate the SHS pre-trained DNN SHS on a novel Real Hospital Study (RHS). More in detail, the research question addressed in this project intends to investigate to what extent previously-learned SHS activity features representations could be transferred to the target RHS.
Table 1 summarizes the main differences between the two above-mentioned settings.

	SHS	RHS
Subjects	20	12
Splitting	Random	(Partly) random
Protocol	Yes	No
Session time	1 hour	≈24 hours
DNN classes	6	5
Majority labels	Dynamic	Static

Table 1: Comparison between simulated and real hospital study

2. Methods

2.1. Data collection

The RHS dataset consisted of 12 general ward patients (7 males and 5 females, body mass index (BMI): $30.13 \pm 9.88 \frac{kg}{m^2}$) from a clinic located in the United States.
The enrollment occurred on a voluntary basis and without any exclusion criteria. An informed consent was dispensed to and signed by the inpatients interested in taking part to the study. This project has been approved by an internal

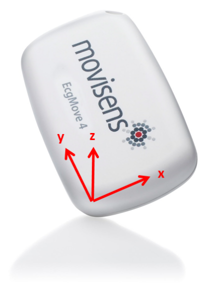


Figure 1: ECGMove4 axes orientations

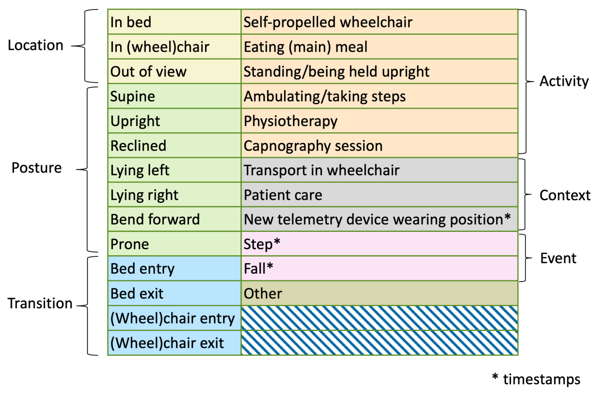


Figure 2: Activity label set

committee at Philips in co-operation with the clinic counterpart.
A pair of movisens® ECGMove4 wearables were attached to the left-side chest in two separate locations: between the first and second rib and in correspondence of the twelfth one. Such locations were termed as *Upper* and *Lower*. Referring to Fig. 1, ECGMove4 x-, y- and z- axes corresponded to caudo-cranial, left-right and postero-anterior anatomical directions, respectively. These devices captured different types of signals (i.e. ECG, temperature, angular rate, ...). Nevertheless, the 3D acceleration only has been selected since the SHS DNN was trained on such data. The latter signal has been logged at 100 Hz in the range of ± 16 g($1 \text{ g} = 9.8 \text{ m/s}^2$).
All data acquisition sessions were recorded with a video camera placed within patient's hospital room and lasted for ≈24 hours. Video clips were examined by a single operator to make annotations on activities and postures.
The acquisition setup was designed to minimally interfere with the standard clinical workflow. Thus, all the activities performed by patients were framed in a free-living context. After a preliminary inspection of the video recordings, a set of 28 labels has been determined (see Fig. 2).

2.2. Data preprocessing

A precondition was to adopt the same SHS preprocessing pipeline and 3D acceleration properties to correctly validate the legacy DNN. The first step consisted in matching the ECGMove4 accelerometer reference system (depicted in Fig. 1) with SHS sensor one. The same applied to the acceleration range, data normalization and windowing strategy (i.e. 6-seconds with 50% overlapping). Data segments were labelled via majority voting and discarded if they contained more than 50% of unlabelled samples. Concerning sampling frequency, ECGMove4 wearables measured acceleration at 64 Hz whereas SHS devices at 100 Hz. We found that downsampling acceleration signals did not remarkably impair the DNN learning. Thus, the ECGMove4 sampling rate was decreased to 16 Hz.
Annotated activities and acceleration samples were synchronized with a specific procedure. The latter consisted in generating acceleration artifacts in front of the camera by shaking both ECGMove4 sensors at the beginning and just before the end of each recording session.
The activities represented in Fig. 2 were grouped in 5 DNN activity classes: Lying, Reclined, Upright, Walking and (self-propelled) Wheelchair.
As reported in table 1, the RHS dataset splitting procedure was partially randomized. In fact, 5 patients were a-priori blindly designated as *test* patients. The remaining 7 ones were randomly split into *train* and *validation* subsets on a per-patient basis.

2.3. Deep Neural Network

Differently from the original intent, we were unable to validate the same DNN model architecture used in SHS. The main culprit lied in a quite remarkable DNN performance instability (i.e. sensitivity to parameter initialization). In addition, the acceleration samples on which the DNN was trained were acquired from a quite different chest location from both *Upper* and *Lower* ECGMove4 devices. Nevertheless, one of the additional sensors employed in SHS almost perfectly matched with *Upper* ECGMove4 in terms of anatomical location. Hence, the final DNN baseline consisted of a slight variation of the SHS DNN architecture (with codename *ccpd2*) trained on 16 Hz 3D acceleration samples acquired from the left-side upper-chest accelerometer. Fig. 3 shows the *ccpd2* baseline DNN architecture.

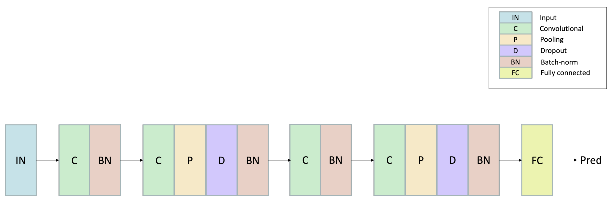


Figure 3: Baseline deep neural network structure

Four 1D convolutional layers (filters: 4, 16, 16 and 32 with kernel sizes: 23, 10, 7 and 7, respectively) performed automatic feature extraction. The *bottleneck features* were then flattened and fed to a fully-connected layer (512 neurons). All the convolutional layers used a zero-padding strategy and a ReLu activation function. Plus, each convolutional block ended with a batch-normalization layer. The dropout layers (ratio: 30%) were included within the second and fourth convolutional block. The last dense layer was composed of 5 neuron (matching the number of DNN activity classes) and activated by a softmax function.
The *ccpd2* model was trained for 100 epochs, using Adam optimizer and with a batch size of 100.

2.4. CPA

CPA stands for *Calibration*, *Posture* and *Activity* that represent the constituting modules. It consists of a rule-based human activity recognition algorithm under Philips intellectual property. This approach is characterized by a strong determinism to predict postures. In fact, we deemed interesting to compare the performance between models with opposite intrinsic natures (i.e. determinism and *black-box* for CPA and DNN, respectively).
The *Calibration* module aimed at aligning the ECGMove4 device frame with the anatomical directions. The main assumption of this procedure is that the posture during Walking should be *Upright*. In addition, multiple consecutive Walking segments were necessary to determine the reference and device vertical directions. If the *Calibration* was successful, the x-axis tilt angle (α) between the two directions would be computed and used to generate a rotation matrix (R_{orient}) to finally align the reference systems.
The *Posture* routine required a set of predetermined reference posture vectors (p_i). Those were rotated by the R_{orient} rotation matrix and compared to the measured acceleration values for each axis. The output posture was selected for the corresponding i_{th} index that minimized the distance between the reference (p_i) and rotated ($\tilde{p_i}$) posture vectors. The CPA algorithm recognized 7 different postures: `Upright`, `Reclined`, `Lie supine`, `Prone`, `Lie on left side`, `Lie on right side`, `Upside down`.
The *Activity* module has not been used since the intensity at which RHS activities were performed was outside the scope of this study.

2.5. Hybrid ensemble method

A further method based on a ensemble machine learning technique was designed. Its aim was to capitalise on the advantages offered by the structural differences between the CPA and *ccpd2* DNN by combining their predictions. Those were considered as *Base-models* (or Level-0 model) providing their output to a *Meta-model* (or Level-1 model) in a *Stacked generalization* framework. In practical terms, the Level-1 model was fed with the DNN softmax-ed probability outputs (one for each of the 5 DNN activity class) and CPA *deltas* (i.e. distances between rotated and reference posture vectors).
A *Logistic regressor* was identified as a suitable *Meta-learner* model. In this way we could easily interpret the contribution of each *Base-model* output (treated as features) to determine the final activity class prediction.

2.6. Deep transfer learning

It is worth to recall that the research question related to this work concerned to assess the degree of feature representations transferability between source SHS and RHS. This can be achieved by transferring kernel parameters contained in a *ccpd2* convolutional layer via *transfer learning* techniques.

2.6.1 Canonical techniques

The first set of tests entailed a standard *transfer learning* approach. Although several configurations were tested, all of them shared the same basic procedure.

1. Retraining the *ccpd2* baseline model on SHS data.
2. Freezing all layers.
3. Detaching and replacing the final *ccpd2* fully-connected classification layer with a custom one (or block).
4. Retraining the tweaked *ccpd2* DNN on target hospital data.

At the end of step 3 the pre-trained *ccpd2* is defined as *backbone*. Step 4 consisted of the *fine-tuning* procedure. Within this stage, a number of frozen *backbone* layers were unfrozen to adapt the model weights to the novel information brought by hospital activity data.

2.6.2 Kernel transfer analysis

This analysis aimed at assessing the feasibility of transferring feature representation in different contexts. At the end of this investigation we were able to discriminate which kernels contained either general or domain-specific feature representations. The procedure carried out in this stage followed the footsteps of a previous study focusing on this matter [3]. Two different scenarios have been addressed:

1. knowledge transfer between users within the same domain
2. knowledge transfer between different domains.

Case 1 referred to different patients within RHS and case 2 to cross-domain transfer from SHS to RHS.
The implementation of this approach consisted of making a replica (*transferred model*) of the original *ccp2* DNN (*source model*). Next, a number of pre-trained *source model* kernel parameters were copied to the *transferred model*. Eventually, the latter was *fine-tuned* on a portion of the *target* subset reserved for *adaptation*. Different combinations of transferred layers and percentage of revealed *adaptation* subset were tested.

2.7. Domain-Adversarial Neural Networks

Since both the SHS and RHS relate to the same activity recognition task, we leveraged this aspect to implement a specific setting of knowledge transfer: *domain adaptation* (according to the definitions outlined in [4]).
Since this project revolved around DNN models, this task has been carried out via Domain-Adversarial Neural Networks (DANNs) [2]. Those can be easily implemented by expanding any pre-existing DNN architecture and are trained via standard *backpropagation*. Fig. 4 represents a general DANN scheme and its main components: *feature extractor*, *label predictor* and *domain classifier*.

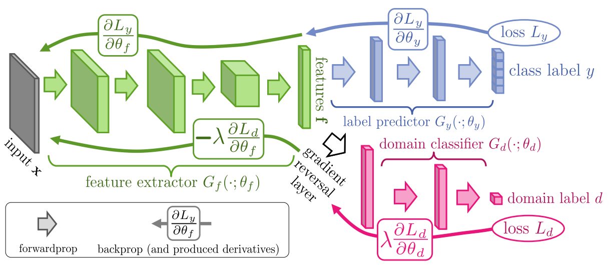

$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
input x
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
$\frac{\partial L_y}{\partial \theta_f}$
backprop (and produced derivatives)

Figure 4: Domain-adversarial neural network structure

Both the *feature extractor* and *label predictor* blocks coincide with the *backbone* and *classification block* of the *ccpd2* architecture represented in Fig. 3.
The novelties are represented by the *domain classifier* and the Gradient Reversal Layer (GRL). This module is devoted to discriminate if the extracted features either belong to the SHS *source* or RHS *target* domain. This is achieved by maximizing the binary cross-entropy loss. This block was added to *ccpd2* by means of two fully-connected layers (512 and 256 neurons, respectively) before the single-neuron output layer.
The GRL acts as a two-way activation layer for the gradients computed during the *backpropagation*. The upstream gradient is simply transferred from the *features extractor* to the *domain classifier* without any changes. The downstream one is instead changed in sign. Both of them are scaled by a λ *domain adaptation* parameter

varying according to the following equation:

$$\lambda_p = \frac{2}{1 + exp(-\gamma \cdot p)} - 1 \tag{1}$$

where p indicates the training progress (from 0 to 1) and $\gamma = 10$.
Concerning the training procedure, a custom batch generation system needed to be implemented. Each batch contained 128 elements equally split in SHS *source* and RHS *target* samples. By default, the DANN are framed in a fully *unsupervised learning* setting. Thus, the *target* samples are considered unlabelled and fed along with the labeled source samples to the *domain classifier*. On the other hand, the *label predictor* is trained on labeled SHS *source* activity data only. However, the training routine can be accustomed to a *semi-supervised* learning scenario by revealing portions of the RHS *target* activity sample to the *label predictor*. Both the *unsupervised* and *semi-supervised* were implemented and tested within this work
To summarize, DANNs promote the emergence of features that are both discriminative for the main predictive learning task and indiscriminate to the shift between domains.

3. Results

3.1. Data analysis

The RHS mostly featured *static* postures. By way of example, Reclined and Upright jointly accounted for ≈60% of the overall activities distribution. As a consequence, all the DNN models were trained using a balanced batch generator.
A related analysis consisted in the quantification of *Inactive* labels compared to *Active* labels. As expected, the former accounted for 85.48±5.76% (*mean* ± *std* over all RHS patients).
Eventually, we found that patients spent ≈ 80% (median value) of their time `in bed`.

3.2. Deep neural network

Fig. 5 represents the confusion matrix associated with the performance obtained by validating the SHS pre-trained *ccpd2* on whole RHS dataset.

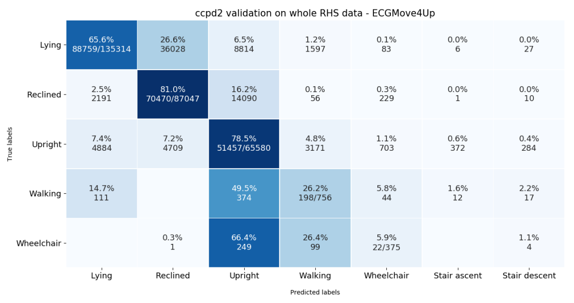


Figure 5: Pre-trained ccpd2 validation on real hospital study

Although the detection accuracy obtained for the *static* postures was fairly high, both the Walking and Wheelchair were poorly recognized.
When retraining the *ccpd2* on RHS patients and validating it on the 5 pre-defined test inpatients, we encountered two contrasting effect. On one side, a remarkable drop in Reclined and *Upright* detection ($\approx -40\%$ for both classes). On the other hand, a noticeable increase in the recognition of Walking activities ($\approx +30\%$). The percentage changes are referred to to the scores shown in Fig. 5. This also holds for the results presented in subsection 3.4.
Within this context, a Leave-One-Subject-Out (L1SO) cross-validation has been implemented to assess *ccpd2* performance heterogeneity between RHS patients. The obtained weighted F1-score values ranged from 0.2724 to 0.6964, implying a wide-spread variability.

3.3. CPA

We verified that, due to the lack of Walking samples, the CPA *Calibration* never successfully occurred across all RHS patients. Thus, we were compelled to pick an a-priori and constant α tilt angle able to provide acceptable performance for the whole RHS dataset.
$\alpha = -18°$ has been identified as the optimal tilt angle providing acceptable detection accuracy with an overall weighted F1-score of 0.65.

3.4. Hybrid ensemble method

The *Stacking* ensemble allowed to increase the detection accuracy of Lying and Upright by $\approx +20\%$ and $\approx +35\%$, respectively. On the contrary, Reclined recognition experienced a consisting degradation of $\approx -17\%$.
By examining the *Logistic regressor* regression

coefficient heatmap, we found that the *Stacking* was able to grasp complexity relationship between postures. For instance, CPA *Lie left/right deltas* were a remarkable indicator to predict a DNN Lying posture. In general, the Level-1 model trusted the CPA outputs rather more than the DNN ones.

3.5. Deep transfer learning

As mentioned in subsection 2.6, several *transfer learning* settings were tested during this stage. However, we did not experience substantial improvement with respect to the *ccpd2* baseline results reported in Fig. 5. In fact, we found that both the accuracy and loss scores limitedly improved during the *fine-tuning* process. Such behaviour has been experienced on all tested *transfer learning* configurations. However, the best results (*Precision*: 0.67, *Recall*: 0.66, *F1-score*: 0.66) have been obtained when unfreezing all *ccpd2* convolutional layers and a single 128-neurons fully-connected layer.
Concerning the *kernel transfer analysis*, two main findings have been obtained at the end of the "same domain" setting. The first was that also including a small percentage (as low as 25%) of the left-out *adaptation subset* the overall F1-score sharply rose to ≈0.9. The additional remark was strictly linked to a statistical analysis based on the *p-value* associated to Wilcoxon paired-test between F1-score populations in different *kernel transfer* settings (see [3]). In particular, we found that features extracted up to the second convolutional layer were patient-independent within this setting.
On the other hand, the "different domain" scenario featured a high variability in F1-scores due to the high RHS within-population and cross-domain (i.e. SHS to RHS) heterogeneity. Thus, no clear findings could be obtained at the end of the investigation.

3.6. Domain-adversarial neural networks

At first, the DANN was evaluated in a *unsupervised learning* scenario. Thus, the *label predictor* module was fed with SHS activity data only. After testing several DANN settings, we concluded that the performance obtained in this scenario was not higher than what achieved with previously techniques.

Hence, we decided to switch to a *semi-supervised* learning setting by revealing a portion of labeled data from a single RHS patient. The best results have been obtained by feeding 50% of P03 activity data to the *label predictor*.
Following this path we finally achieved the overall optimal DANN model. This configuration entailed revealing 10% activity data from 5 different RHS patients. This DANN setting outperformed all the models tested throughout this project with a weighted F1-score of 0.73. In addition, this optimal DANN achieved the best (i.e. lowest) cross-entropy score on *validation* subset while training. This finding corroborated the robustness of such model.
Eventually, a further assessment of the identified optimal DANN was performed by evaluating the *feature alignment* success. After the *domain adaptation* process, the features representation associated to activity data from both SHS and RHS should be more similar. This has been visually inspected by relying on ""t-Distributed Stochastic Neighbor Embedding" (t-SNE)" [5]. Fig. 6 shows a t-SNE representation of SHS and RHS features emphasizing a fairly good alignment on Walking speeds up to 1.5 km/h.

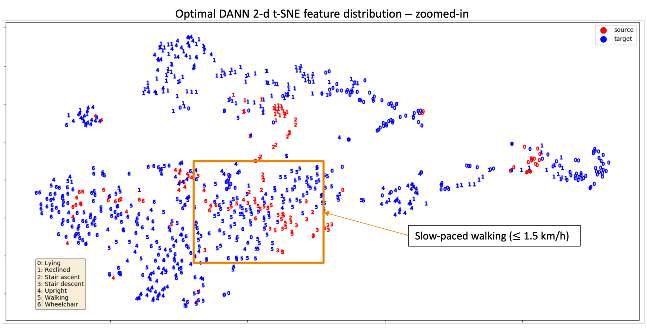


Figure 6: Feature alignment between simulated and real hospital study after *domain adaptation*

4. Conclusion

The inherent discrepancies between SHS and RHS both in terms of population and activity patterns affected the performance of all activity recognition models implemented in this work. This can be proved by examining the results obtained when validating the SHS pre-trained *ccpd2* on RHS data (see Fig. 5). The quite poor performance obtained on Walking is justified by the irregular steps taken by RHS patients while moving around the room. The same

applies to RHS Wheelchair class that usually entails a very slow self-propelling. In addition, the very high *ccpd2* performance variability when performing L1SO cross-validation indicated that RHS featured quite diverse activity patterns.
For this reason, we considered using a more deterministic approach like CPA. However, the main issue encountered by using this algorithm was the inability of automatically calibrating the ECGMove4 sensor due to the lack of Walking samples. Nevertheless, the *Stacking* hybrid ensemble method consistently relied on CPA outputs. This supported the hypothesis that the intrinsic determinism of the CPA algorithm could result beneficial to possibly mitigate the effect of domain shift between SHS and RHS.
At this concern, the results obtained from both the *transfer learning* techniques and *kernel transfer analysis* contributed to directly address the research question. In fact, we found that SHS previously-learned knowledge could be **limitedly** transferred to RHS setting. This behaviour still holds if adopting standard *fine-tuning* techniques.
Thus, a more refined approach relying on the task similarity between SHS and RHS was identified in *domain adaptation* and DANNs models. In particular, what we found at the end of "same-domain" *kernel transfer analysis* was confirmed at this stage. Indeed, providing small chunks of RHS labeled activity data from multiple patients during DANN training procedure improved its overall performance. The latter featured the best trade-off between annotation burden and performance (0.73 weighted F1-score).
As future works, we recommend to devise an alternative CPA *Calibration* routine based on a different activity trigger (e.g. sit-to-stand event). Then, the optimal DANN identified in this project should be validated in a similar future scenario by partially annotating subjects activities in a spot-check fashion.

References

[1] Esther Fridriksdottir and Alberto G Bonomi. Accelerometer-based human activity recognition for patient monitoring using a deep neural network. *Sensors*, 20(22):6424, 2020.

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[3] Francisco Javier Ordóñez Morales and Daniel Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 92–99, 2016.

[4] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[5] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Deep learning and domain adaptation acceleration-based techniques for human activity recognition in a hospital setting

Tesi di Laurea Magistrale in
Biomedical Engineering - Ingegneria Biomedica

Author: **Lorenzo Principi**

Student ID: 10709887
Advisor: Prof. Enrico Gianluca Caiani
Academic Year: 2020-21

Abstract

Human activity recognition via wearables constitutes a pervasive and unobtrusive approach to perform patient monitoring. The first goal of this project was to validate a deep neural network (DNN) devoted to classify activities carried out by healthy patients in a Simulated Hospital Setting (SHS). The robustness of this model was assessed by evaluating it on free-living activities performed by 12 hospitalized patients. 3D acceleration activity data were acquired by a pair of wearables attached on two separate chest locations. The baseline DNN adopted in this work slightly diverged from the SHS one and was able to recognize 5 different hospital activities: Lying, Reclined, Upright, Walking and (self-propelled) Wheelchair. Plus, a rule-based algorithm was evaluated as a standalone and in combination with the DNN activity class probability outputs within a *Stacked generalization* ensemble method. The above-mentioned methods performed quite modestly and suggested to adopt *knowledge transfer* approaches. Thus, this study addressed a specific research question: to what extent the previously-learned knowledge from *source* SHS activity data could be transferred to *target* real hospital setting (RHS)? The first set of test entailed conventional *transfer learning* techniques. Then, a more refined *domain adaptation* approach was implemented via domain-adversarial neural networks (DANNs). By testing such architectures both in a *unsupervised* and *semi-supervised* learning setting, the optimal configuration (0.73 weighted F1-score) was identified within the latter scenario by revealing small portions of activity data from multiple RHS patients. In conclusion, the RHS dataset resulted quite diverse and different from SHS. Hence, the task similarity between SHS and RHS was leveraged to implement *knowledge transfer* via DANNs. An in-depth *kernel transfer analysis* demonstrated that the cross-domain transferability of convolutional layers was quite limited. However, providing small hints of target RHS samples could substantially improve the overall activity recognition performance. This proposed DANN methodology might be used and validated in future tasks by partially annotating target domain activity in a sport-check fashion. This may remarkably unburden the annotation process by preserving fairly good activity recognition performance.

Keywords: deep learning, domain adaptation, activity recognition, neural networks

Abstract in lingua italiana

Il riconoscimento delle attività motorie attraverso sensori indossabili rappresenta un approccio non-invasivo per il monitoraggio di pazienti ospedalizzati. L'obiettivo iniziale di questo studio è stato validare una *deep neural network* (DNN) capace di riconoscere attività simil-ospedaliere svolte da soggetti sani (progetto SHS). La robustezza della DNN è stata valutata su attività svolte da 12 pazienti ricoverati. Esse sono state acquisite come segnali di accelerazione da una coppia di sensori posizionati sul torace. La DNN di riferimento utilizzata in questo studio è in grado di riconoscere 5 classi: Disteso o Reclinato sul letto, postura Eretta, Cammino e spinta della sedia a rotelle. Inoltre, un algoritmo *rule-based* è stato valutato in autonomia e in combinazione con gli output probabilistici della DNN in un metodo ensemble (*Stacked generalization*). Tali modelli hanno offerto risultati modesti e indirizzato verso approcci basati sul *trasferimento di conoscenza*. Questo progetto risponde ad una specifica domanda di ricerca: fino a che punto è possibile trasferire la conoscenza appresa sulle attività dal dominio SHS *sorgente* a quello di *destinazione* (definito RHS)? I primi esperimenti hanno coinvolto tecniche tradizionali di *trasferimento di apprendimento*. In seguito è stato implementato un approccio basato su *adattamento di dominio* attraverso domain-adversarial neural networks (DANNs). Tali architetture sono state addestrate in modalità *non(semi)-supervisionata*. Il modello DANN ottimale (F1-score pesato: 0.73) è stato ottenuto rivelando porzioni di attività svolte da diversi pazienti RHS durante l'addestramento. In conclusione, il dataset RHS è risultato variegato e differente da quello SHS. La similitudine nell'obiettivo tra i progetti SHS e RHS è stata sfruttata per un *trasferimento di conoscenza* attraverso DANNs. Un'analisi sul *trasferimento dei kernel* convoluzionali ha dimostrato che la trasferibilità di essi (da SHS a RHS) è limitata. Fornendo alcune informazioni su attività del dominio RHS durante l'addestramento, si è ottenuto un miglioramento del riconoscimento delle attività. La metodologia DANN proposta in questo lavoro potrebbe essere validata in studi futuri annotando a campione le attività del dominio target. Ciò potrebbe alleggerire il processo di annotazione, preservando una buona performance nella classificazione delle attività.

Parole chiave: deep learning, adattamento dominio, riconoscimento attività, reti neurali

Contents

1 | Introduction

This chapter presents the main research area and question involved in this Master thesis project. The latter has been accomplished during a full-time 1-year internship within the Patient Care & Measurements department of Philips Electronics Nederland B.V. in Eindhoven. Due to the Covid-19 pandemic, most of the working experience has been carried out remotely. Thus, the access to in-place High Tech Campus (Eindhoven) labs and facilities has been limited. Nevertheless, the dataset analyzed and processed in this work had been previously acquired and made available for this study.

1.1. Reference study

The core of this research project is closely linked to a former one carried out within the same Philips department. That study has been published as a paper [15] and used as main reference throughout this work. More in detail, the objective of that work was to assess the accuracy of a deep neural network (DNN) model to recognize 6 different typical activities for hospitalized patients: LYING in bed, UPRIGHT posture, WALKING, self-propelled WHEELCHAIR transport, STAIR ASCENT and STAIR DESCENT. Those have been performed by 20 healthy volunteers in a simulated hospital environment following a specific activity protocol. The raw data have been captured by a single tri-axial accelerometer placed on the subjects' trunk. This project has been defined as "Simulated Hospital Study" (**SHS**). The results obtained by the developed DNN were quite promising in this setting.
The next step was to validate the latter model on a real hospital scenario. As a consequence, the dataset used in this project consisted in free-living activities carried out by 12 hospitalized patients. The under-exam setting has been termed "Real Hospital Study" (**RHS**). Table 1.1 summarizes the main differences between the two studies.

	Simulated Hospital Study	Real Hospital Study
Subjects	20	12
Dataset splitting	Random	(Partially) random
Activity protocol	Yes	No
Session time	1 hour	≈24 hours
DNN classes	6	5
Majority labels	Dynamic	Static

Table 1.1: Comparison between simulated and real hospital study.

1.2. Research question

The research question of this project focuses on a human activity recognition (HAR) task and has been formulated as follows.

- **Question**: to what extent is it possible to use a DNN model pre-trained on a simulated setting to classify patients' activities carried out in a free-living context?

The above-mentioned statement is supported by several motivations presented hereinafter. Physical activity and mobility are key indicators for the recovery process of general ward patients in the hospital. Despite the fact that inpatients spend the majority of their time in bed during hospitalization, tracking patterns in patient movement can be a useful predictor of consciousness, mental and physical healing. As an example, getting out of bed and taking a few steps can indicate patients' ability to tolerate activities of daily living and discharge readiness. Physical activity in the hospital is nowadays often monitored solely through direct observation by caregivers. Developing objective methods to assess mobility can help discriminate patients who require dedicate medical attention and those who are ready to be discharged safely.

1.3. State-of-the-art

The classification of human activities via wearable devices has been extensively covered in many studies. Their main differences involved different sensors types and locations, data collection protocols and processing or classification techniques [25]. Concerning sensor modalities, accelerometers stand out for being fairly parsimonious in production cost and power usage [34]. Several works achieved remarkable HAR performance using 3D acceleration data with different strategies [1, 6, 47]. In the last few years, acceleration-based

HAR approaches consistently shifted from feature-based models [4] to deep-learning (DL) ones [46]. Cutting edge DL models offer state-of-the-art performance without the need of strong physiological bases required for optimal feature extraction. The side-effect of adopting such models consists of their lack of interpretability. Plus, properly designed DL architectures are nowadays embedded within wearables for real-time processing.
The process of combining the information acquired from multiple sensor locations is called "*sensor fusion*". This approach can be applied to HAR tasks by placing a number of sensors in different body locations. Thus, *sensor fusion* might provide a more robust technique to perform HAR with respect to using a single recording device [2, 17]. On the other hand, the obvious by-product effect of using several sensors consists in a cumbersome data acquisition protocol. According to the use-case, this design choice might represent an issue. For instance, recognizing activities of daily living performed by healthy subjects may afford the use of several sensors [14, 43]. In contrast, this might become a serious problem when performing the same task entailing and elderly population [7], especially if hospitalized.
Furthermore, it is still challenging to identify the optimal sensor body location since this heavily depends on the specific HAR scenario [8]. Among several viable options, the sternum [32, 37] is often selected as candidate body position for placing wearable accelerometers. This setting matches with the sensor location used for this study.
Eventually, it is well-known that self-reported activities carried out by adults suffer from basement effect and recall bias due to under-reporting [42]. Pervasive monitoring via wearable devices constitutes a reliable and objective means to mitigate this issue.
One of the main challenges related to HAR using DL methods consists in reproducing the performance obtained in a laboratory setting in a real-life scenario. This facet encapsulates the research question of this project entailing SHS and RHS. A previous study highlighted how changes in activity patterns, initially following a protocol and then in a free-living setup, might overestimate the classification accuracy on training data [19]. Generally speaking, there might be many root causes of why DL models trained on a specific setup might perform differently on a different HAR scenario. Under the point of view of machine learning, it is possible to group discrepancies between source and target task into two main categories: *covariate* and *concept* shifts. The first can be due to differences in data acquisition protocol, activity distribution or sensor modalities. The latter can occur when there is a discrepancy between source and target activity classes. On top of that, the intrinsic variability of activity patterns between subjects contributes as an additional factor for dataset shift, especially in a clinical setting.
However, the latter issue is not only attributed to HAR tasks. In fact, it is desirable that machine learning models may have enough generalization power to preserve acceptable

performance when evaluated on different contexts. It is possible to transfer knowledge from source to target predictive models via *transfer learning* [33] assuming that a relationship between the two domains exists. Indeed, many studies focusing on HAR relied on this approach [9]. Due to the already-mentioned recent emergence of DL models this technique might also be defined as "*deep transfer learning*". The key benefit of using *deep transfer learning* models is the chance of unburdening the annotation process in target activities [11, 38, 45]. Such technique especially comes into hand when analyzing sensitive data privacy-wise. A prime example consists of activity data acquired from inpatients, as in our use-case.

2 | Material and methods

This chapter will thoroughly describe the pipeline designed to obtain pre-processed signals. In addition, some first insights obtained at the end of the data analysis will be presented. As a general rule, all the steps involved in the data acquisition process have been tailored to be embedded in a standard clinical workflow. This implied minimizing the interaction between the nurse or patient with the acquisition equipment while providing almost continuous monitoring.

2.1. Data acquisition

2.1.1. Dataset

The population consists of 12 general ward patients (7 males and 5 females, body mass index (BMI): $30.13 \pm 9.88 \frac{kg}{m^2}$) from a clinic located in the United States. This implied a consistent within-population heterogeneity in terms of length of stay, number of comorbidities, medical treatments and several other patient-specific features.
The enrollment occurred on a voluntary basis without exclusion criteria. An informed consent was dispensed to and signed by the inpatients interested in taking part to the study. This project has been approved by an internal committee at Philips in co-operation with the clinic counterpart.

2.1.2. Experimental protocol

The acquisition procedure entailed a ≈24-hours recording for both camera-based ground-truth and raw data acquired through a pair of wearable sensors.
The former consists of video recordings obtained via a cart-mounted camera with a telescopic pole and laptop stand placed within patient's hospital room. The camera has been stopped at least once to allow battery replacement or potential room transfer. Concerning its technical specifications, it is sufficient to point out the sampling frequency: $f_s = 15$ frames per second (fps). In addition, each frame has been associated with a timestamp (nanoseconds resolution) expressed in Posix time. A MATLAB® `.mat` file containing the

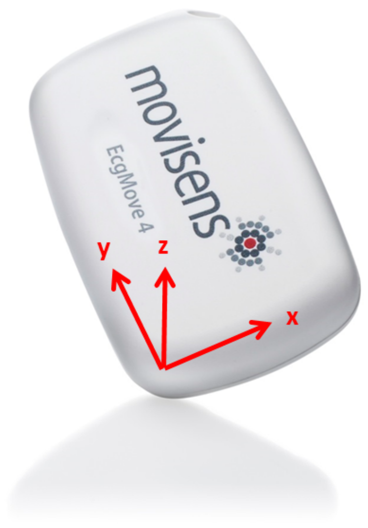


Figure 2.1: ECGMove4 axes orientation

frame timestamps has been generated for each video chunk (see annotation setup subsection 2.2.2). Further technical details cannot be disclosed since the camera is an internal Philips product.

The employed wearable devices consisted of two movisens® ECGMove4 sensors. The latter allows the on-line measurement of a short-lead ECG and the physical activity of a subject via a 3D accelerometer (see Fig. 2.1 for the axes orientation), gyroscope and atmospheric air pressure. The sensors configuration occurred via a preliminary USB connection to a PC hosting a dedicated software to set up acquisition parameters and starting time. It is relevant to note that the movisens® sensors have been almost always switched on after the camera. Section 2.3 outlines how the synchronization between the camera ground-truth frames and ECGMove4 samples has been handled. As a side note, ECGMove4 data logging has not been interrupted even when the camera recording has been stopped. However, acquired data in such time window have been discarded due to the lack of ground-truth data. Moreover, the recording session starting time occurred in different hours of the day across inpatients.

After stopping both sensors, the recorded raw signals for each ≈24-hours session have been stored as tabular (*.csv*) and compressed files (*.bin*). However, only the former data format has been imported and processed throughout this project. Furthermore, a `unisens.xml` file containing metadata information has been generated. The latter has been parsed to obtain information about the ECGMove4 internal sensors settings. Table 2.1 shows the ECGMove4 signals recorded during the acquisition procedure and their corresponding de-

Signal	Properties
ECG	Resolution: 12 bit, Range: ±5 mV, Output rate: 1024 Hz
3D acceleration	Range: ±16 g, Output rate: 64 Hz
Angular motion	Range: ±2000 dps, Resolution: 70 mdps, Output rate: 64 Hz
Pressure	Range: 300-1100 hPa, Resolution: 0.03 hPa, Output rate: 8Hz
Temperature	Output rate: 1 Hz

Table 2.1: ECGMove4 acquired signals and properties

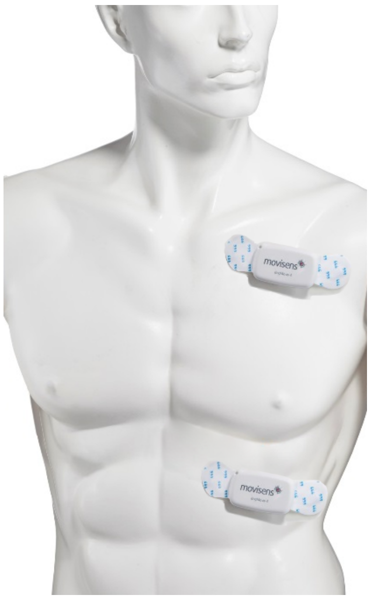


Figure 2.2: ECGMove4 placement within patient's chest

fault settings. An additional ad-hoc movisens® software could have been used to perform off-line analysis of the acquired raw data. However, this procedure has been carried out via custom `Python` scripts.
Concerning ECGMove4 placement, the two sensors have been attached on patient's left-side chest using disposable electrodes for improved comfort. One of the two movisens® devices has been placed between the first and second rib whereas the other one in correspondence of the twelfth rib (see Fig. 2.2). Those have defined as *Upper* and *Lower* ECGMove4, respectively. Figure 2.3 summarizes the acquisition environment presented so far.
The data acquisition technical protocol has been supervised and performed by two onsite Philips researchers and consisted of different steps.

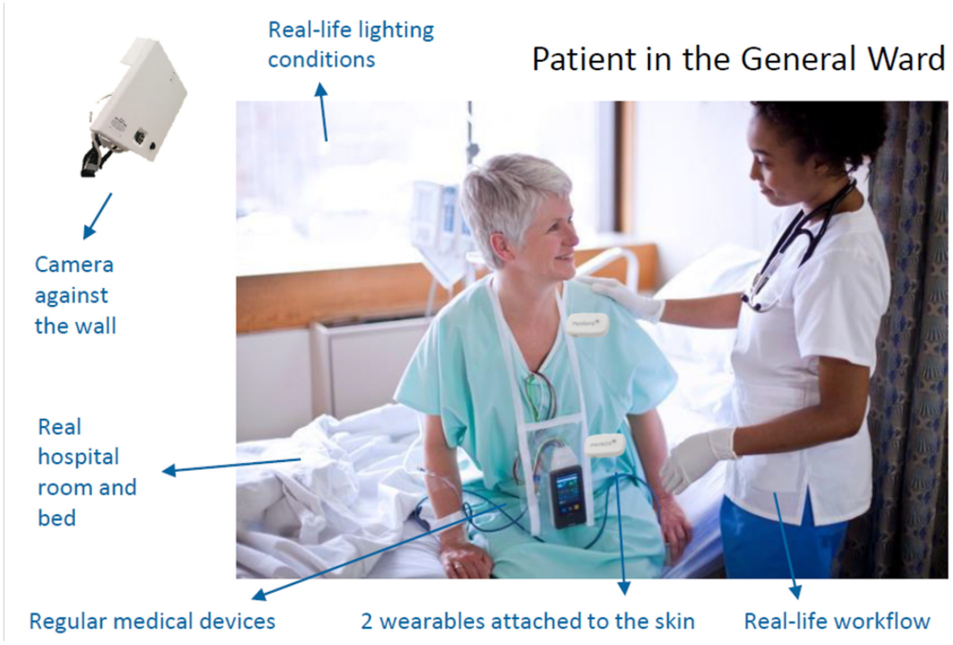

Real-life lighting conditions
Patient in the General Ward
Camera against the wall
Real hospital room and bed
Regular medical devices
2 wearables attached to the skin
Real-life workflow

Figure 2.3: Acquisition setup overview

1. Starting camera video recording after placing it within the hospital room.
2. Starting ECGMove4 devices acquisition.
3. Stacking and shaking them together in front of the camera to create motion artifacts (required for the synchronization procedure as described in section. 2.3)
4. Placing the two sensors on patient's left-side chest.
5. Starting a 30-minutes capnography session.
6. Removing both ECGMove4 wearables from patient's chest.
7. Repeating the stacking and shaking procedure (as outlined in step 3).
8. Switching both movisens® devices off.
9. Turning off the cart-mounted camera.

The corresponding timestamp for each of the listed events along with any relevant patient-specific annotations have been stored in a separate tabular `.csv` file. Moreover, the medical staff has been trained and involved in this procedure uniquely when placing or removing the ECGMove4 sensors.

2.2. Data annotation

It is worth to recall that patients carried out their activities in a free-living context. The following subsection will provide the rationales behind choosing the specific activity labels that have been eventually annotated.

2.2.1. Labelling strategy

Since the essence of this work is to perform an HAR task mainly based on DL techniques, it is required to carry out a careful and exhaustive labelling procedure on the recorded ground-truth video data. This implied the design of an ad-hoc labelling strategy, involving the generation of a label set.
The a-priori knowledge on activities performed by patients was limited. Thus, the final 28-classes label set represented in table 2.2 has been obtained after several revisions produced along with the annotation progress. The rationale behind choosing individual activity or posture labels was multi-faceted.

1. Making relevant distinctions from a clinical standpoint.
2. Preserve the SHS labelling strategy as much as possible.
3. Including postures used in pre-existing HAR algorithms.
4. Including contextual information.

Point 2 and 3 have been fulfilled for correctly validating the SHS pre-trained DNN (see DNN validation subsection 2.7.3) and CPA (see section 2.6) models, respectively. Point 4 allowed to use this dataset in future Philips R&D projects (e.g. in-bed detection).
Focusing on point 1, it mainly concerned categorizing static postures and has been addressed by relying on *Fowler's positions* [5] represented in Fig. 2.4. By definition, Fowler's postures only refer to patient's trunk inclination while sitting. However, it might be expected that the *Low-Fowler's* position (i.e. 30°) could be associated to low-intensity activities (e.g. reading, watching TV, talking to visitors, ...) whereas *Standard Fowler's* posture (i.e. 60°) to medium-to-low-intensity tasks (e.g. eating, grooming, ...). Thus, the final decision was to consider RECLINED as a distinct label from SUPINE and UPRIGHT.

2.2.2. Technical setup

The data labelling procedure has been carried out using a proprietary Philips annotation tool with codename `Barista`. All the annotations on the RHS dataset have been made by a single operator (i.e. the undersigned). Although this might have introduced annotation

Label	Definition
In bed	*Start*: Bed entry start *End*: Bed exit end
In (wheel)chair	*Start*: (Wheel)chair entry start *End*: (Wheel)chair end
Out of view	Patient legs no more in camera field of view
Supine	Patient trunk lying flat or $\leq 30°$ w.r.t[1]mattress
Upright	Patient trunk being vertical or $\geq 60°$ w.r.t[1]mattress
Reclined	Patient trunk being $\geq 30°$ and $\leq 60°$ w.r.t[1]mattress
Lying left	Patient lying on left side of his/her body
Lying right	Patient lying on right side of his/her body
Bend forward	Patient trunk being $\geq 45°$ between vertical and floor
Prone	Patient lying on belly
Bed entry	*Start*: patient making first contact with mattress *End*: all patient limbs lying within bed area
Bed exit	*Start*: patient starting moving towards edge of bed *End*: patient not making contact with mattress anymore
(Wheel)chair entry	*Start*: patient bending knees to sit in (wheel)chair *End*: patient making first contact with (wheel)chair
(Wheel)chair exit	*Start*: patient bending forward while in (wheel)chair *End*: patient not touching the (wheel)chair anymore
Self-propelled wheelchair	*Start*: patient touching wheelchair wheels to start propelling it *End*: patient not touching wheelchair wheels anymore
Eating (main) meal	*Start*: patient grasping food/drink/cutlery for the first time *End*: patient putting down food/drink on the overbed table
Standing/held upright	*Start*: patient standing with both feet touching ground *End*: patient lifting one foot or moving walking aid to start walking
Ambulating/taking steps	*Start*: patient starting initial swing of gait cycle *End*: both patient's feet touching ground (double support)
Physiotherapy	Nurse interacting with patient for a physiotherapy session
Capnography session	*Start*: referring to case report form *End*: referring to case report form
Transport in wheelchair	*Start*: nurse touching wheelchair seat-back to start propulsion *End*: nurse not touching the wheelchair seat-back anymore
Patient care	Nurse physically interacting with patient
Telemetry device repositioning	Nurse relocating telemetry device
Step	Occurrence of heel strike phase of gait cycle
Fall	Patient touching ground due to balance loss

Table 2.2: Label set description

[1] with respect to

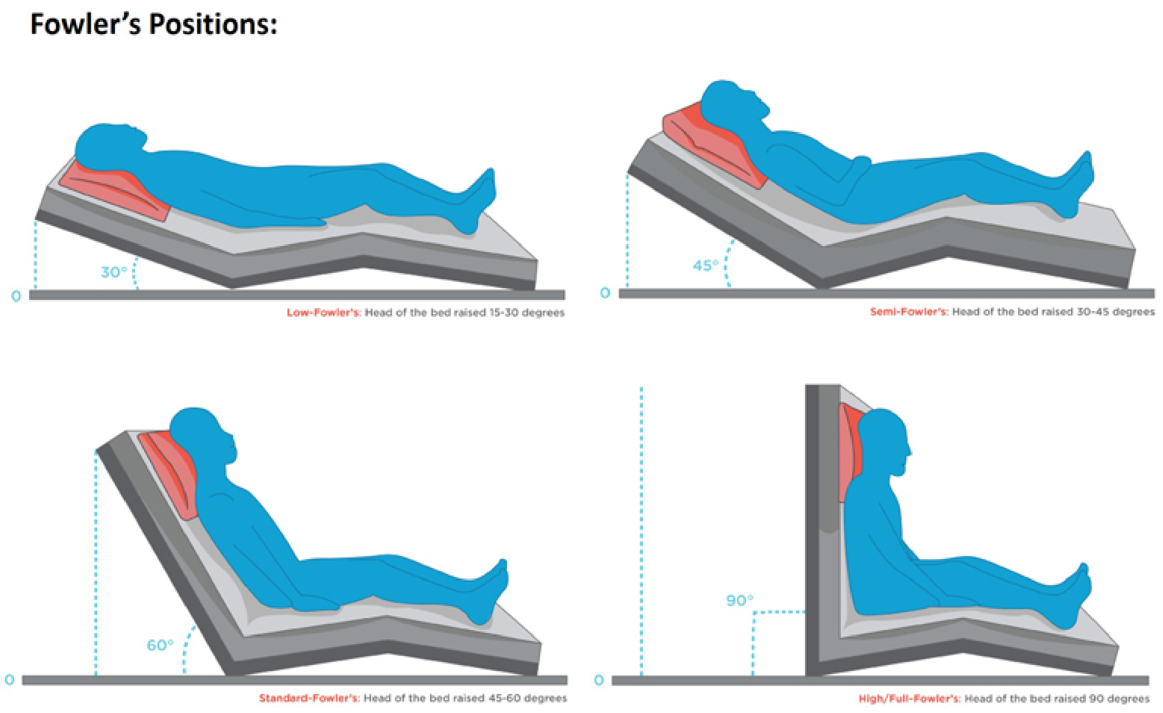

Fowler's Positions:
30°
0
Low-Fowler's: Head of the bed raised 15-30 degrees
45°
0
Semi-Fowler's: Head of the bed raised 30-45 degrees
60°
0
Standard-Fowler's: Head of the bed raised 45-60 degrees
90°
0
High/Full-Fowler's: Head of the bed raised 90 degrees

Figure 2.4: Fowler's positions

biases, it is valuable to point out that test patients activities have been annotated only when the development of HAR models used in this study was complete.
Practically speaking, annotations have been made on 10-minutes video chunks denoted as *"part"* (as described in subsection 2.7). Recalling that each of the 12 patients has been monitored for around 24 consecutive hours, the labelling process resulted fairly cumbersome with an overall estimated completion time of ≈165 hours. The `Barista` graphical user interface (depicted in Fig. 2.5) consisted of a blurred 10-minute video clip being played back (top pane) and a grid used for making annotations (bottom pane). The shaded strip on top reported the current video frame (top-left), the video chunk codename it belongs to (top-center) and its associated timestamp (top-right). The bottom grid allowed to visualize the annotations made so far. Each grid row was associated with a unique label and each dark-blue segment to the time span in which that specific posture or activity has been annotated. Conveniently, the light-blue endings could be used to resize and adjust the annotation time duration with a fine-grained resolution. According to the camera technical specification outlined in the experimental protocol subsection 2.1.2, its time resolution was ≈66 ms. The same applies to the video playback, in which it was possible to scan through individual frames. As a side note, the pink background indicated that two or more labels were concurrent.

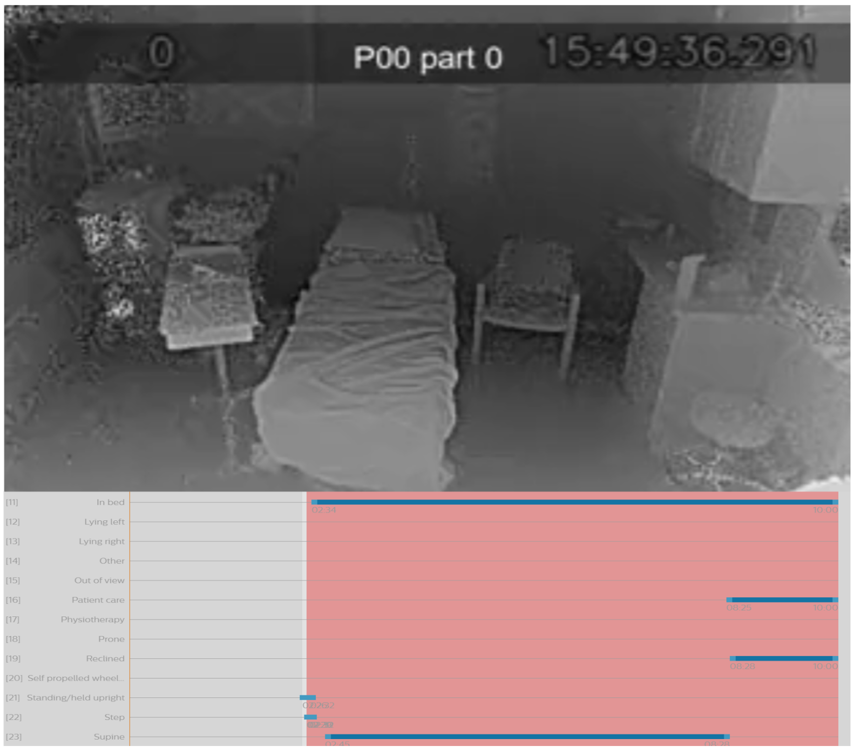


Figure 2.5: Annotation platform graphical user interface

2.3. Data synchronization

This routine allowed to align ECGMove4 3D acceleration samples with the ground-truth video frames. Practically, this has been achieved by generating acceleration artifacts (see data acquisition protocol steps in subsection 2.1.2). The procedure consisted in comparing the artifacts timestamps recorded by the wearables and the camera. In this way, the possible discrepancies between the two information have been qualified.

During the ≈24-hours recording the movisens® devices proved to be very stable in terms of sampling frequency drift. Conversely, the camera showed occasional frame drops. Although limited, such inconsistencies were mitigated by customizing two ECGMove4 parameters: sampling frequency (to a float value) and offset (of the first ECGMove4 sample with respect to camera start timestamp).

A simple visual test has been performed to verify the correctness of the above-mentioned adjustments. As an example, Fig. 2.6 illustrates the shaking artifacts generated at the beginning of P00 recording session. The red vertical lines indicates a specific sensor shaking direction. Ideally, whenever a movement artifact starts the corresponding acceleration norm values should immediately increase (and vice versa). This behaviour should be preserved by repeating the same procedure just before the end of the recording session for the same patient. Thereby, it can be assumed that the synchronization task was successful.

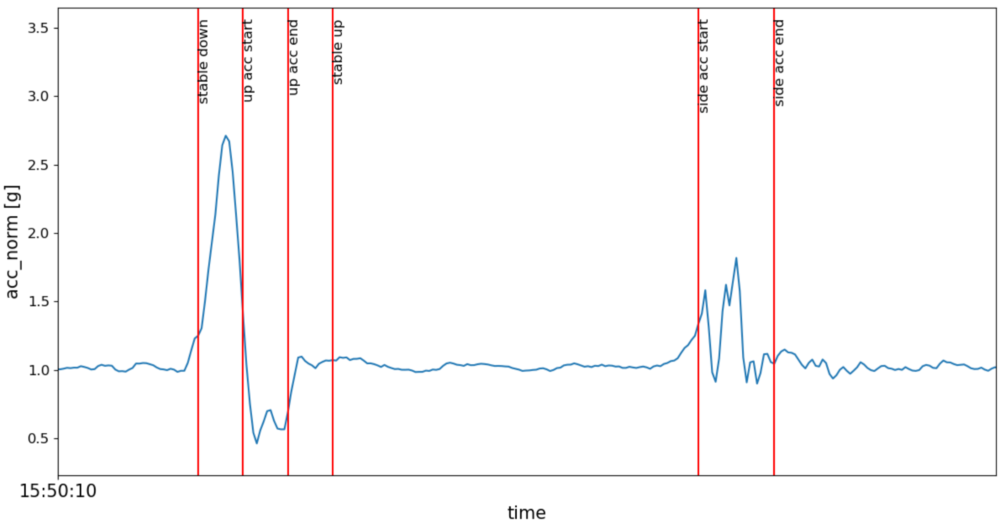


Figure 2.6: Synchronization shaking artifacts

No a-priori time lag criterion between shaking movements and acceleration norm increase (or decrease) timestamps has been set. Thus, a qualitative visual inspection has been performed on all patients.

2.4. Data pre-processing

A consistent part of this work consisted in designing an end-to-end data processing pipeline (represented in Fig. 2.7).

2.4.1. Preliminary steps

As reported in table 2.1, ECGMove4 devices were composed of multiple internal sensors acquiring different signals. However, 3D acceleration only has been selected as signal of interest throughout this work. In fact, one of the objectives of this project consists in the validation a pre-existing HAR DNN model trained on acceleration data. This implicitly required the same pre-processing steps followed in SHS to perform a fair validation
As a consequence, the 3D acceleration signal range has been halved in its range to ± 8 g (1 g = 9.8 m/s^2). On the other hand, the sensors adopted in the SHS study featured a 100 Hz acceleration sampling frequency whereas the movisens$^{\circ}$ devices were limited to 64 Hz. We found that by either *upsampling* or *downsampling* the ECGMove4 data did not remarkably impact the DNN performance. The final decision was to subsample ECGMove4 acceleration data to 16 Hz to allow a faster training procedure.

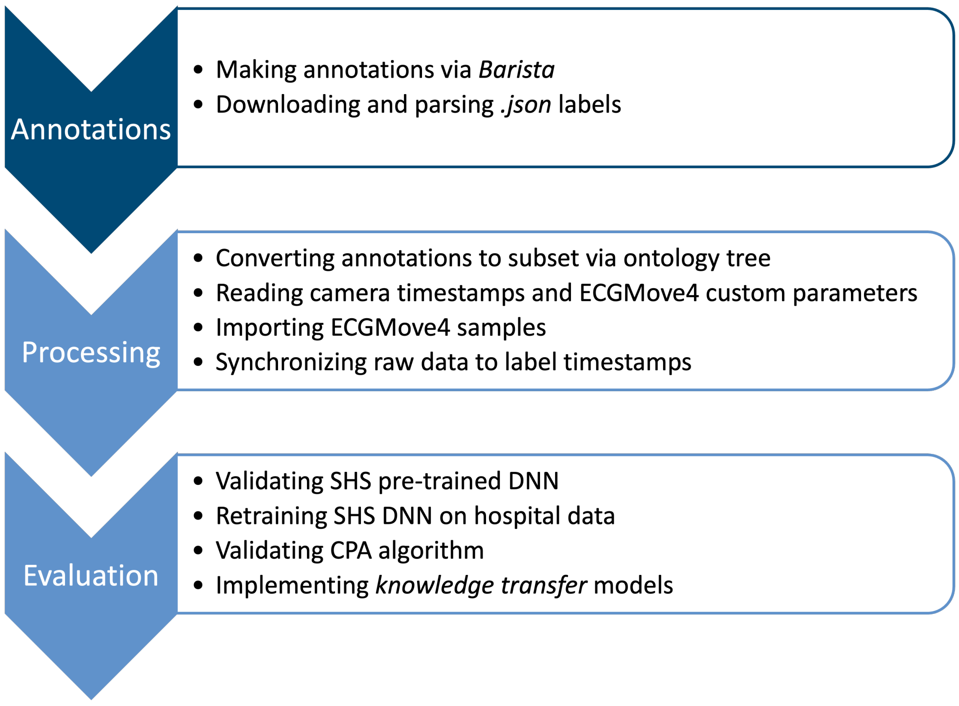

Annotations
Making annotations via *Barista*
Downloading and parsing *.json* labels
Processing
Converting annotations to subset via ontology tree
Reading camera timestamps and ECGMove4 custom parameters
Importing ECGMove4 samples
Synchronizing raw data to label timestamps
Evaluation
Validating SHS pre-trained DNN
Retraining SHS DNN on hospital data
Validating CPA algorithm
Implementing *knowledge transfer* models

Figure 2.7: Data processing pipeline

Eventually, the ECGMove4 acceleration axes orientation (represented in Fig. 2.1) has been transformed to match the SHS sensor reference system (see Fig. 2.8). The above-mentioned transformations along with the data windowing strategy (presented at the end of this section) have been preserved when developing novel HAR models throughout this project, unless otherwise stated.

What follows refers to label consistency across SHS and RHS as explained in point 2 within the labelling strategy subsection 2.2.1.
The first discrepancy between the two settings concerned concurrent RHS labels. By way of example: `Supine` while lying `In bed`). Since all the models developed in this study performed a single-activity classification, the *contextual* information has been filtered out. Thus, only the information about posture or activity has been retained.
Moreover, three special cases of multiple concurrent posture (or activities) have been identified and handled.

1. `Reclined` + `Lying right/left` → `Reclined`
2. `Ambulating/Taking steps` + `Upright` → `Ambulating/Taking steps`
3. `Upright` + (self-propelled) `Wheelchair` → (self-propelled) `Wheelchair`

Focusing on case 1, `Reclined` and `Lying left/right` might be equivalent under a physiological standpoint. However, the followed rationale was to make the HAR models more

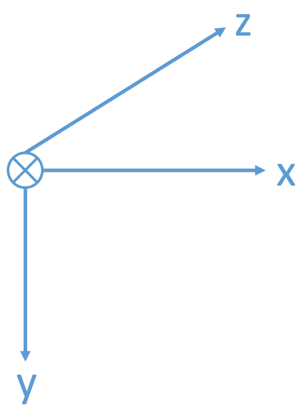


Figure 2.8: Transformed ECGMove4 axes orientation

robust to real use-case scenarios including posture variations.

Considering that this project entailed privacy-sensitive data, an extensive anonymization procedure has been implemented on the acquired video recordings. As shown in the sample video frame in Fig. 2.5, the original video was processed by converting it to a grayscale sequence of frames and superimposing a blurring effect. Such procedure has been performed to avoid the detection of the patient's face or peculiar features during the annotation process.
The adopted codename for each ≈10-minutes video chunk (e.g. top-center of sample snapshot in Fig. 2.5) was: *PXX part YYY (/2)*, where *XX* [01-12] referred to the patient ID, *YYY* to the video chunk number and *(/2)* to video segments acquired after premature camera stopping.
The same SHS data windowing strategy has been used to segment 3D hospital activities acceleration data (i.e. sliding 6-seconds windows with an overlapping of 50%). Each data window has been labeled via majority voting. Data segments containing more than 50% of unlabeled samples have been discarded.

2.4.2. Annotations download and parsing

The annotations made via `Barista` have been automatically retrieved via its Application Programming Interface (API). For this aim, a `Python` script has been developed to complete a two-folded task: logging-in to the online *Barista* platform via a local pre-generated token and downloading all annotations for all patients at once. Those have been stored as individual (i.e. one for each video *"part"*) `.json` file. The following represents a reduced information set related to an annotated event within a `.json` file.

```
{
"content": "{\"start\": 244.332, \"end\":600067}},
"context": {
    "formId: "P00 part 2",
    ...
    },
"created_at": "2020−07−18T18:06:40.994000",
"created:by": "Lorenzo Principi",
...
"labels": "In bed",
"last_modified_at": "2020−07−18T20:24:22.541000",
"last_modified_by: "Lorenzo Principi",
"owner_id": "Lorenzo Principi",
"type": "video"
},
```

The main information are reported in the `content` and `labels` field. The former indicates the start and end timestamps corresponding to the light-blue endings in the annotation grid as shown in Fig. 2.5. Those are expressed as milli-seconds time unit offset from the 10-minute video starting timestamp. The latter refers to one of the label names contained in the RHS label set reported in table 2.10. The remaining fields relate to various meta-data.
Next, an additional `Python` script has been implemented parse the content of `.json` files. After an appropriate routine, each labeled event has been associated to its respective camera timestamps (stored in a `.mat` file as described in the experimental protocol subsection 2.1.2). This final pre-processing step of allowed to obtain labeled data. Fig. 2.9 shows ECG and acceleration norm for a sample 30-seconds segments with annotations represented as colored regions with the same color code as in Fig. 2.10 according to their category. ECG has been visualized and overlaid to acceleration signal to verify that the two were correctly synchronized. In this way, the RHS dataset could be used for future projects entailing different signals and tasks.

To summarize, a final label set composed of 28 labels has been produced (see Fig. 2.10 for a compact representation of what already represented in table 2.2). However, a subset of activity classes has been chosen when developing HAR models in this work. The reason consisted in focusing on the most relevant posture and activities. This has been achieved by referring to an ad-hoc ontology tree (see table 2.3) used as lookup table. The unique values of its last column have been considered the final 5 RHS activity classes recognized by DL models: Lying, Upright, Reclined, Walking, Wheelchair).

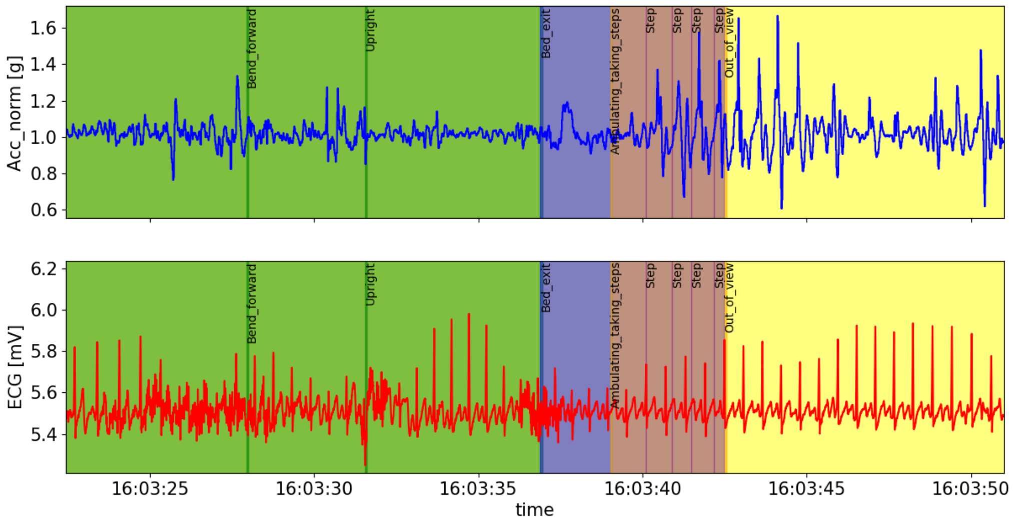
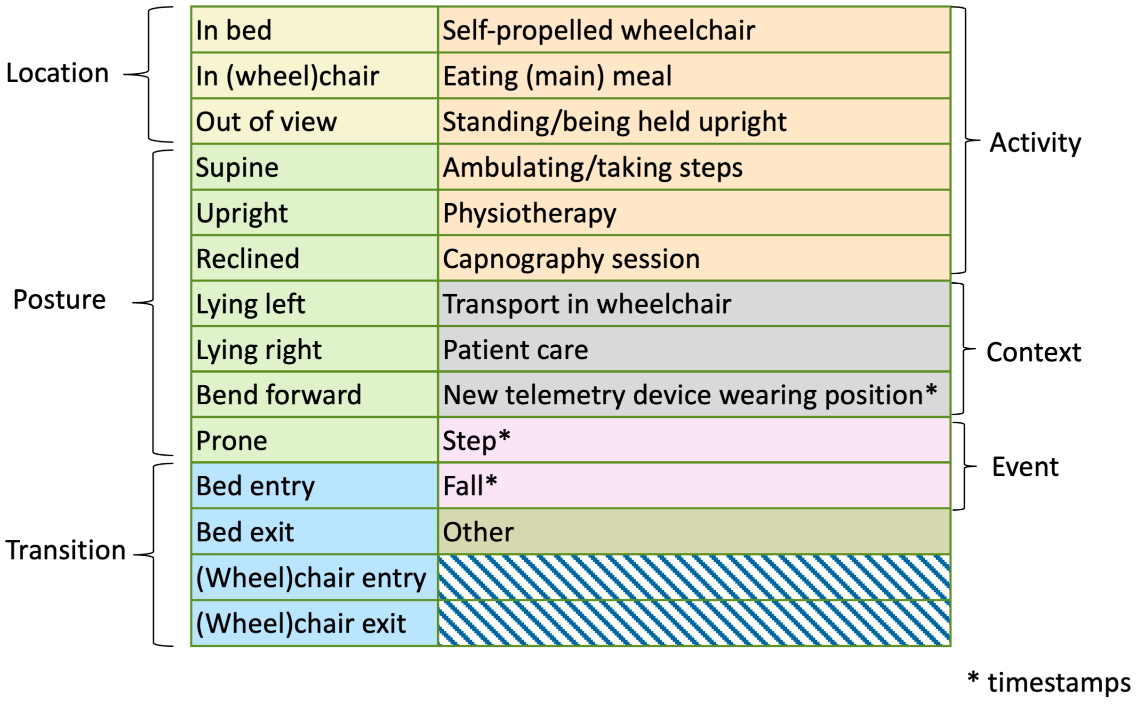


Figure 2.9: Sample ECG and acceleration annotated segment



Location	In bed	Self-propelled wheelchair	Activity
Location	In (wheel)chair	Eating (main) meal	Activity
Location	Out of view	Standing/being held upright	Activity
Posture	Supine	Ambulating/taking steps	Activity
Posture	Upright	Physiotherapy	Activity
Posture	Reclined	Capnography session	Activity
Posture	Lying left	Transport in wheelchair	Context
Posture	Lying right	Patient care	Context
Posture	Bend forward	New telemetry device wearing position*	Context
Posture	Prone	Step*	Event
Transition	Bed entry	Fall*	Event
Transition	Bed exit	Other	
Transition	(Wheel)chair entry	▨	
Transition	(Wheel)chair exit	▨	

* timestamps

Figure 2.10: Label set compact representation

Label	Category	Group	Posture	DNN class
In bed	Lie	Inactive	Unknown	Lying
In (wheel)chair	Sit	Inactive	Upright	Upright
Out of view	Other	Unknown	Unknown	None
Other	Other	Unknown	Unknown	None
Supine	Lie	Inactive	Supine	Lying
Upright	Sit	Inactive	Upright	Upright
Reclined	Lie	Inactive	Reclined	Reclined*
Lying left	Lie	Inactive	Left	Lying
Lying right	Lie	Inactive	Right	Lying
Bend forward	Transition	Inactive	Bend forward	None
Prone	Lie	Inactive	Prone	Lying
Bed entry	Transition	Active	Transition	None
Bed exit	Transition	Active	Transition	None
(Wheel)chair entry	Transition	Active	Transition	None
(Wheel)chair exit	Transition	Active	Transition	None
Eating main meal	AHDL	Active	Unknown	None
Standing/held upright	Stand	Inactive	Upright	Upright
Ambulating/taking steps	Walk	Active	Upright	Walking
Physiotherapy	AHDL	Active	Unknown	None
Patient care	AHDL	Inactive	Unknown	None
Telemetry device repositioning	Other	Unknown	Unknown	None
Step	Walk	Active	Upright	Walking
Fall	Walk	Active	Unknown	None
Capnography session	Lie	Inactive	Unknown	Unknown
Transport in wheelchair	AHDL	Inactive	Upright	Upright
Self-propelled wheelchair	AHDL	Inactive	Upright	Wheelchair

* LYING if validating deep neural network pre-trained on Simulated Hospital Study

Table 2.3: Label ontology tree

2.5. Data analysis

Before getting any insights on the pre-processed RHS dataset, the precondition was to perform a sanity check (presented in subsection 2.5.1). The aim of this procedure was to exclude potential outlier samples and unravel their root causes. Thereafter, some basic statistics (i.e. overall percentage of activity or inactivity, labels distribution, posture transition frequency) on the hospital activity dataset have been obtained. The results obtained at the end of the above procedure have been presented in section 3.1.1. Lastly, possible correlation between patient BMI and baseline 3D acceleration values during static postures (i.e. Supine and Upright) has been investigated (see subsection 3.1.2). The latter analysis aimed at quantifying the heterogeneity in ECGMove4 orientations and placements among RHS patients.

2.5.1. Sanity check

This procedure was carried out by selecting two opposite postures (i.e. Upright and Supine). Then, it has been assessed if the associated 3D acceleration readings matched with the axes orientation (depicted in Fig. 2.8. Specifically, Supine samples should take [0, 0, -1] values for x-, y- and z- axes, respectively (conversely, [0, -1, 0] for Upright). Many factors (e.g. sensor placement, chest shape, ...) might be the cause for slight deviations from the theoretical acceleration baseline values. Thus, this procedure was confined to identifying highly abnormal acceleration values for the above-mentioned postures. Namely, positive values for x-axis and z-axis when Upright and Supine, respectively. This has been carried out by computing the 3D acceleration values distributions on both ECGMove4 locations for the two postures. However, only the Supine posture featured abnormal values (within the red box in Fig. 2.11). It has been found that all those outliers referred to P12 activity patterns. More in detail, P12 often dragged her left arm (same side where the movisens® device was placed) under her head while lying on bed. This gesture implied a remarkable skin deformation, dragging the *Upper* ECGMove4 back. This behaviour, along with patient's chest shape and her very high BMI (> 40) constituted the root causes for the detected outliers. Since this does not relate to sensor malfunctions or defects, no actions concerning those abnormal samples have been taken.

2.6. CPA algorithm

This section introduces the first HAR algorithm evaluated within this project. Although this model is not DL-based, it has been involved in a later machine learning ensemble

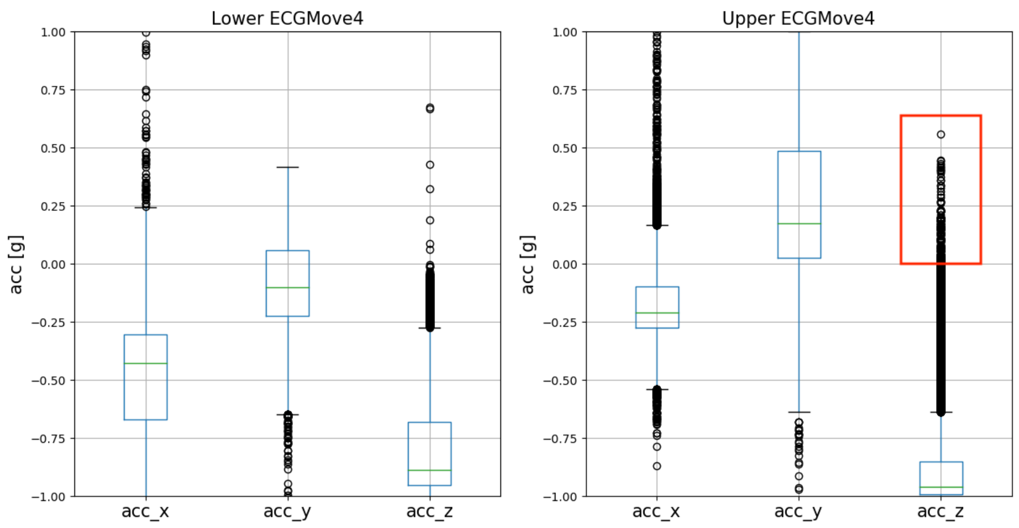


Figure 2.11: SUPINE 3D acceleration values distributions across patients

method (see subsection 2.7.5).

CPA stands for *Calibration*, *Posture* and *Activity* and refers to the three separate modules of which the algorithm is composed. It falls into the category of *rule-based* algorithms, meaning that the activity recognition task is performed by comparing the CPA output with a set of pre-determined conditions to determine the final posture or activity. Such conditions will be adequately explained within this section.

The structure of this section is organized so that an explanation on the *Calibration* and *Posture* routines is provided at first. The *Activity* one will not be covered because the intensity level at which tasks were performed was outside the scope of this project. Eventually, section 3.2 summarises the results obtained with different CPA configurations by also reporting some final considerations and recommendations for future use. Since the CPA algorithm is under Philips intellectual property, only general details to understand its high-level functioning will be disclosed.

2.6.1. CPA requirements and modules

Differently from the data windowing strategy used for DL models (see end of preliminary pre-processing subsection2.4.1)), CPA algorithm requires 5-seconds 3D acceleration segments with no overlapping as input. The acceleration data should be acquired from a sensor placed in the upper-chest location on the left side. This requirement has been fulfilled by selecting *Upper* ECGMove4.

Calibration

The aim of the *Calibration* module was to align the sensor axes reference system with the anatomical body directions. Referring to the transformed ECGMove4 axes orientations in Fig. 2.8, the x-, y- and z- acceleration axes should have been aligned with the right-left, cranio-caudal, antero-posterior anatomical directions, respectively. The purpose of this procedure was to even out potential interfering factors (e.g. BMI, chest shape, sensor placement, ...) for the posture detection. The *Calibration* procedure entailed four separate steps.

1. Walking detection.
2. Selecting calibration data.
3. Determining reference and device frame vertical axes.
4. Adjusting orientation matrix

Step 1 involved an algorithm that outputted the probability associated with a WALKING activity within each 5-seconds 3D acceleration window. In case enough valid and consecutive WALKING acceleration segments were detected, step 2 used those as input to retrieve the reference vertical direction. The core idea behind the *Calibration* procedure is that the posture taken during *Walking* should be vertical. Thus, the next step consisted of calculating the vertical axis within the device frame by dividing the acquired mean acceleration by its norm. Eventually, the reference and device vertical directions retrieved in step 3 were compared to assess the x-axis tilt angle (α) between the two. Such angle was the used to generate a 3x3 orientation matrix (R_{orient}) used to rotate and align the device reference system with the anatomical body directions. By default, $\alpha = 12.045°$ (rotated backwards).

Posture

The CPA algorithm has been designed to classify 7 different postures: `Upright`, `Reclined`, `Lie supine`, `Prone`, `Lie on left side`, `Lie on right side`, `Upside down`. This module is composed of four different steps.

1. Creating a 7x3 posture matrix (P) by horizontally stacking unit-length posture vectors p_i ($i = 0, ..., 6$).
2. Rotating the 3x1 posture vectors (p_i) by the R_{orient} orientation matrix: $\tilde{p}_i = R_{orient} p_i$.
3. Calculating a normalized mean value for each acceleration axis within each data

segment.

4. Selecting the final posture index ($i = 0, ..., 6$) based on a minimal distance criterion between p_i and $\tilde{p}_i$.

It is clear that the approach followed in the *Posture* module is similar the *Calibration* one. In this case, the p_i posture vectors represented the reference vectors (e.g. $p_{upright} = [0, -1, 0]$) to which the rotated $\tilde{p}_i$ ones were compared. After a normalization step, the output posture was selected for the associated i_{th} index that minimized the distance between the reference p_i and rotated $\tilde{p}_i$ posture vectors.

2.7. Deep-learning techniques

In the recent decades, approaches based on artificial intelligence (AI) increasingly took place in several engineering-related fields to overcome a variety of challenges. At the same time, the technological advancement allowed the design of unobtrusive devices capable of continuously acquiring data. In this way, it has been made possible to take full advantage of DL or machine learning models usually requiring a large amount of labeled data. Also HAR tasks have recently been addressed via such techniques, especially thanks to the development of wearable devices. Thus, the under-exam RHS task has been considered well-suited for being tackled with DL techniques.
This section is structured so that an introduction with general concepts about DL models will be provided at first. Secondly, a detailed analysis of the legacy DNN used for the reference SHS and the rationale followed to develop the actual baseline DNN used throughout this work will be presented. Then, the results obtained by validating the latter model on the RHS will be reported. Next, the strategy pursued to perform a complete retraining of the baseline DNN on RHS will be discussed. Finally, in order to combine the respective advantages of the CPA and DL models, a hybrid DL ensemble method will be discussed.

2.7.1. General concepts

This project heavily relied on testing and developing *artificial neural networks* (ANNs) in a deep paradigm. Hereafter, an high-level overview of the structure and functioning of such models is given.
ANNs are a subset of machine learning techniques aimed at tackling AI-related tasks. Those consist of circuits of artificial neurons reflecting the behaviour of biological neural networks contained in the human brain. The building block of such models is represented

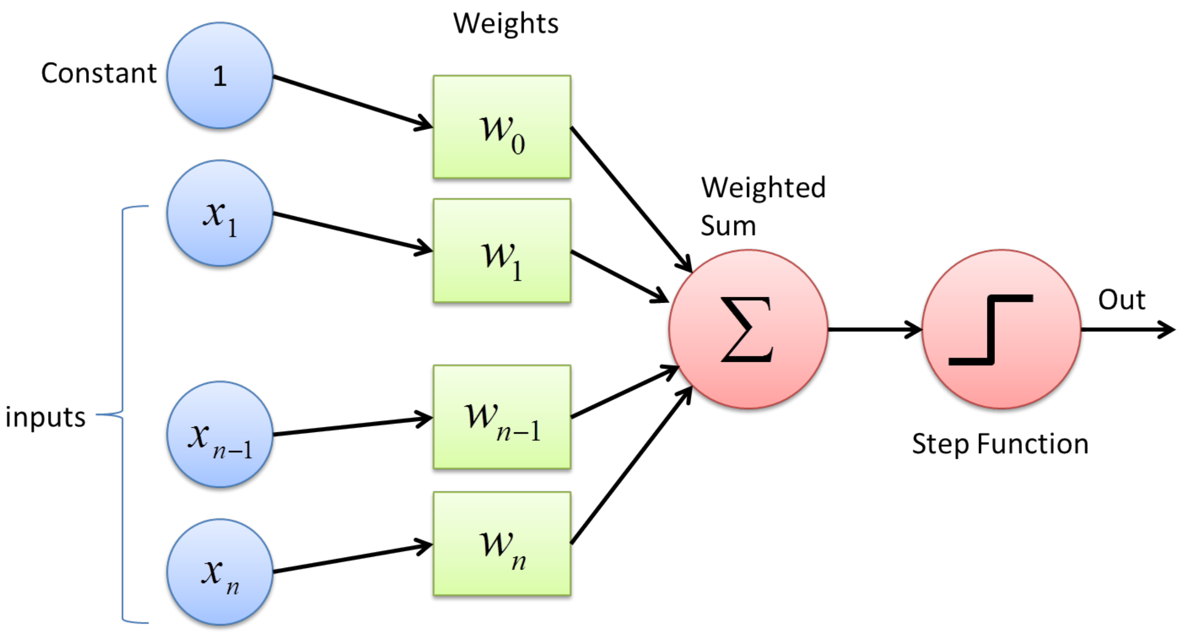


Figure 2.12: Graphical representation of a *perceptron*

by the above-mentioned artificial neurons. Vast research has been carried out throughout the last century in order to correctly model it. The first milestone has been achieved by understanding how the human brain could produce complex patterns, by comparing neurons with a binary threshold to a Boolean logic [30]. On a later stage, F. Rosenblatt expanded the concept of the all-or-none neuron by introducing neural connections weights and redefining it as *perceptron*[35]. Within the same work, a learning rule (*delta rule*) aimed at finding the optimal weight coefficients for neural connections has been presented. A schematic representation of a *perceptron* is depicted in Fig. 2.12 and described by the following equations:

$$o(\vec{x}) = sgn(\vec{\omega} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1, & \text{if } y > 0. \\ -1, & \text{otherwise.} \end{cases}$$

where $\vec{x}$, $\vec{o}$ and $\vec{\omega}$ represent the *input*, *output* and *weight* vectors respectively. However, *perceptron* per se can only be used to accomplish classification tasks involving linearly-separable classes.

The latter limitation has been overcome by the so-called *multilayer perceptrons* (MLP) models being able to perform non-linear classification tasks. Those structures resembled the intricate circuitry of biological neural networks devoted to automatic feature extraction. MLP are composed of *input layers*, *hidden layers* (performing to the actual feature extraction process) and a final *output layer* (see Fig. 2.13). However, the *delta rule*

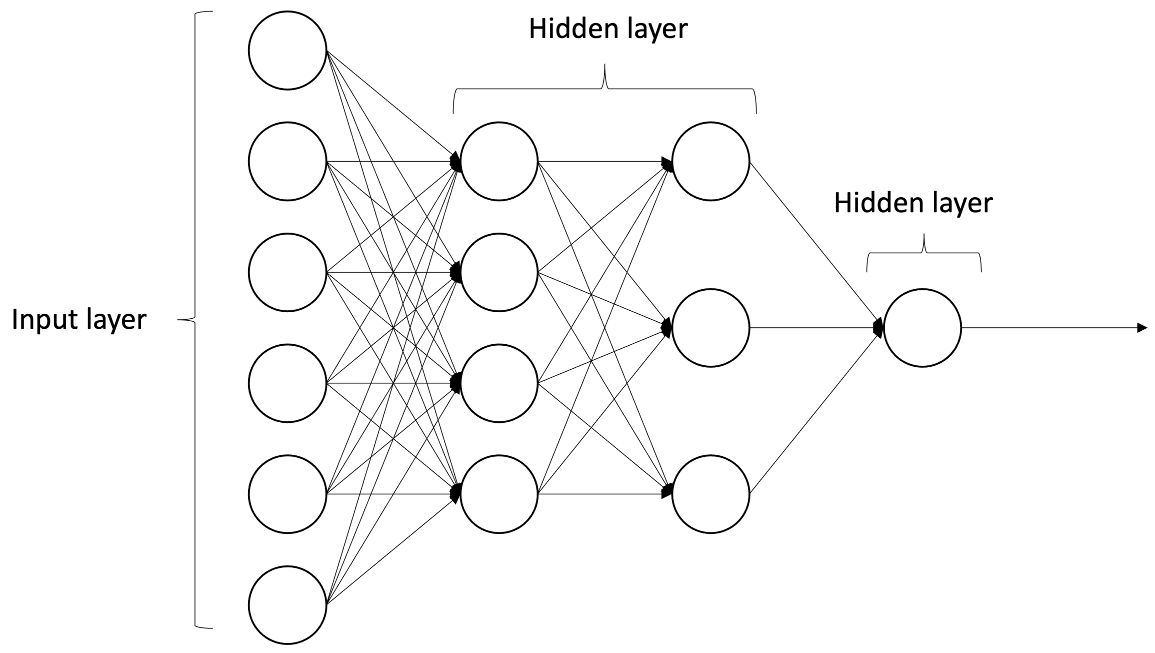


Figure 2.13: Multilayer perceptrons scheme

learning strategy is no more applicable in this complex context. As a consequence, a generalized learning rule based on computing the gradient of the loss function with respect to the weights called *backpropagation* [36] has been designed. The following equation synthetically describes the *backpropagation* algorithm coupled with stochastic gradient descent iterative method:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta^t}$$

where θ^t denotes the parameters of the ANN at iteration t, $X = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}$ the set of input-output pairs of size N, $E(X, \theta^t)$ the error function and α (*learning rate)* the coefficient that regulates the amount of novel information overwriting the old parameters. As a general principle, the aim of an optimization problem is to find the optimal weights that minimize the error function associated to the predictive task. In this regard, it is typical that ANNs error functions are characterized by non-convexity. This implied that the convergence to the global minimum via *backpropagation* with gradient descent was not guaranteed. However, it has been proved that the latter issue is not a major drawback in most of the practical scenarios [26].
Nowadays, thanks to an ever-increasing production of data made possible by pervasive sensing solutions, data-driven models found their success in many applications. Accordingly, it has been deemed necessary to additionally increase the complexity of MLPs models to identify meaningful patterns and perform successful feature extraction within a large amount of labelled data. As a consequence, the "deep-learning" umbrella-term

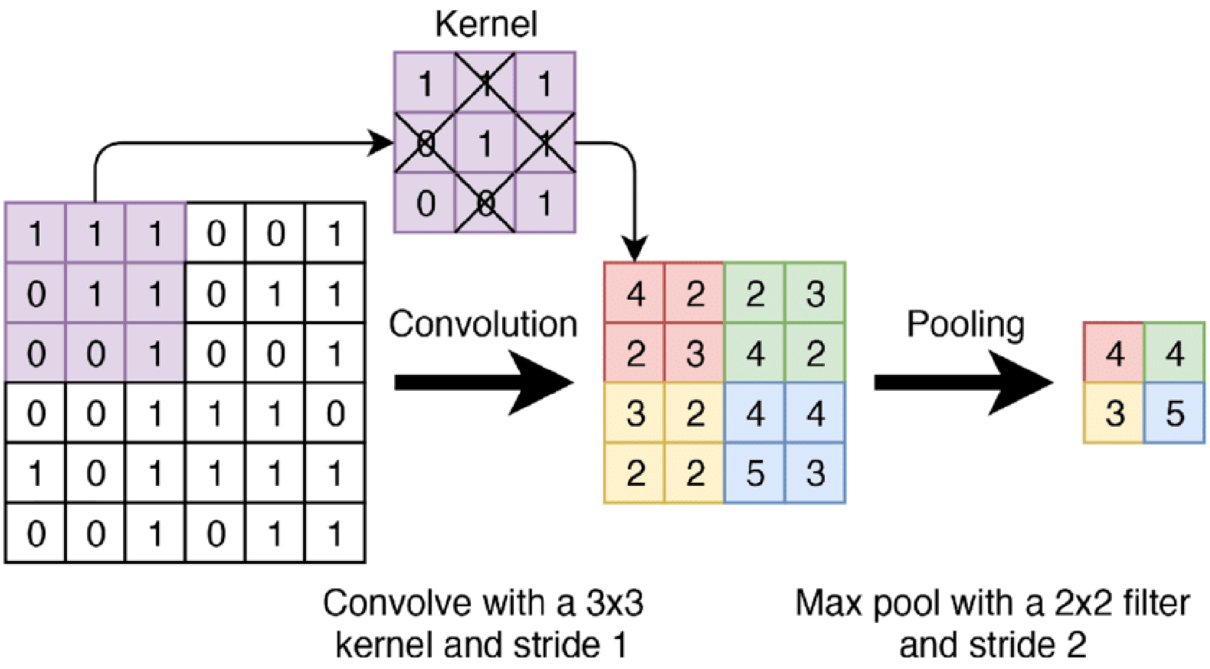


Figure 2.14: Example of a convolutional block

encompasses ANNs composed of several MLP *hidden layers* and, possibly, some changes to their standard structure (e.g. sparse or non-feedforward neural connections for convolutional and recurrent neural networks, respectively) [18]. Such architectures are broadly defined as deep neural networks (DNNs). Since often employed within this project, it might be beneficial to briefly introduce the main aspects of convolutional neural networks (CNNs). Their sparse connectivity allows for local feature detection, better resembling the feature extraction process performed by biological neural networks involved in visual tasks (i.e. from coarse-grained to low-level features). The essential components of a convolutional layer are: the input data, the *filter* (acting as a feature detector) and the *feature map*. The latter represents the result of the convolution (dot product) between the filter kernel striding over the input data. In general, each convolutional layer is mainly characterized by three parameters: the number of filters, the stride and the padding strategy (in case the filter does not fit the input data shape). Right after, a sub-sampling (or pooling) layer is placed in order to reduce the feature map dimensionality and retaining the useful information. Fig. 2.14 shows an example of a convolutional and pooling layer in series which are commonly referred as *convolutional block*. The crossed kernel filter squares in Fig. 2.14 indicate a null dot product. A series of convolutional blocks, used as feature extractor, is usually provided as input to a MLP to perform the final predictive task.

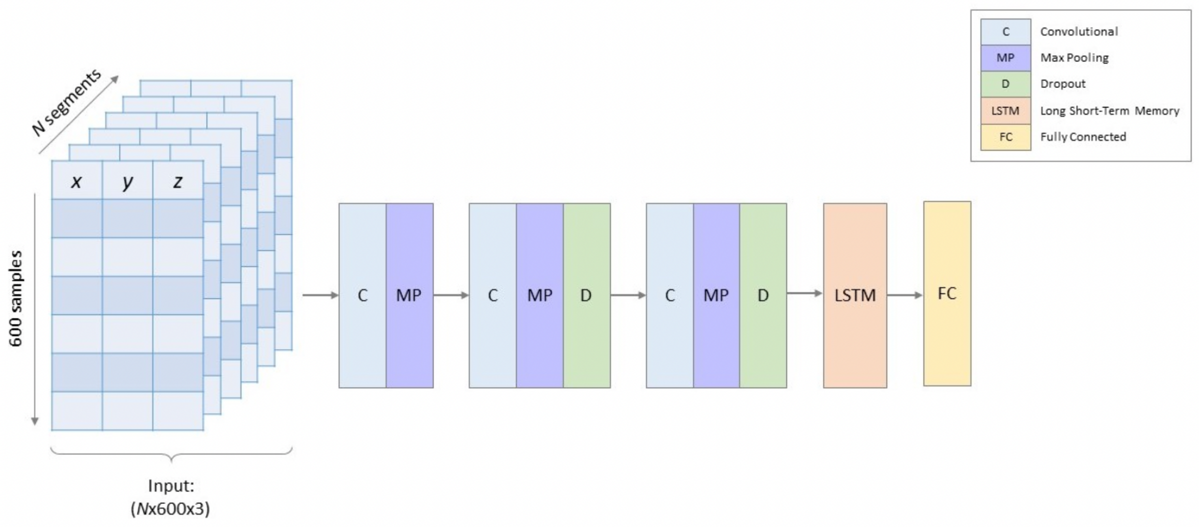


Figure 2.15: Simulated Hospital Study deep neural network architecture

2.7.2. Baseline deep neural network architectures

This study bases its foundations on the DNN developed in the SHS. Thus, it is of key importance to describe the DNN architecture designed for that use-case. It has been developed using `TensorFlow` framework and its main components are 3 zero-padded convolutional layers (number of filters: $f_1 = 8$, $f_2 = 8$, $f_3 = 8$ with kernel sizes: $k_1 = 23$, $k_2 = 10$, $k_3 = 7$) characterized by Rectified Linear Unit (ReLu) activation function. The latter allowed to zero out weights taking negative values, implying a sparsity of the DNN connections and an overall improved model efficiency. The last 2 convolutional blocks were fed to a *dropout* layer (ratio: 30%). Moreover, *batch normalization* layers have been interposed after each convolutional one to even out the differences between each mini-batch and speed up the DNN training process [21]. The pooling layers for each of the 3 convolutional blocks consisted of *max-pooling* (pool sizes: 10, 4 and 2 respectively). The final two layers consisted of a Long Short-Term Memory (LSTM) with 6 units and a fully-connected one. The latter featured 6 neurons, coinciding with the number of SHS DNN activity labels: Lying in bed, Upright, Walking, Walking downstairs, Walking upstairs, Wheelchair). The block-diagram representation of the SHS DNN architecture is depicted in Fig. 2.15.

However, some issues arised when retrieving above-described DNN model. In fact, no `Tensorflow` model checkpoint with which the results in [15] have been obtained was available. After an intensive investigation, the replication of such results turned out to be unfeasible. In addition, it has been found out that the SHS DNN was quite sensible to parameter initialization with performance gaps as large as ≈20% for the same posture

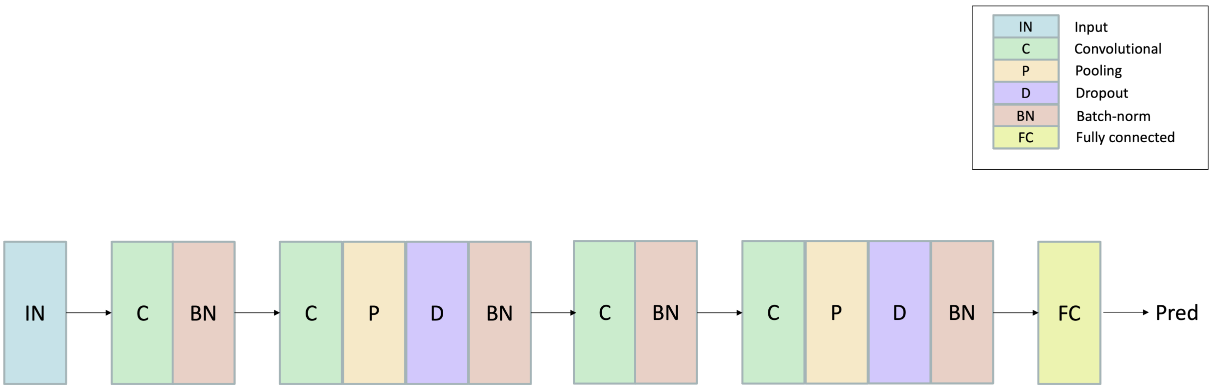


Figure 2.16: Baseline deep neural network architecture

(e.g. Upright or Wheelchair) by simply changing the random number generation seed.
On top of that, the sensor location used in [15] roughly matched with the *Lower* ECG-Move4. Such location neither resembled to *Upper* or *Lower* ECGMove4 locations. However, SHS acquisition protocol entailed multiple sensor placed in different locations. One of those was attached to the left-side upper-chest area, almost perfectly matching with *Upper* ECGMove4 placement.
Other than the legacy DNN model above described, additional testing have been carried out implementing variations on its core structure. In particular, a novel DNN architecture featuring an extra 1D convolutional layer in place of the final LSTM one has been selected as reference DNN for further testing on ongoing Philips R&D projects. With the vision of deploying such DNN on an embedded system, it is much more convenient to remove LSTM layers. In fact, those are generally known to be computationally cumbersome and difficult to implement in such environment. Therefore, such DNN architecture with codename *ccpd2* (see Fig. 2.16) has been actually used as baseline DNN for the current study.

Concerning training process hyperparameters, all the DNN models implemented in this work have been trained for 100 epochs with early stopping enabled, using Adam optimizer [24], 10^{-3} learning rate and a batch size of 100, unless otherwise stated. In addition, due to the class imbalance (see label distribution Fig. 3.1) a balanced batch generator based on minority class oversampling has been implemented via `imbalanced-learn Python` module. Concerning performance metrics, a *Cross-Entropy* loss function has been adopted whereas the weighted *F1-score* has been usually used to assess the overall model accuracy.

To summarise, different reasons made the validation of the exact same DNN architecture used in SHS unfeasible. Hence, the *ccpd2* architecture has been selected baseline DNN.

This model received 3D acceleration data sampled ad 16 Hz acquired on sensors placed on the upper-chest as input. Hereafter a list of motivations followed to select the *ccpd2* model as baseline model is provided-

1. Model architecture (*ccpd2*)
    - **Kernel transfer analysis**: since this project intends to understand to what extent features extracted from SHS might be representative for different datasets. Having more convolutional layers might be helpful in better differentiating which layers provide generic or specific (to the use-case) features
2. Sensor location (*Upper*)
    - **Comparison with feature-based model**: to allow a fair comparison between DNN-based approaches and CPA algorithm (see section 2.6) designed to receive input data acquired from sensors placed around the upper-chest
3. Sampling frequency (*16 Hz*)
    - **Computational efficiency**: with the vision of later implementing a DNN model within an embedded system. Within this setting, computational costs are a top priority in terms of energy consumption and real-time processing. Obviously, down-sampling 3D acceleration data helps in this task. A by-product effect consists of speeding up DNN models training without losing any relevant information about activities and postures patterns

It is important to note that, at this point, the validation procedure reported in the next subsection not only aims at evaluating the previously developed DNN models on a real setting scenario but also at benchmarking its robustness to inevitably slight variations associated to a different target use-case.

2.7.3. Baseline model validation

The validation procedure could not be straightforwardly performed due to the intrinsic differences between SHS and RHS (reported in the comparison table 1.1).
The first inconsistency concerned DNN activity classes. The main difference consisted of the lack of STAIRS WALKING and RECLINED DNN labels within RHS and SHS settings, respectively. The former can be justified by the fact that the camera field of view was restricted to patient's room. The latter was instead merely considered as a variation of the LYING posture within SHS. Thus, a required step was to include the *Reclined* posture as a separate class for the SHS. Considering that, at this stage, the validation process

was no more strictly bounded to the same SHS DNN architecture and settings, this final adjustment was considered harmless.
The additional discrepancy related to dataset splitting into train, validation and test subsets. Referring again to table 1.1, SHS performed a complete per-patient random splitting whereas RHS identified 5 patients as test ones even before starting the data acquisition procedure. Indeed, this consisted in the best strategy to avoid possible biases during the annotation process. Although not an issue during the DNN validation process, this resulted quite problematic when retraining the *ccpd2* on hospital activity data due to the consistent label distribution shifts between train, validation and test inpatients. During the experiments performed on the latter scenario, various combinations of hospitalized patients in the three data subsets were tested. Nevertheless, no substantial improvement of the DNN model performance has been achieved by doing so. By default, other than the 5 pre-defined hospitalized test patients, 5 patients were assigned to the train set whereas 2 inpatients to the validation set, unless otherwise stated.
As a side note, since the SHS DNN involved the normalization of the input data via a *scikit-learn* `StandardScaler`, it was necessary to re-fit the same scaler instance when validating it on hospital activity data. In fact, the two settings are characterized by completely different 3D acceleration values distributions.

2.7.4. Baseline model retraining on hospital data

The next step was to retrain the baseline *ccpd2* DNN architecture using training set inpatients acceleration samples and validating it on the 5 pre-defined test inpatients activity data.
As a "debugging" procedure consequent to the results reported in the retrained *ccpd2* confusion matrix in Fig. 3.5, a per-patient investigation has been carried out. The intent of this analysis was to possibly highlight patients featuring unexpected 3D acceleration patterns for specific activities or postures. According to the poor performance achieved by the retrained *ccpd2* DNN validated on P01 acceleration samples only, such inpatient could be a candidate outlier with respect to the hospitalized population. As an example, Fig. 2.17 shows a ≈1-minute 3D acceleration segment acquired by *Upper* ECGMove4 device annotated as Upright but entirely misclassified as Reclined. It is clear that the ≈ 0.7 x-axis stationary value is not an expected to occur when Upright. Such behaviour is not sporadic and has been possibly associated to an exaggerate left-lateral placement of the *Upper* movisens® device or interfering factors that caused this peculiar sensor displacement. This occurrence has already been discussed in the BMI and 3D acceleration correlation section 3.2. No other inpatients featuring prolonged abnormal acceleration

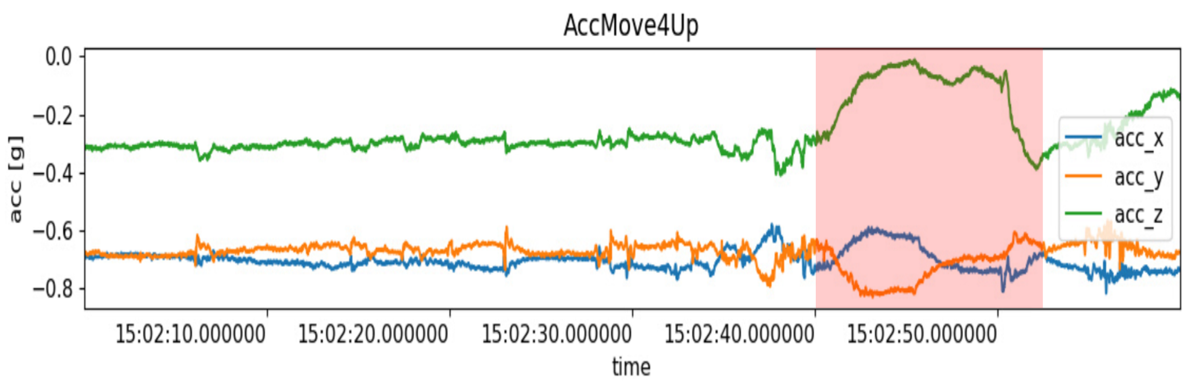


Figure 2.17: Incorrectly classified P01 3D acceleration segment

values showed up at the end of this analysis.
As a by-product finding of this procedure, it has been observed that P05 featured two very noisy and long-lasting (≈ 1-hour) 3D acceleration segments within the 24-hours recording. The culprit was a High-Frequency Chest Wall Oscillation device used as daily therapy to dissolve mucus congestions and free the airway.

2.7.5. Hybrid ensemble method

The rationale behind this method was to capitalizes on the advantages offered by intrinsic difference between CPA and DNN model (i.e. determinism and *black-box*, respectively). The devised solution xwas to implement an ensemble method. Among the many viable solutions, a hybrid *Stacked Generalization* framework has been designed. The term "hybrid" indicates that a heterogeneous collection of weak learners (i.e. CPA and *ccpd2*) was involved.

General concepts

A *Stacked Generalization* ensemble method, commonly defined as *Stacking*, involves combining the predictions from different models used for the same dataset. Such task is performed by designing a single machine learning model (*Meta-Model* or *Meta-Learner*) able to receive as input the outputs from two or more *Base-Models*. The former (also defined as Level-1 model) learns how much to trust the predictions obtained from the Level-0 models during its training phase. A *Logistic regressor* is usually identified as a suitable *Meta-Model* since it conveniently allows to visualize and assess the contribution of each weak learner to the final prediction.

Concerning the training procedure, it has been decided to follow a random per-patient splitting ratio for each of the three data subsets (D_1, D_2, D_3) involved in this process:

- D_1 (9 patients): used to train Level-0 models
- D_2 (1 patient): used to train the *Meta-Learner* and validate *Base-Models*
- D_3 (2 patients): used to validate the Level-1 model

Implementation

The Level-1 model consisted of a *Logistic regressor* whereas the Level-0 ones of *ccpd2* DNN model and CPA algorithm. Their output before the classification step consisted of softmax-ed values and distances (defined as *deltas*) from the reference posture vectors for each activity class, respectively. Such values have been vertically stacked and used to train the Level-1 model. Table 2.4 shows an example of a *Meta-Learner* training sample in which both the *ccpd2* DNN and the CPA are in agreement (bold values) to classify it as LYING. Thus, the Level-1 model training set has shape Nx12, where N represents the

Lying	*Reclined*	*Upright*	*Walking*	*Wheelchair*
0.99	0	0	0	0

(a) DNN softmax-ed outputs

Lying	*Reclined*	*Supine*	*Prone*	*Lie left*	*Lie right*	*Upside down*
1.46	0.82	**0.12**	2	1.34	1.46	1.37

(b) CPA deltas

Table 2.4: Meta-Model training sample

number of *Base-model* output samples.

Although the two *Base-Models* recognized different activity labels, it was still valuable to preserve such differences to identify possible relationships between specific CPA postures and DNN classes. It is important to note that the *Stacking* ensemble method had to be compared to the previously obtained DNN results. As a consequence, the hybrid ensemble method eventually outputted the same 5 RHS activity classes.

Some preliminary transformations were required to implement the *Stacking ensemble* framework. As already mentioned, CPA and DNN entailed two different data windowing strategies. However, the DNN windowing strategy has been picked for a fair comparison. In addition, the *Logistic regressor Meta-Model* required a preliminary feature normalization step for a correct interpretation of the regression coefficients. Thus, a `MinMax` scaler has been fitted and applied to the *Meta-Learner* training dataset (D_2). Eventually, the choice of the specific Level-1 *Logistic regression* model parameters has been carried out via a `GridSearch` with a 10-fold cross validation. The optimal *Logistic regression* settings obtained after the hyperparameters tuning procedure is reported below:

```
LogisticRegression(C=0.0001, class_weight='balanced', solver='newton-cg')
```

where `C` represented the inverse of the regularization strength, `class_weight='balanced'` indicated that the weights have been adjusted according to the inverse empirical frequency for each activity class and `solver` related to the algorithm used during the optimization problem. Both the standardization and hyperparameter tuning have been performed using `scikit-learn Python` module.

2.8. Deep transfer learning approaches

One of the main objectives of this project was to evaluate the performance of a DNN trained on simulated hospital activities on RHS. This lent to the implementation of *transfer learning* techniques. According to the results obtained by using the methods outlined in the previous sections, the domain switch between SHS and RHS was broad. The features extracted by the DNN trained on SHS were not representative enough for some inpatient activities (especially Walking and (self-propelled) Wheelchair). Hence, transferring knowledge acquired from SHS data and adapting it to the target hospital domain might have been beneficial. An additional advantage of using this approach consisted in not fully retraining the *ccpd2* DNN on the hospital dataset. In this way, the issues related to activity distributions between inpatients (see cross-validation results in table 3.2) might have been mitigated.
This section is divided into three parts. The first addresses some basic concepts related to *transfer learning*. The second introduces the different settings and strategies used to implement such technique. Eventually, an in-depth kernel transfer analysis has been carried out to investigate about the information brought by each convolutional layer during the transfer process .

2.8.1. General concepts

Traditional (semi-)supervised deep-learning classification models are tailored to make predictions on future testing data by relying on the knowledge acquired from (partially) labeled training data. A pre-requisite of such methods is that the distributions of both labeled and unlabeled data should be the same. However, this is is not the case in many of the practical real-world scenarios. *Transfer learning* instead allows the domain, tasks and data distribution between train and test set to be different. More in detail, *transfer*

learning allows to improve the learning of the predictive function on a target domain by using the knowledge acquired from the source domain and task [33]. Referring to our use-case, the aim is to find meaningful feature representations to minimize divergence between SHS and RHS. In parallel, the HAR classification error should be be minimized leveraging on the great availability of labeled data on both source (SHS) and target (RHS) domains. Thus, the under-exam *transfer learning* approach can be categorized as *Inductive Transfer learning* based on feature-representation-transfer. The term "*inductive*" indicates that part of the target domain is treated as training data to *induce* the target predictive function.
The HAR field is not stranger to this approach. In fact, it is quite common that deep-learning models developed to tackle a specific activity recognition task might be adapted and re-used for a different one by taking advantage of the connections between source and target domain. This setting is defined as *transfer-based activity recognition.* When performing the latter, several factors might represent the causes for HAR domain shift: sensor modality, labelling strategy, data acquisition protocols and so on [9].

2.8.2. Canonical deep neural network transfer learning

Referring to the current use-case, *ccpd2* DNN pre-trained on SHS has been used as a *feature extractor* block (also called "backbone"). Then, those feature have been *fine-tuned* to hospital activity data. This procedure could be carried out by following the steps below.

1. Retraining the *ccpd2* baseline model on SHS data.
2. Freezing all layers.
3. Detaching and replacing the final *ccpd2* fully-connected classification layer with a custom one (or block).
4. Retraining the tweaked *ccpd2* DNN on target hospital data.

Step 2 consisted in disabling the weight update for all parameters contained in *ccpd2* layers. Step 3 was required to change the number of neurons of the last baseline DNN fully-connected layer so to match the number of RHS activity classes (i.e. 5). During this step, the final dense layer could optionally be preceded by multiple fully-connected layers creating a *classification block.* By doing so, the extracted features (also called "bottleneck features") were gradually processed before being mapped to the final output target RHS activity class. Step 4 consisted of the *fine-tuning* procedure. Within this stage, a number of layers frozen during step 2 has been unfrozen in order to adapt the model weights to information brought by hospital activity data. In case a batch-normalization got unfrozen, it was essential to set it in *inference mode* so that its mean and variance parameters learned during stage 1 were not updated.

2.8.3. Kernel transfer analysis

A detailed analysis concerning the feasibility of transferring feature representations has been carried out. To achieve this, the same footsteps outlined in a previous study have been followed [31]. Practically speaking, two separate high-level *transfer learning* scenarios have been considered.

1. Knowledge transfer between users within the same domain.
2. Knowledge transfer between different domains.

The core idea behind this approach was to determine if specific convolutional kernels were able to grasp either generic or domain-specific feature representation. By creating a copy of the baseline DNN model (defined as *transferred model*) it was possible to gradually transfer pre-trained kernel parameters to it and adapting the DNN replica by training them on a target domain subset. After statistically comparing the results obtained with the *transferred model* and the original baseline DNN (called *source model*), useful insights on the overall knowledge transfer process could be derived.

Technical implementation

Translating the above-mentioned process to the current task, case 1 referred to different RHS inpatients whereas 2 to the feature representation transfer from source SHS to target RHS domains.
The following technical details were shared among the two *transfer learning* settings.

- The *source model* has been trained on 90% of the whole *source* dataset and validated on the residual 10% of it .
- N [1..4] convolutional layers were transferred from *source* to *transferred* model and frozen. The remaining layer parameters have been randomly initialized and tuned to a portion of the target dataset defined as *adaptation subset*.
- The *transferred model* has been trained on different percentages (50%, 75%, 100%) of the *adaptation subset*.

Concerning the training process, the *source model* and *transferred model* have been both trained for 50 epochs. To avoid quick overfitting on RHS data, the learning rate has been

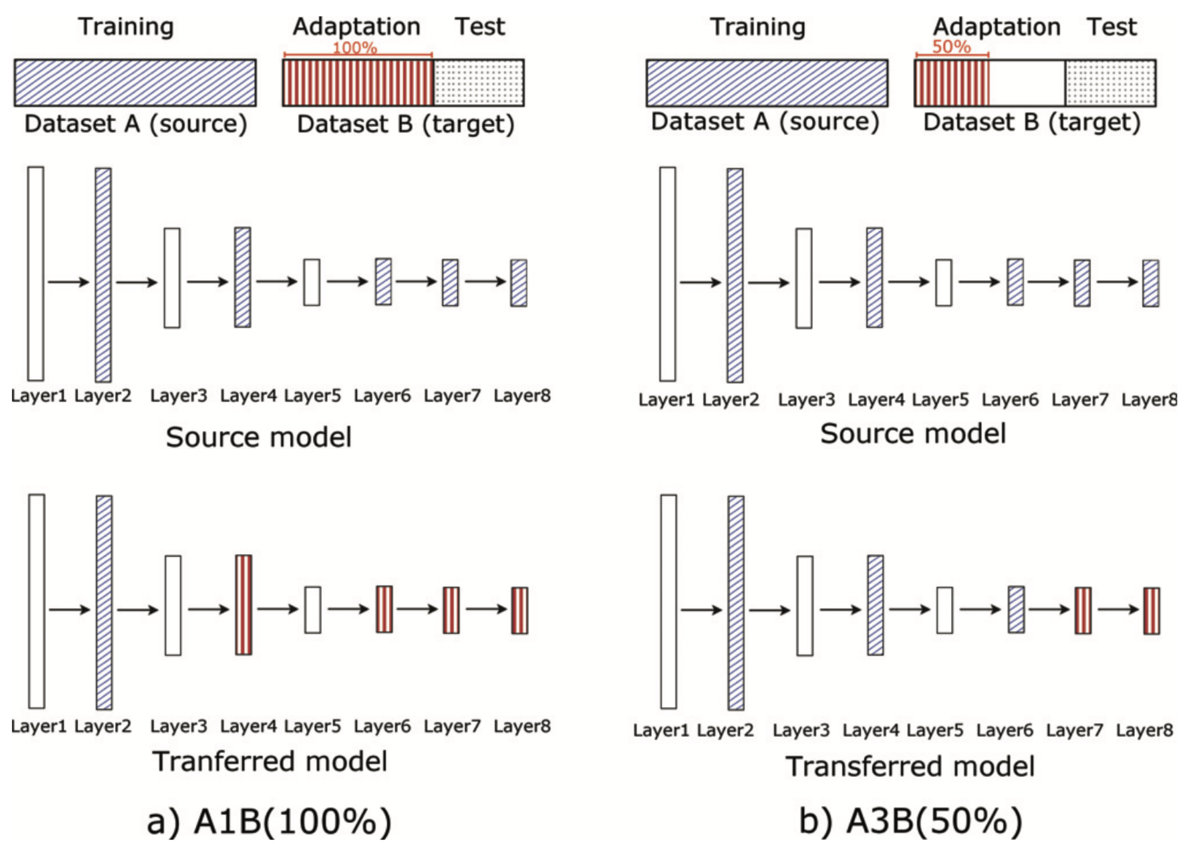


Figure 2.18: Kernel transfer analysis general scheme

decreased to 10^{-7} during *fine-tuning*.

Fig. 2.18 graphically summarizes what outlined so far for a generic DNN architecture. Each kernel transfer setting was identified by a codename: AXB(N%) where A and B indicated the source and target domain, respectively, whereas N% to the portion of *adaptation subset* used to *fine-tune* the transferred kernel parameters. Referring to the current use-case, S and R have been used to denote SHS and RHS in place of A and B, respectively.

As a side comment, it is valuable to remark that, due to the high class imbalance in both the source and target domain activities, the splitting procedure has been performed with *stratification* in order to preserve label distribution between dataset splits.

2.9. Domain-adversarial neural networks

The approach described in this section consisted of a specific setting of knowledge transfer: *domain adaptation*. The latter, by referring to the formal definitions of the various *transfer learning* configurations introduced in [33], can be identified as a special case of

transductive transfer learning. The term "*transductive*" refers to the fact that (part of) the unlabeled *target* data should be available during the training process to compute its marginal probability distribution. *Domain adaptation* assumption relates to the difference between the marginal probability distributions of *source* and *target* data involved into the same *learning task*
Within the context of DNNs, it is possible to perform *domain adaptation* by means of *domain-adversarial neural networks* (DANNs) [16]. Those architectures can be developed on the basis of any pre-existing DNN architecture by adopting almost effortless structural changes to it. Such feature resulted particularly suitable for the RHS activity recognition task in which the baseline model consisted of a DNN.

2.9.1. Motivation

The main advantages of performing *domain adaptation* from *source* (SHS) to *target* (RHS) domains via DANNs consists of two key reasons.

1. The predictions are made based on features that cannot discriminate between the training (*source*) and test (*target*) domains.
2. The training process relies on labeled data from the *source* domain and unlabeled data from the *target* domain.

Commenting point 2, labeled *target-domain* data are not strictly required, configuring *unsupervised domain adaptation* task. Nevertheless, it is indeed possible to reveal a portion of labeled *target* data (if available) during the training process. The latter scenario can be defined as *semi-supervised domain adaptation.* [16] uniquely focused on the more challenging *unsupervised domain adaptation* setting. However it has been stated that DANNs can be easily generalized to the *semi-supervised* case.
To summarize, DANNs promote the emergence of features that are both *discriminative* for the main predictive learning task and indiscriminate with respect to the shift between the domains.

2.9.2. Technical implementation

The *ccpd2* baseline DNN model has been tweaked in order to be considered as a DANN. Fig. 2.19 shows the three main components of a DANN structure: the *feature extractor* (in green), the *label predictor* (in blue) and the *domain classifier* (in pink). The first two are rather straightforward to address since they are already present on almost every pre-existing DNN model devoted to automatic feature extraction and classification. In the

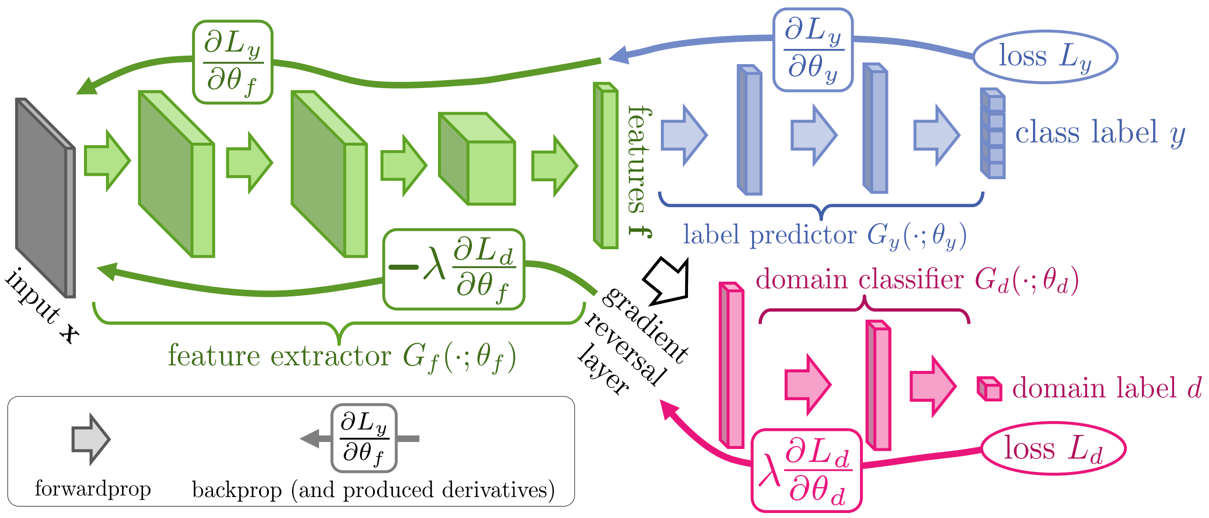

$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features f
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 2.19: Domain-adversarial neural network structure

case of *ccpd2* model, feature extraction is carried out by a series of 1D convolutional layers generating bottleneck features (i.e. *features f* in figure 2.19). The classification block is instead constituted by the respective final fully-connected layer before the 5-neurons output one.

The *domain classifier* (or *domain discriminator*) module has been added to the original *ccpd2* model architecture. It consisted of two fully-connected layers (256 and 128 units, respectively) interposed by dropout layers (ratio: 30%) and followed by the single-neuron dense output layer activated by a *sigmoid* function. This final layer discriminates if the extracted features are either related to the *source* or *target* domain. Since the domain classification task only involves two classes (1 or 0 for *target* or *source* samples, respectively), the corresponding L_d loss is a binary cross-entropy loss function. On the other hand, the L_y loss used for the *label predictor* block consists of the already-implemented (multi-class) cross-entropy loss.

An attractive feature of DANN model is that its parameters are updated during the training process by using standard backpropagation. Referring to Fig. 2.19, the produced *label* and *domain* predictor loss derivatives are represented by $\frac{\partial L_y}{\partial \theta_y}$ and $\frac{\partial L_d}{\partial \theta_d}$, respectively, where θ indicates the parameter vector. By following the corresponding backpropagation flows (indicated by the arrows) it now becomes clear the crucial role that the novel gradient reversal layer (GRL) plays. Such layer is interposed between the *feature extractor* and *domain classifier* blocks. By changing sign to the gradient produced by the *domain* predictor and scaling it with a λ *domain adaptation* parameter, its negative contribution on the *feature extractor* helps in generating domain-agnostic features. At the same time, the information contained in the gradient produced by the *label predictor* concurs to the generation of meaningful features for the activity recognition task. On the other hand, the gradient transferred from the *feature extractor* to the *domain classifier* is not altered by the GRL. The GRL layer has been implemented by relying on a useful *Tensorflow* `custom_gradient` decorator. This allowed to customize the upstream and downstream gradients computations.

```
import tensorflow as tf

@tf.custom_gradient
def gradient_reverse(x, lambda_p):
    # Forward pass
    y_forward = tf.identity(x)

    # Backward pass
    def custom_gradient(dy):
        return -dy*lambda_p, None
    return y_id, custom_gradient
```

In addition, two hyper-parameters values schedulers have been implemented. The first aimed at promoting convergence and low error on the *source* domain and concerned the μ *learning rate*:

$$\mu_p = \frac{\mu_0}{(1 + \alpha \cdot p)^\beta} \tag{2.1}$$

where p represents the training progress (changing from 0 to 1), $\mu_0 = 0.01$ the initial *learning rate* value and $\alpha = 10$ and $\beta = 0.75$ two constants.
The second scheduler related to the λ *domain adaptation* parameter, linearly changing from 0 to 1 according to the following equation:

$$\lambda_p = \frac{2}{1 + exp(-\gamma \cdot p)} - 1 \tag{2.2}$$

where $\gamma = 10$. By gradually increasing λ_p, the *domain classifier* became more robust to noisy inputs during the early stages of the training process, ensuring a smooth *domain adaptation*. As a side note, the λ_p has been only used to update the *feature extractor* component whereas the *domain classification* block adopted a unitary λ *domain adaptation* parameter to train as fast as the *label predictor*.
Concerning the dataset splitting strategy has been performed by splitting both the *source* and *target* data in *train* and *validation* subsets with a 90%-10% ratio. The batch generation system has been designed so that each data batch used during the training process was composed of 128 elements. Half of them consisted of labelled *source* SHS data and the remaining half to unlabeled *target* RHS data. Within the *unsupervised-learning set-*

ting, the labeled half was routed to the *label predictor* whereas the whole batch was fed to the *domain discriminator* module. All DANN configurations tested at this stage have been trained for 150 epochs with early stopping enabled (patience: 50 epochs). The best DANN model state (i.e. lowest validation loss on *source* validation set) has been saved at the end of the training process.

Caveats for the current use-case

Due to the label set differences between SHS and RHS, the same actions on the RECLINED and *Stairs walking* classes as explained in the DNN validation subsection 2.7.3 have been taken. Thus, the DANN architectures developed in this stage classified 7 separate activity classes: LYING, RECLINED, UPRIGHT, STAIRS ASCENT, STAIRS DESCENT, WHEELCHAIR, WALKING.

Focusing on class imbalance within *source* and *test* domains, a resampling strategy has been tested when generating data batches. The latter has been implemented via the already-implemented (for DNN training) *Imbalanced-learn* `RandomOversampler`. In alternative, the complementarity between SHS and RHS labels distribution allowed to not use any resampling technique withing a *semi-supervised learning* scenario.

Referring to dataset splitting, *training* and *validation* data subsets have been generated from both *source* (SHS) and *target* (RHS) activity data. The splitting process has been performed a per-subject basis. As a result, the DANN training involved four different data subsets:

1. *source_ train* (18 SHS subjects)
2. *source_ validation* (2 SHS subjects)
3. *target_ train* (6 RHS patients)
4. *target_ validation* (6 RHS patients)

Concerning *target* subsets, the splitting ratio was adjusted to 50%-50% to avoid the risk of creating a non-representative *target_ validation* data subset. The same does not apply to SHS data because of the activity protocol involved in the related acquisition procedure.

Throughout the DANN experiments reported in the corresponding Results subsection 3.5, the *learning rate* annealing routine described in equation 2.1 has been either preserved or disabled. The latter option avoided that the DANN model could get stuck in local minima during its training process (as reported in [16]).

3 | Results

3.1. Data analysis

3.1.1. Basic data statistics and first insights

Most of the HAR models adopted in this project relied on large quantities of data. Thus, it resulted challenging to perform an in-depth analysis on the whole labeled dataset. Nevertheless, it was possible to carry out an high-level analysis concerning general properties characterizing the use-case setting.
The first examined feature consisted of the label distribution within the hospital activity dataset. Although elementary, this investigation was crucial to be performed before implementing algorithms that suffered from unbalanced labels during their training phase. As a consequence, the percentual distribution of each label along with the corresponding average duration (in hours) has been calculated. Fig. 3.1 only reports the top-8 most occurring labels for visualization purposes. The label distribution is heavily skewed towards static postures and featured few WALKING segments.
As already mentioned in the labelling strategy subsection 2.2.1, contextual information has been annotated in conjunction with posture or activities. For the purposes of this project, only the context-sensitive `In bed` and `Patient care` labels have been quantified. The latter may have possibly represented the cause for interferences on ECGMove4 acceleration samples. Realistically, such events should not often occur. The obtained results have been expressed in percentiles $(P_{25\%}, P_{50\%}, P_{75\%})$:

- `In bed` : $(66.59\%, 79.56\%, 83.15\%)$
- `Patient care` : $(3.58\%, 5.82\%, 8.35\%)$

The next step was to further investigate on the label imbalance between dynamic and static activities. The latter kind has been generalized as *Inactive* labels comprising: `In bed`, `In (wheel)chair`, `Supine`, `Upright`, `Reclined`, `Lying left`, `Lying right`, `Bend forward`, `Prone`, `Standing/Held upright`, `Patient care`, `Capnography session` and `Transport in wheelchair`. Such *Inactive* labels accounted for 85.48±5.76% (*mean*±*std*

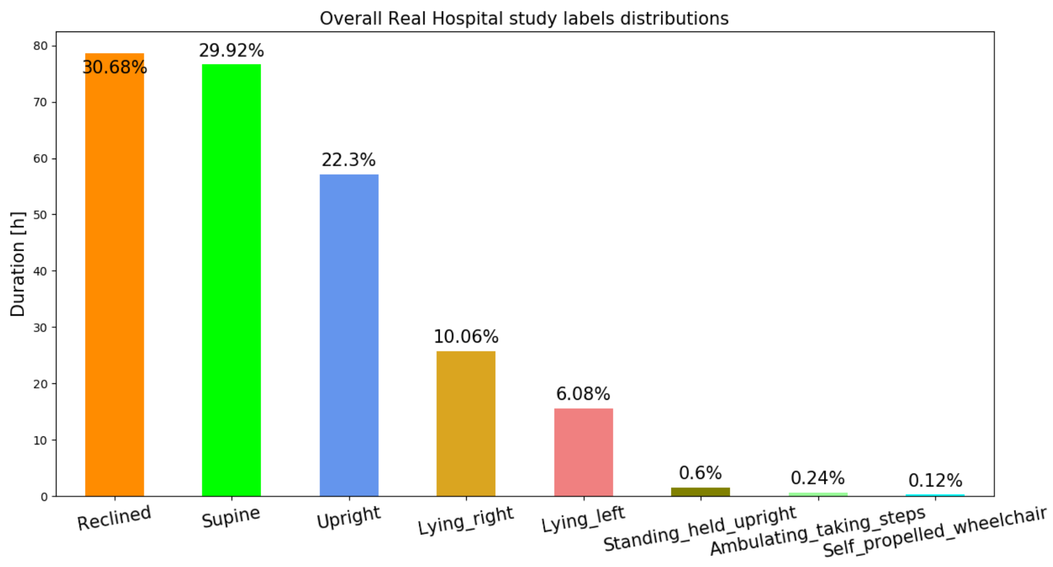


Figure 3.1: Top-8 most occurring activity labels

over all patients).
An additional cause for possible interferences with sensor readings has been identified with posture transitions (e.g. switching from Reclined to Upright while `In_bed`). As expected, those segments occurred for less than 1% on average during the 24-hour recording session for each patient.
As a concluding note, it has been observed that only 3 (i.e. P06, P07 and P09) out of the 12 hospitalized patients consistently used walking aids (i.e. cane, roller, walker or IV pole) to move around the room.

3.1.2. Acceleration values and BMI correlation

Referring to the experimental protocol subsection 2.1.2, it has been outlined that two movisens® devices have been placed on patient's upper and lower chest. Such body locations are quite distinct and their anatomical shape heavily depended on patient's gender and BMI. As a consequence, the 3D acceleration baseline values (`acc_x`, `acc_y`, `acc_z`) for the same posture might strongly vary between patients due to the above-mentioned factors. Thus, a correlation analysis between BMI and 3D acceleration median values on both wearables has been performed. The rationale behind this procedure was that inpatients with higher BMI should systematically present more remarkable differences between the 3D acceleration values acquired by *Upper* and *Lower* ECGMove4 when `Upright` or `Supine` in comparison to patients with lower BMI.
After performing a detailed analysis (including grouping by gender), no clear correlation

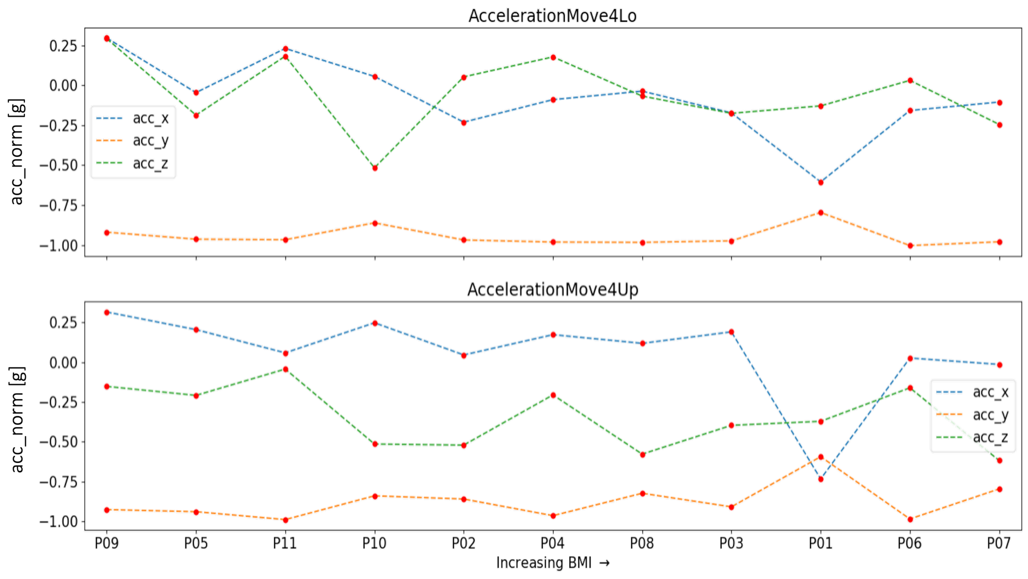


Figure 3.2: Correlation between body mass index and median 3D acceleration values

pattern has been found. However, some byproduct insights have been obtained at the end of this analysis. In fact, observing Fig. 3.2, it could be noticed that P01 showed an abnormal negative median `acc_x` value compared to the positive one for all the other patients when `Standing/Held upright`, especially for the *Upper* location (bottom pane). Without any information available on the P01 case report form, it has been concluded that the ECGMove4 devices might have been placed more left-laterally in this case. The encountered abnormality may have been emphasized by the quite high P01 BMI.

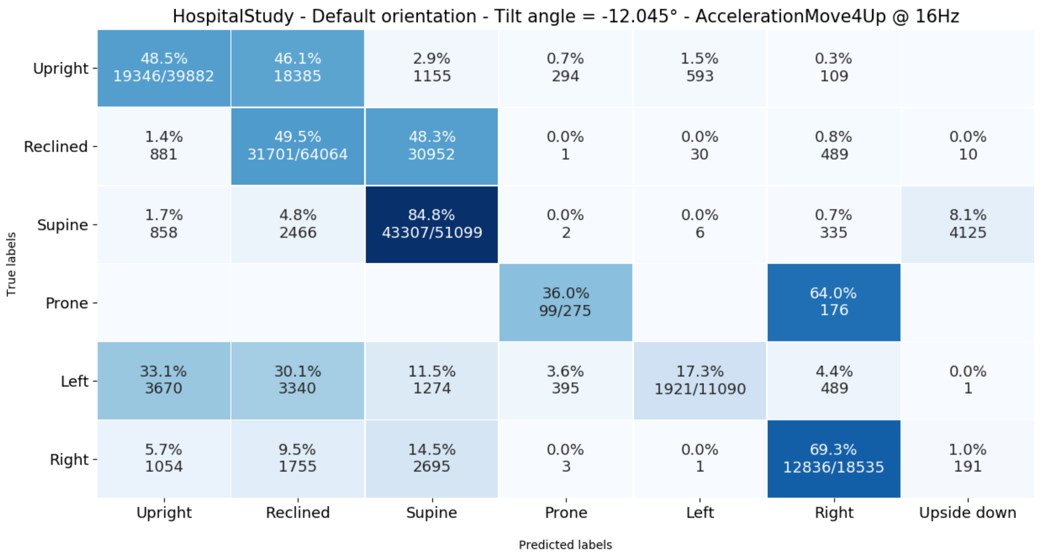
3.2. CPA

It has been verified that the CPA *Calibration* procedure failed for all RHS patients. A thorough explanation on the causes for this behaviour has been outlined in the Discussion chapter 4.

As one would expect, the *Posture* module optimally performs after a successful *Calibration*. However, the only option was to choose an a-priori and static α tilt angle. The latter should be simultaneously optimal for all patients. Naturally, this task is unfeasible and only a sub-optimal α value has been found at the end of this investigation.

3.2.1. Testing and optimal configuration

The first experiment consisted in running the CPA algorithm with a default configuration on the whole RHS dataset. Fig. 3.3 depicts a confusion matrix obtained by evaluating



	Upright	Reclined	Supine	Prone	Left	Right	Upside down
Upright	48.5% 19346/39882	46.1% 18385	2.9% 1155	0.7% 294	1.5% 593	0.3% 109	
Reclined	1.4% 881	49.5% 31701/64064	48.3% 30952	0.0% 1	0.0% 30	0.8% 489	0.0% 10
Supine	1.7% 858	4.8% 2466	84.8% 43307/51099	0.0% 2	0.0% 6	0.7% 335	8.1% 4125
Prone				36.0% 99/275		64.0% 176	
Left	33.1% 3670	30.1% 3340	11.5% 1274	3.6% 395	17.3% 1921/11090	4.4% 489	0.0% 1
Right	5.7% 1054	9.5% 1755	14.5% 2695	0.0% 3	0.0% 1	69.3% 12836/18535	1.0% 191

Figure 3.3: CPA performance with default parameters

CPA with no parameter adjustments. Since the `Upside down` posture is not part of the label set reported in Fig. 2.10, the corresponding confusion matrix row (i.e "true labels") has been omitted.

As expected, the overall CPA performance resulted modest. The next step was to understand the rationales behind every misclassification category.

The first consideration concerned the *Upright-as-Reclined*. It has been found that the reference tilt angle used to define RECLINED and its related CPA posture vector was 60°. This angle constituted an edge-case (i.e. between RECLINED and UPRIGHT) according to the corresponding definitions used for this study (see label set description table 2.2). The *Reclined-as-Supine* case followed the same line of reasoning. In fact, the CPA *Supine* definition was set at 30°which corresponds to a *Supine-Reclined* edge-case for the adopted labelling strategy.

Next, the *Prone-as-Lying right* misclassifications were examined. The *Prone* posture rarely occurred within the RHS dataset and usually deviated from its ideal position. Indeed, it was way more common that patients might have laid in an intermediate position (i.e. between `Lying right/left` and `Prone`) for comfort reasons, especially while sleeping.

Finally, the *Lying left-as-Reclined/Upright* instance was an already-known issue for the adopted CPA implementation. This behaviour has been documented and analyzed by a Philips colleague. The root cause consisted in a remarkable skin deformation around the *Upper* sensor location when a subject was *Lying left*. The devised solution was to update

α (°)	Posture % accuracy (with updated posture vectors)					
	Upright	Reclined	Supine	Prone	Lie left	Lie right
-29	**76.3** (63.3)	**81.9** (51.1)	49.1 (38.8)	0 (0)	21 (50.1)	65.22 (64.9)
-18	58.1 (44.5)	68.6 (43.6)	**89.3** (65.3)	18.9 (0)	18.6 (62.7)	94.6 (**96.3**)
-15	54 (63.3)	56.4 (51.1)	84.8 (38.8)	28.8 (0)	18.3 (50.1)	65 (64.9)
-10	46.6 (33.9)	45.5 (36.1)	84.2 (83.7)	36.9 (0)	15.9 (53.9)	65 (65.4)
-5	39.7 (26.2)	33.5 (29.5)	82.6 (85.2)	37.6 (0)	15.2 (59)	64.8 (65.6)
0	31.6 (20)	21.6 (23.3)	81.9 (84.3)	**38** (0)	15.6 (**74**)	64.5 (65.6)
5	24.4 (14.9)	15.9 (14.3)	82 (82.5)	18 (0)	18 (66.4)	64.2 (65.3)
10	18.7 (9.7)	10.7 (10.2)	79.3 (82)	**38** (0)	19.3 (64.9)	64 (65.1)
15	12.2 (5.2)	1.2 (3.7)	76.6 (88.3)	37.5 (0)	20 (69.8)	93.3 (95)
18	8.8 (4)	0.7 (1.1)	74.3 (87.6)	36.4 (0)	20.7 (65.6)	93.4 (93.4)
29	3.5 (0.5)	2.5 (1.6)	60.9 (69.6)	0.4 (0)	19.5 (50.4)	64.1 (65.3)

Table 3.1: CPA configurations and postures detection accuracy

the reference posture vectors to mitigate this confusing factor.
Without the aid of a *Calibration* step, the adopted strategy was evaluating CPA performance by selecting a range of α tilt angles in combination with original and updated posture vectors. Table 3.1 summarises the percentage accuracy obtained for each posture using the CPA *Posture* detection module with different configurations. Results obtained with the updated posture vectors are included within the parentheses whereas the best scores for each posture are indicated in bold. What follows is an interpretation of the results obtained with different CPA parameters. By increasing the α orientation tilt angle from the negative reference value (-12.045°) towards positive ones both the `Upright` and `Reclined` were increasingly not recognised. Instead, the `Supine` condition was fairly often well-detected but suffered from extreme *alpha* tilt angles (i.e. ±29°). The same applied to `Prone` although the accuracy scores were way lower in absolute value than the ones obtained for the `Supine` posture. Concerning `Lie left`, the results obtained with the updated posture vectors outperformed the scores achieved with default ones under the same α tilt angle. This finding proved the efficacy of the identified updated posture vectors. However, the trade-off consisted in a worse detection of the remaining postures. Eventually, the `Lie right` has been optimally identified for $\alpha = \pm 18°$.
The choice of the suboptimal α tilt angle has been made upon these assumptions. As illustrated in the label distribution Fig. 3.1, RHS labels were heavily skewed towards `Upright`, `Reclined` and `Supine`. Thus, $\alpha_{opt} = -18°$ has been identified as the angle providing the best performance for the above-mentioned postures. Following the same rationale, the default posture vectors have been preferred over the updated ones since

`Lie left` consisted of a minority class. To summarise, $\alpha_o pt = -18°$ with original posture vectors has been picked as the suboptimal CPA configuration for RHS posture detection.

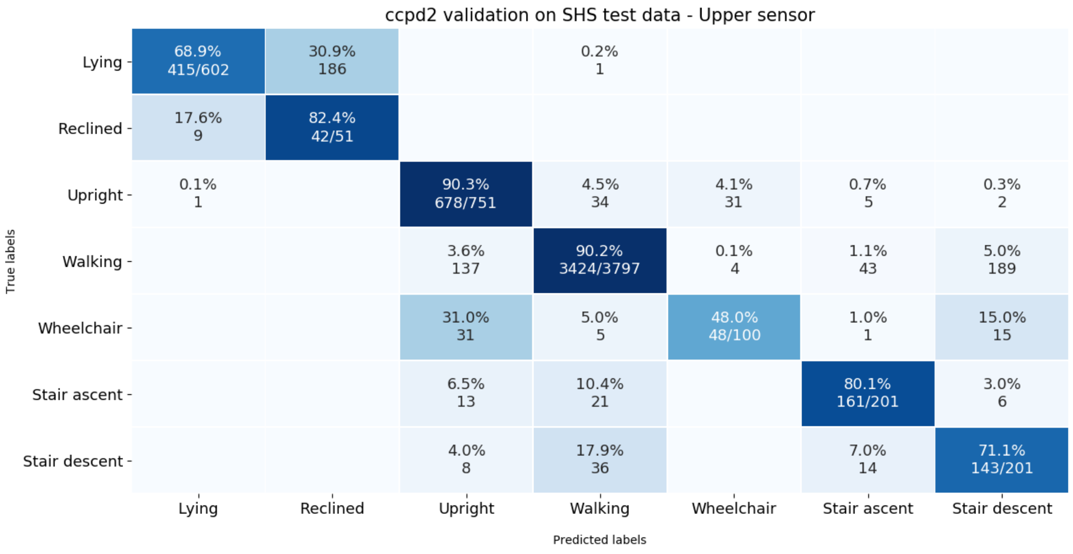
3.3. Deep-learning techniques

3.3.1. Baseline model validation

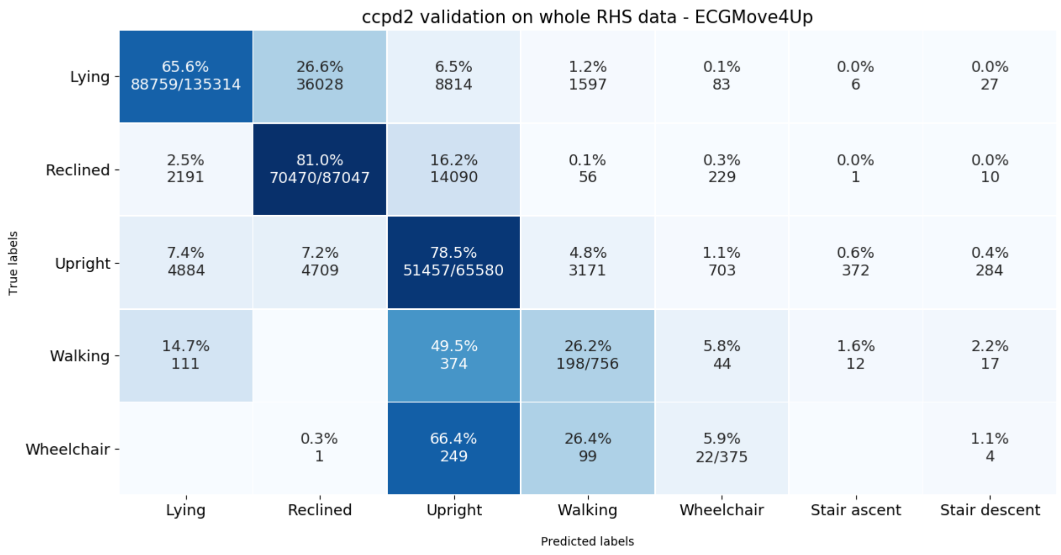
As explained in the baseline DNN subsections 2.7.2 and 2.7.3, some adjustments concerning model architecture and labelling needed to be implemented prior to its validation on RHS data. Fig. 3.4a and 3.4b depict two confusion matrices obtained by validating the *ccpd2* model on the SHS test set and the whole hospital study dataset respectively.
Commenting the former case, the introduction of a novel Reclined class in the SHS dataset increased the *Reclined-as-Lying* misclassifications. The absence of confusion between Reclined and Upright was because Reclined has been defined as a bed tilt angle between 30°and 45°within SHS. Indeed, this condition could be easily confused as Supine. In addition, a remarkable drop in Wheelchair recognition has be appreciated. This can be partially explained by referring to the DNN instability mentioned in baseline DNN subsection 2.7.2. This behaviour might also apply for the *ccpd2* model. However, since this activity class represented the least-occurring within RHS, no further investigation has been performed to understand the rationale behind this behaviour.
Concerning Fig. 3.4b, the most-occurring static postures (i.e. Lying, Reclined and Upright) were fairly well recognized. On the other hand, both the Walking and (self-propelled) Wheelchair labels are almost always erroneously identified as Upright.
As a concluding note, since both the Stair ascent and Stair descent were not part of the RHS label set (see label set Fig. 2.10), the corresponding rows of Fig. 3.4b have been removed to improve its visualization.

3.3.2. Baseline model retraining on hospital data

Fig 3.5 shows the baseline *ccpd2* performance when trained and validated on RHS data. At first sight, the Reclined posture was quite often incorrectly classified as Lying. This might have been linked to two possible root causes: the overall majority of Lying segment over *Reclined* ones and subtle bed tilt angles.
Concerning *Upright-as-Wheelchair* and vice versa, it can be supposed that the Wheelchair samples featured by hospital study test patients were not representative enough. Thus, the baseline DNN failed in extracting meaningful patterns and features to differentiate the two conditions.

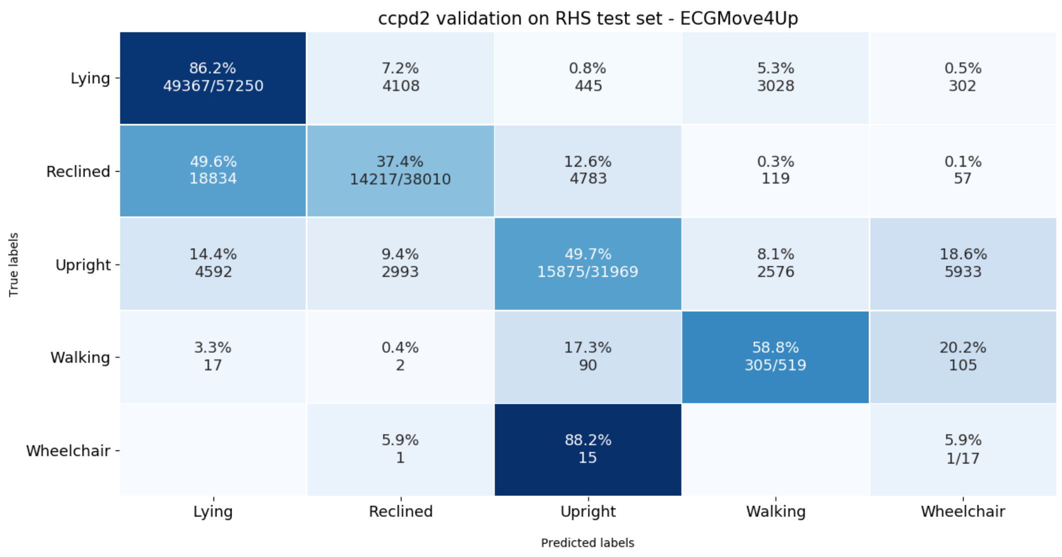


(a) Baseline deep neural network validated on simulated hospital study



(b) Baseline deep neural network validated on real hospital setting

Figure 3.4: Baseline deep neural network validation performance comparison between simulated and real hospital study



	Lying	Reclined	Upright	Walking	Wheelchair
Lying	86.2% 49367/57250	7.2% 4108	0.8% 445	5.3% 3028	0.5% 302
Reclined	49.6% 18834	37.4% 14217/38010	12.6% 4783	0.3% 119	0.1% 57
Upright	14.4% 4592	9.4% 2993	49.7% 15875/31969	8.1% 2576	18.6% 5933
Walking	3.3% 17	0.4% 2	17.3% 90	58.8% 305/519	20.2% 105
Wheelchair		5.9% 1	88.2% 15		5.9% 1/17

Figure 3.5: Baseline deep neural network retrained on hospital activity data

The Walking detection was just about acceptable. Comparing this result to what obtained by validating the SHS pre-trained DNN on RHS (see Fig. 3.4b), a remarkable improvement in Walking detection can be appreciated. This might indicate that SHS and RHS Walking patterns could be quite different.
In general, the under-exam hospital setting dataset was way more complex and heterogeneous than the simulated one. To prove this statement, it is possible to observe the accuracy and loss scores for both training and validation RHS subsets. Focusing on the blue curves in Fig. 3.6, it can be noticed that the improvement in those scores was quite limited and slow across epochs.
In order to make a more robust assessment of the baseline *ccpd2* performance on such a diverse dataset, a cross-validation was deemed appropriate. In particular, a Leave-One-Subject-Out (L1SO) strategy was implemented by creating 12 unique dataset folds each containing a different hospitalized patient in its test set. The train and validation set inpatients (i.e. 9 and 2, respectively) were randomly selected. Table 3.2 reports the main performance metrics (*precision*, *recall*, *F1-score*) obtained for each L1SO fold. The best scores for each metric across folds are highlighted in bold. As expected, all the reported metrics dramatically depended on the holdout patient for a specific fold. The same values have been represented in Fig. 3.7 by using boxplots to emphasize their heterogeneity between patients within the same performance metric.

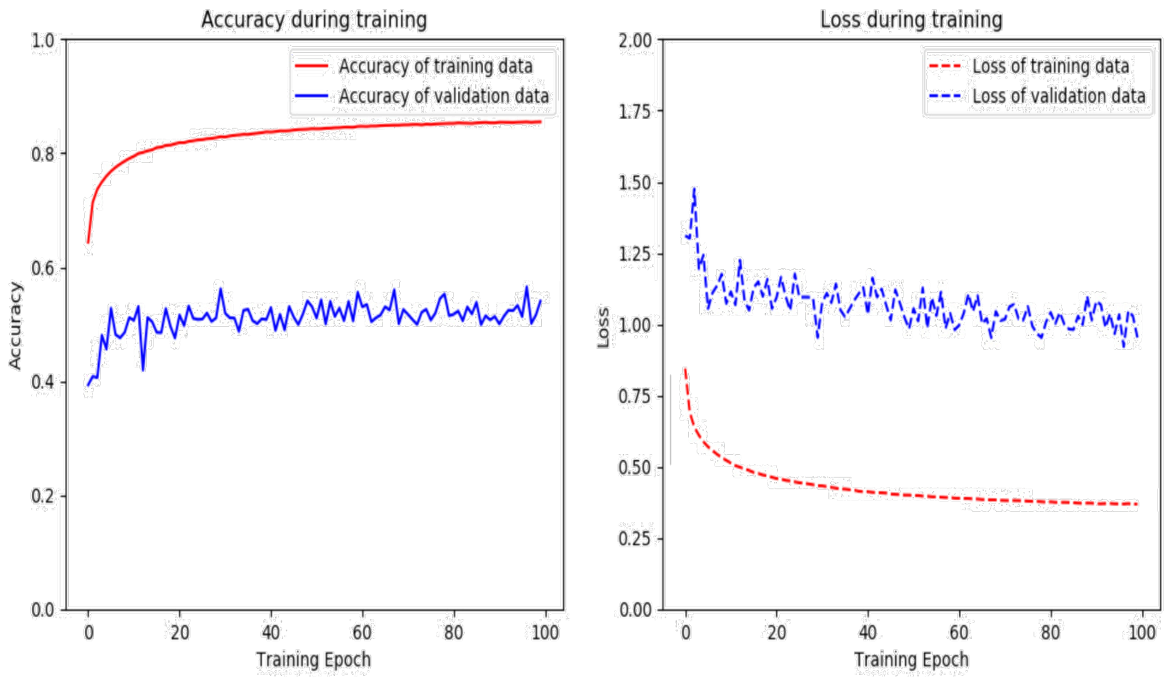


Figure 3.6: Accuracy and loss history when retraining baseline deep neural network on hospital activity data

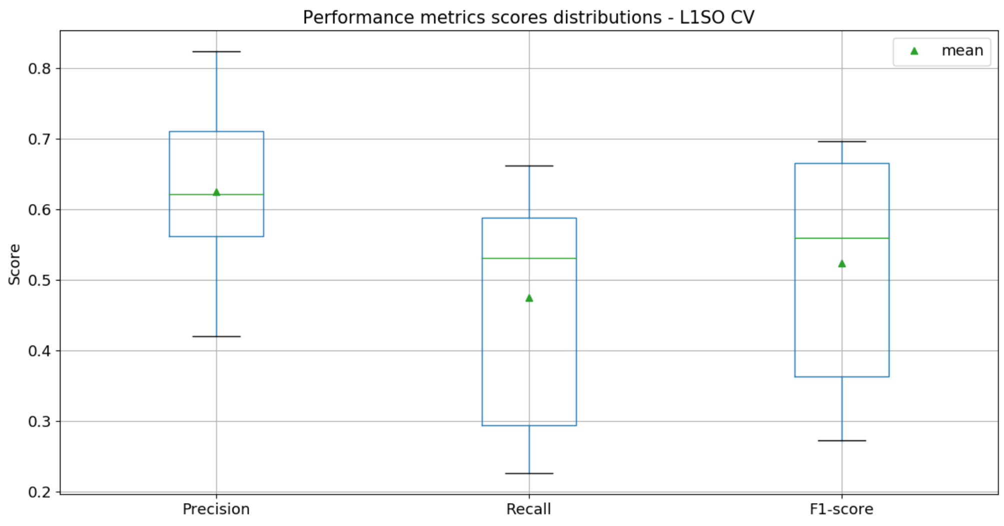


Figure 3.7: Performance metrics cross-validated scores distribution

Fold #	Precision	Recall	F1-score	Train IDs	Validation IDs	Test ID
1	0.4576	0.2865	0.331	P02, P03, P04, P05, P06, P08, P09, P10, P11	P07, P12	P01
2	0.8231	0.5684	0.6612	P01, P04, P05, P06, P07, P08, P09, P11, P12	P03, P10	P02
3	0.5942	0.4699	0.523	P04, P05, P06, P07, P08, P09, P10, P11, P12	P01, P02	P03
4	0.6532	0.5645	0.605	P01, P02, P03, P05, P06, P07, P08, P11, P12	P09, P10	P04
5	0.4198	0.226	0.2724	P01, P03, P06, P07, P08, P09, P10, P11, P12	P02, P04	P05
6	0.7101	0.6498	0.6745	P01, P02, P03, P04, P07, P08, P09, P11, P12	P05, P10	P06
7	0.6484	0.5564	0.5944	P01, P03, P04, P05, P06, P08, P09, P11, P12	P02, P10	P07
8	0.5272	0.273	0.3508	P01, P02, P04, P05, P07, P09, P10, P11, P12	P03, P06	P08
9	0.5724	0.5053	0.5201	P01, P02, P03, P05, P07, P08, P10, P10, P12	P04, P06	P09
10	0.7125	**0.6612**	0.6812	P01, P02, P03, P04, P05, P07, P09, P11, P12	P06, P08	P10
11	**0.7897**	0.6454	**0.6964**	P02, P03, P04, P05, P06, P07, P08, P09, P12	P01, P10	P11
12	0.5807	0.2959	0.367	P01, P02, P03, P04, P05, P06, P07, P10, P11	P08, P09	P12

Table 3.2: Retrained baseline deep neural network cross-validated performance

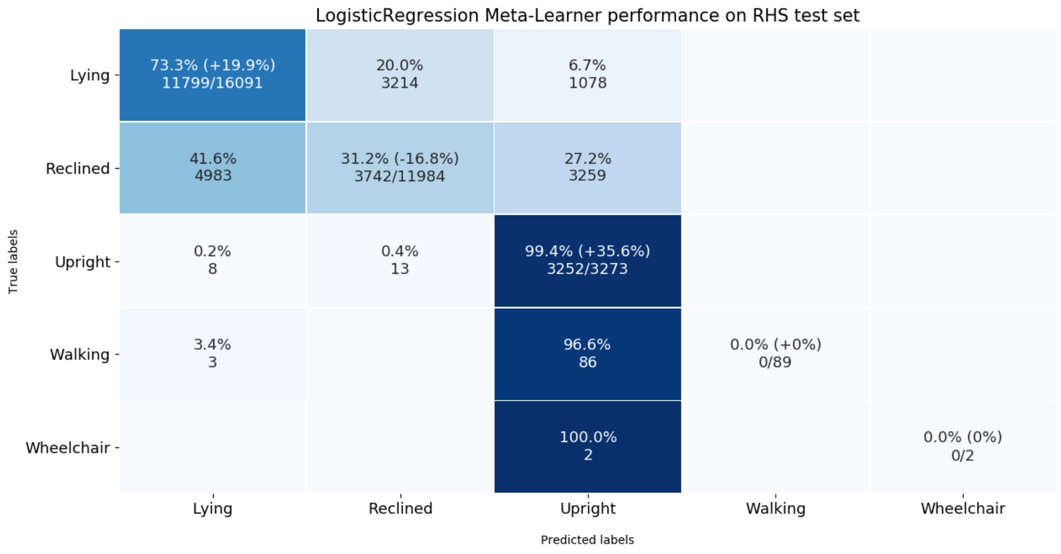


Figure 3.8: Stacking ensemble default configuration performance

3.3.3. Hybrid ensemble method

Fig. 3.8 shows the confusion matrix obtained for the initial described in the corresponding implementation subsection 2.7.5. The percentage values contained within parentheses in its diagonal refer to the scores variation with respect to the one obtained by using the *ccpd2* DNN model only under the same settings.

A preliminary consideration can be made on the WALKING and WHEELCHAIR classes. The randomly selected test patients did not feature many samples associated to those activities. Even though both LYING and UPRIGHT have been better classified compared to the baseline *ccpd2* model, the RECLINED detection degraded quite consistently. The cause for this behaviour might consists of the already-mentioned discrepancy in the definition of this label between the two studies. On top of that, the wide range of RHS RECLINED variations may contribute to this occurrence.

Next, a detailed examination of the contributions of *Base-Models* output probabilities to the final Level-1 model activity class output has been performed. Practically, a heatmap of the optimal *Meta-Model* regression coefficient has been represented. Fig. 3.9 shows the weight of each input class probability (x-axis) to determine the *Stacking* ensemble model output (y-axis). The orange and blue borders indicates CPA *deltas* and DNN probability outputs, respectively.

The first consideration concerns the negative coefficient values obtained for the CPA *deltas*. Since the latter based its output on a "minimal distance" criterion, a negative coefficient implied a consistent contribution to the *Stacking* ensemble output.

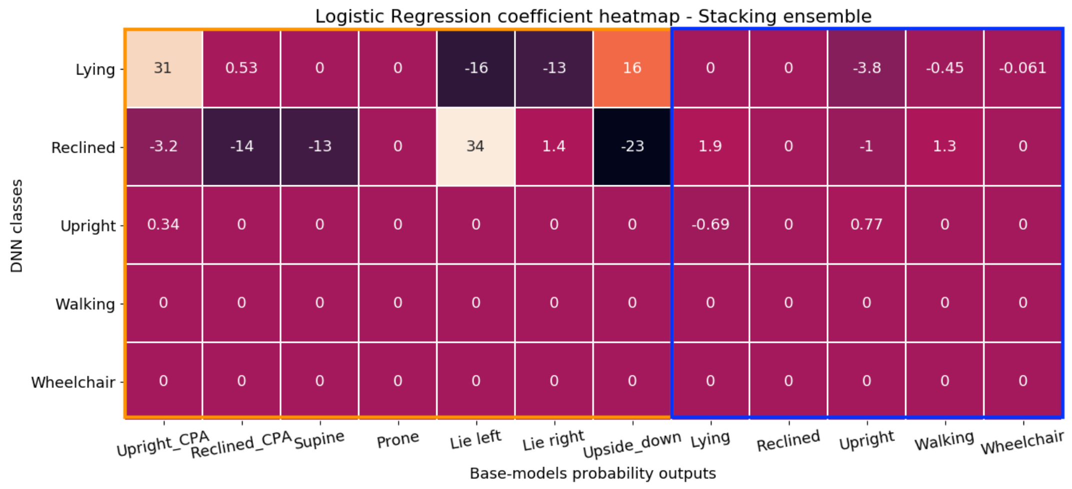


Figure 3.9: Meta-Model coefficient heatmap

Starting from the top row associated to LYING, it can be noticed that both CPA `Lie Left` and `Lie Right` *deltas* gave a substantial contribution to its classification. This was expected since such postures consists in a variation of LYING. In turn, they were more challenging to be recognized. The *Logistic regression* model was able to grasp this complexity relationship by giving utmost importance to the above-mentioned CPA outputs. Without surprise, the `Upright_CPA` regression coefficient took high positive value, meaning that the above-mentioned information acted as a negative feedback on the *Lying* classification. The same behaviour was reinforced by the negative UPRIGHT DNN probability output coefficient.

Next, RECLINED class has been examined. Due to the already-discussed variations of this posture, `Supine` CPA *delta* coefficient was quite high. On the other hand, the `Upright` CPA *delta* provided a smaller contribution to its recognition. This has been confirmed by the limited amount of *Reclined-as-Upright* misclassifications in the confusion matrix in Fig. 3.8).

What follows is a possible explanation for the CPA `Upside down` coefficients involved in the LYING and RECLINED classifications. UPSIDE_DOWN might have acted as negative contribution to LYING-like postures (especially if supine) since they are opposite conditions. The same cannot be stated for the `Upside_down` contribution on RECLINED. In contrast, it seems like that a patient with a RECLINED posture might have been closer to an `Upside_down` one. The root cause for this could be associated to peculiar sensor placement and `Reclined` posture variations (e.g. slightly lying on side).

The further step was to design a second *Stacking* ensemble configuration. Assuming that

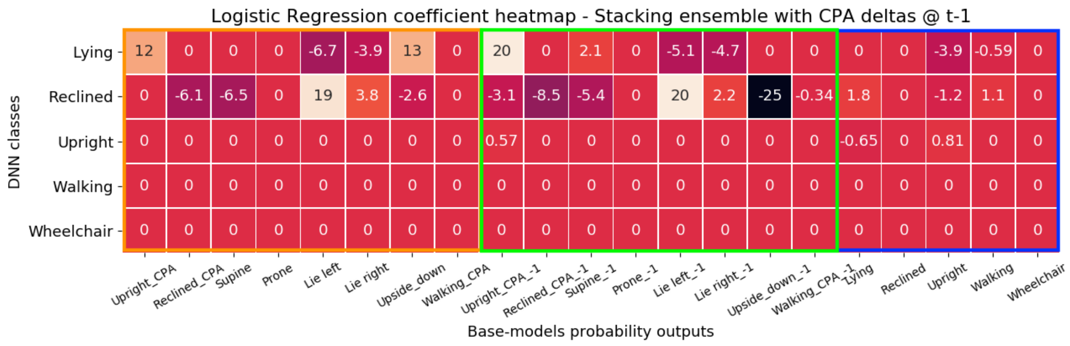


Figure 3.10: Modified Meta-Model coefficient heatmap

static postures taken by hospitalized patients may have been long-lasting, it might have been beneficial to introduce information contained in CPA *deltas* from the previous prediction. Moreover, the CPA algorithm involved a walking detection algorithm used during its *Calibration* procedure (described in subsection 2.6.1) outputting a probability score associated with such activity. Thus, the `Walking_CPA` score has been added as additional input probability feature for the Level-1 model.

The results obtained with these adjustment were practically equivalent to the previous ones. Also in this case, a further *Logistic regression* coefficient heatmap has been depicted in Fig. 3.10. The CPA *deltas* obtained at the current and previous prediction are delimited with an orange and green box, respectively. The DNN probability outputs are contained within a blue rectangle.

The coefficients associated to previous CPA *deltas* were either null or equivalent to the ones for the current timestamp. This explained why the results obtained with this configuration were equivalent to the previous *Stacking* ensemble setting. Concerning the *Walking_ CPA* feature, it appeared that the inclusion of such information was worthless.

3.4. Deep transfer learning

3.4.1. Canonical deep neural network transfer learning

Fig. 3.11a and 3.11b represent the confusion matrices associated to the performance obtained for two representative *transfer learning* configurations: unfreezing all SHS *ccpd2*

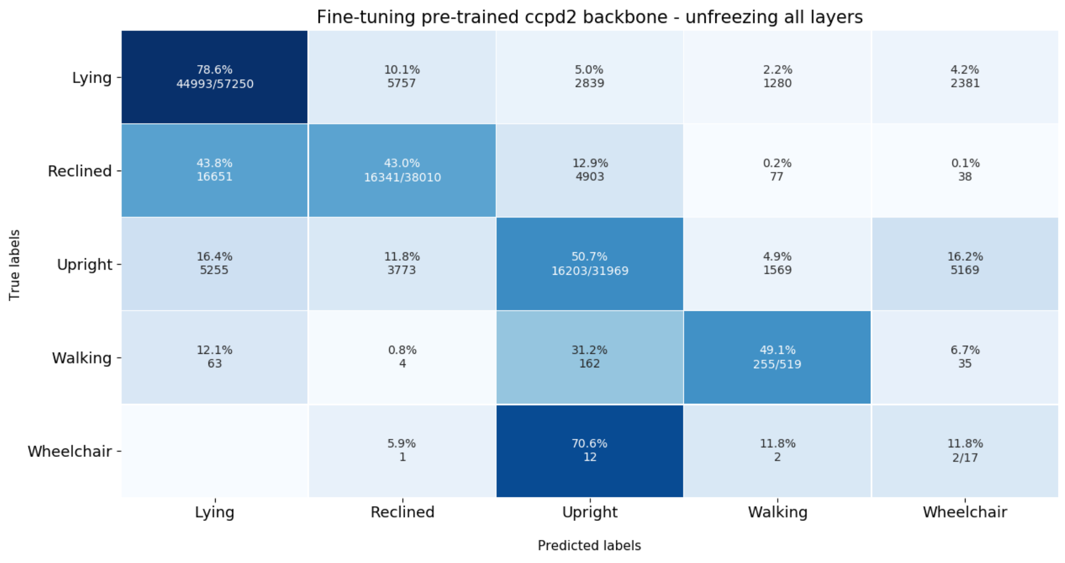
Unfrozen convolutional layers	Fully-connected block	Precision	Recall	F1-score
-	GAP	**0.68**	0.61	0.62
	1x FC	0.65	0.64	0.64
	2x FC	0.63	0.63	0.63
	3x FC	0.63	0.63	0.63
Last one	GAP	0.64	0.60	0.61
	0x FC	0.64	0.60	0.61
Last two	GAP	0.65	0.61	0.61
	0x FC	0.65	0.61	0.62
Last three	GAP	0.65	0.62	0.62
	0x FC	0.65	0.63	0.63
All	GAP	0.65	0.61	0.62
	0x FC	0.66	0.61	0.62
	1x FC	0.67	**0.66**	**0.66**
	2x FC	0.63	0.63	0.63
	3x FC	0.63	0.63	0.63

Table 3.3: Summary of transfer learning settings performance

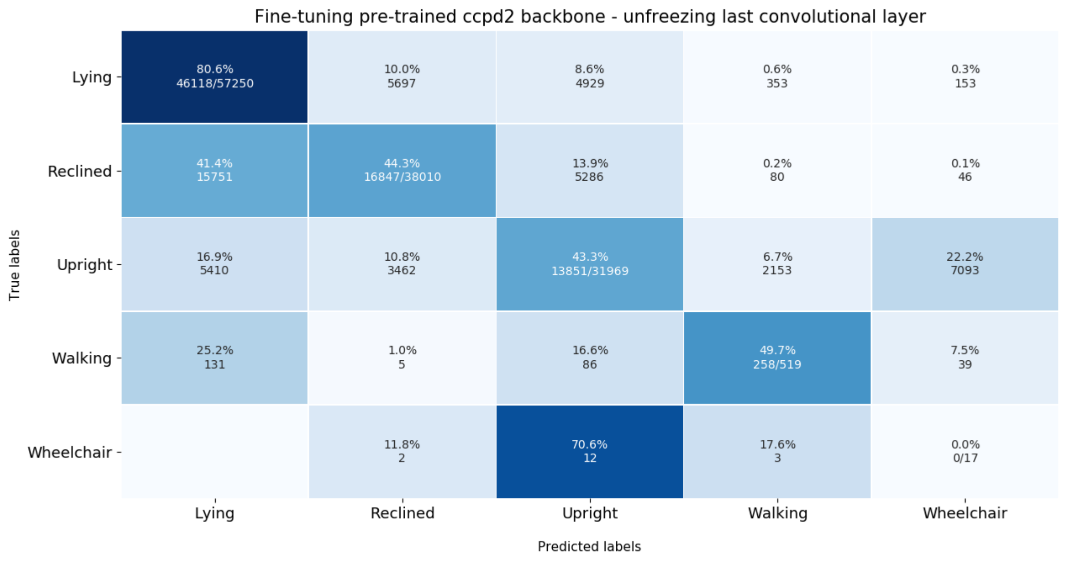
backbone layers and only the last convolutional layer before the final fully-connected one, respectively. The accuracy and loss curves for both training and validation set (blue and orange lines, respectively) obtained for the setting represented in Fig. 3.11b are shown in Fig. 3.12. Both metrics very slightly improved after the *fine-tuning* starting epoch for both the training and validation sets. As a side comment, the sharp peaks in both curves just after the beginning of the *fine-tuning* process (green vertical line) are caused by the initialization of the Adam optimizer parameters learned up to that epoch.

Table 3.3 reports the *Precision*, *Recall* and *F1-score* obtained for several *transfer learning* configurations. The first column indicates how many convolutional layers have been unfrozen during the *fine-tuning* procedure. The strategy was to gradually unfreeze backbone layers from DNN top (just before the fully-connected classification layers) to bottom (towards the input layer). As mentioned in step 3 of the *transfer-learning* procedure outlined in subsection 2.8.2, different *fully-connected blocks* configurations have been tested. The best scores for each metric have been highlighted in bold.

The codename "Nx FC" indicates how many N hidden fully-connected layers have been added. At most 3 dense layers with 512, 256 and 128 neurons interposed by dropout layers (ratio: 30%) have been sequentially included to the original DNN architecture. In this case, the bottleneck features have been flattened before being fed to the final *fully-connected block*. In alternative, a single 1D Global Average Pooling (GAP) layer [28] has been implemented in place of the fully-connected layers. The main advantage of GAP layer is that it generated one feature map for each corresponding activity output class. This was performed by averaging each feature map and feeding them to softmax layer.



(a) Performance obtained by unfreezing all backbone layers



(b) Performance obtained by unfreezing last convolutional layer

Figure 3.11: Performance comparison between two transfer learning settings

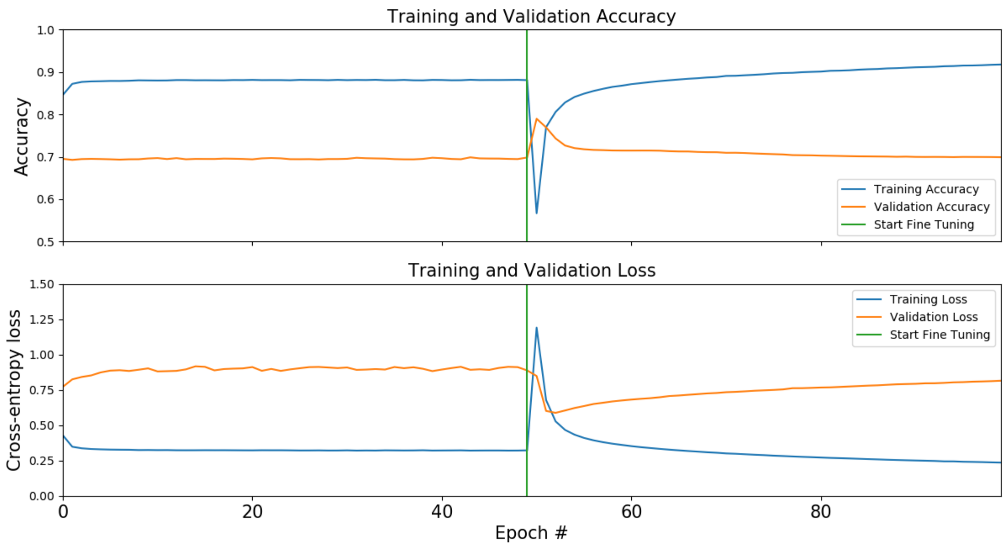


Figure 3.12: Accuracy and loss curves obtained when fine-tuning last convolutional layer

The obtained scores for each metric are quite similar independently from the *transfer learning* setting. The overall best performance has been achieved by unfreezing all backbone convolutional layers and adding a hidden 128-neurons dense layer.

3.4.2. Kernel transfer analysis

Same domain

The following results have been obtained by analyzing case 1 outlined in the *kernel transfer analysis* section 2.8.3. Namely, this consisted in transferring different features representations between different inpatients within RHS. According to the splitting ratios indicated in the implementation subsection 2.8.3, 11 hospitalized patients have been randomly assigned to the training subset whereas a single one to the *target* dataset. The latter has been further equally split (50% ratio) into *adaptation* and *test* subsets. Concerning the testing phase, a L1SO cross-validation strategy has been adopted. Fig. 3.13 shows the *F1-score* scores distributions obtained for each kernel transfer configuration. The first boxplot on the left (indicated as *RnR(0%)*) corresponds to the performance obtained by validating the *source model* without adaptation. In general, F1-scores were fairly high for all the represented kernel transfer and adaptation subset configurations.
The above analysis has been coupled with a Wilcoxon signed-rank test on the obtained F1-score populations. This has been performed to gain useful insights on the effect of sequentially transferring kernels. Fig. 3.14 shows the boxplots obtained by transferring

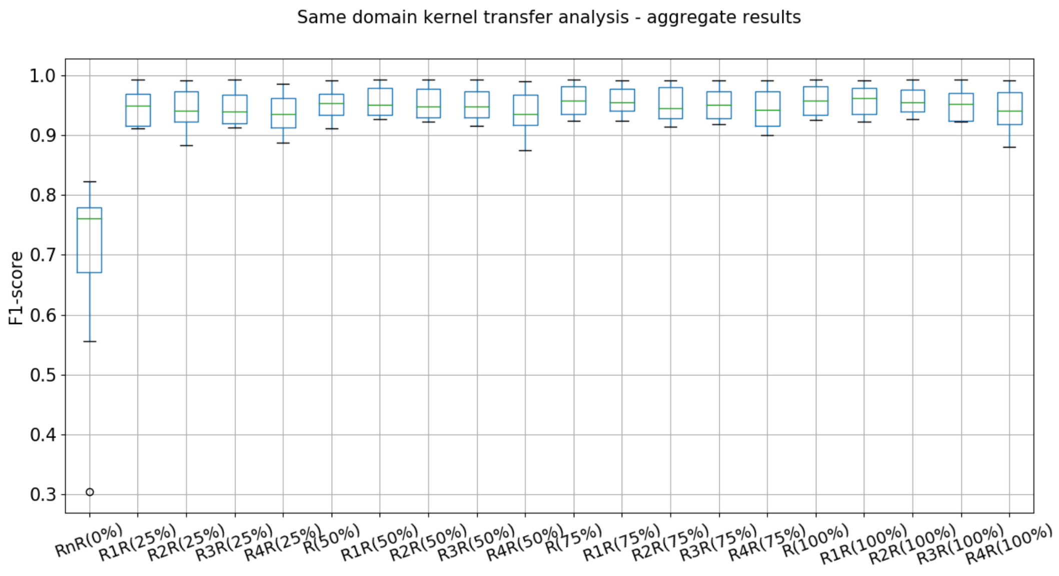


Figure 3.13: F1-score distributions for "same domain" kernel transfer analysis

up to four convolutional kernels and revealing 25% of the *adaptation* subset. The two asterisks (**) placed on top of the line connecting R1R(25%) and R4R(25%) indicate that the *p-value* associated to the Wilcoxon test is lower than 0.01. As a consequence, the two F1-score populations are significantly different. Although not represented in Fig. 3.14, R1R(25%) F1-score population difference from either R2R(25%) or R3R(25%) is not statistically significant.

Different domains

This scenario added an additional layer of complexity with respect to the previous one: the shift between different SHS *source* and RHS *target* domains. The first one has been split in *training* (18 subjects) and *validation* (2 subjects) subsets whereas the latter in *adaptation* (10 patients) and *testing* subsets (2 patients). Similarly to the previous setting, the F1-scores obtained by performing a 10-fold cross-validation across the different kernel transfer configurations are shown in Fig. 3.15. Differently from before, the F1-scores are now far more lower and spread out. Thus, it is now more challenging to carry out and interpret the same statistical analysis as above.

3.5. Domain-adversarial neural networks

For needs of comparison, it was necessary to introduce the performance obtained by training the DANN architecture described in subsection 2.9.2 uniquely on *source* or *target* data

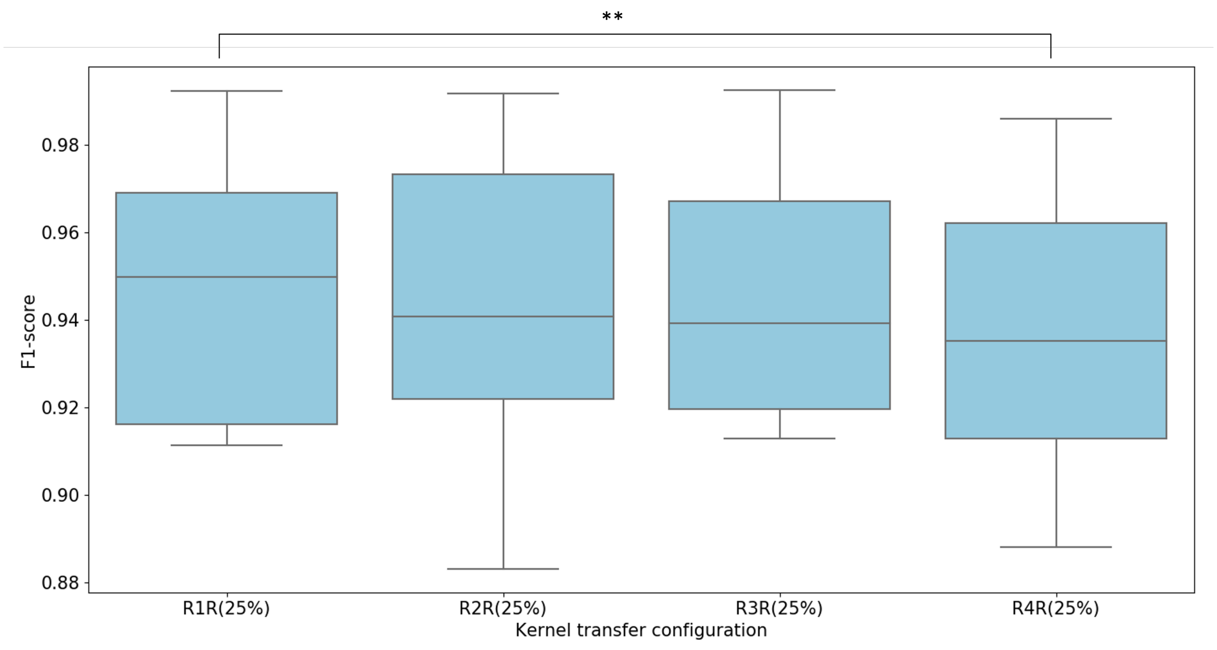


Figure 3.14: F1-score distributions using 25% of the adaptation subset

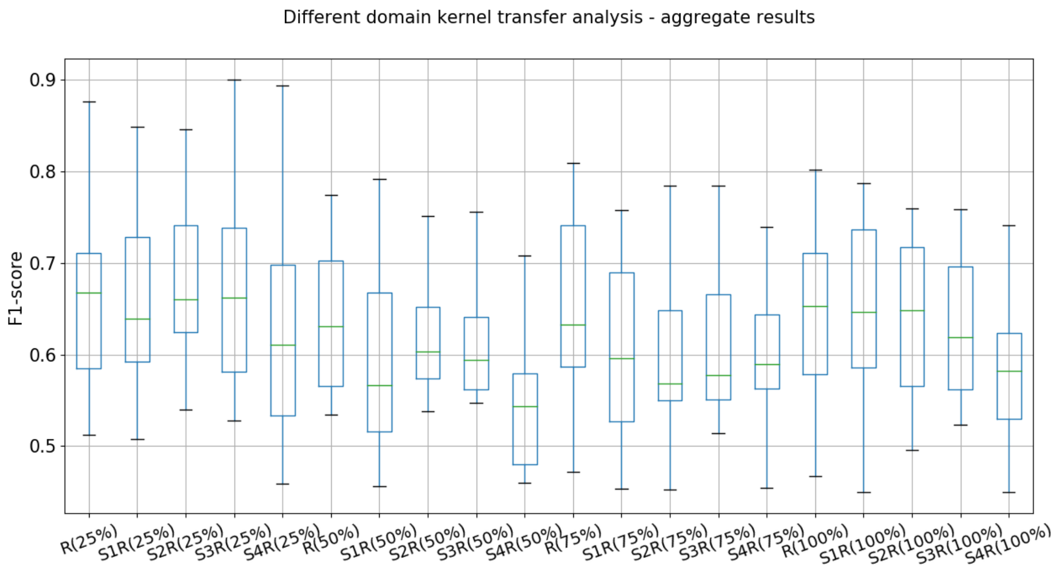
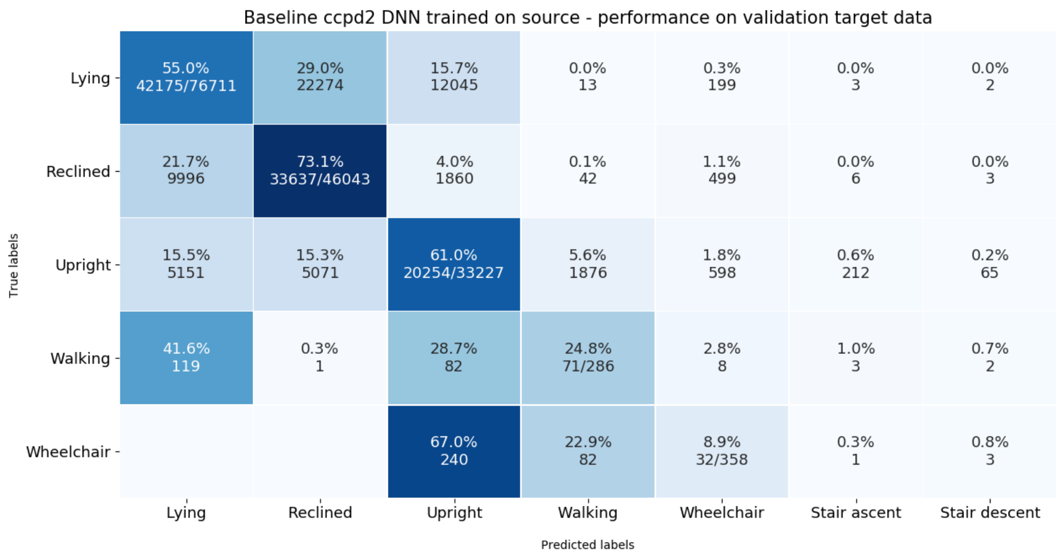
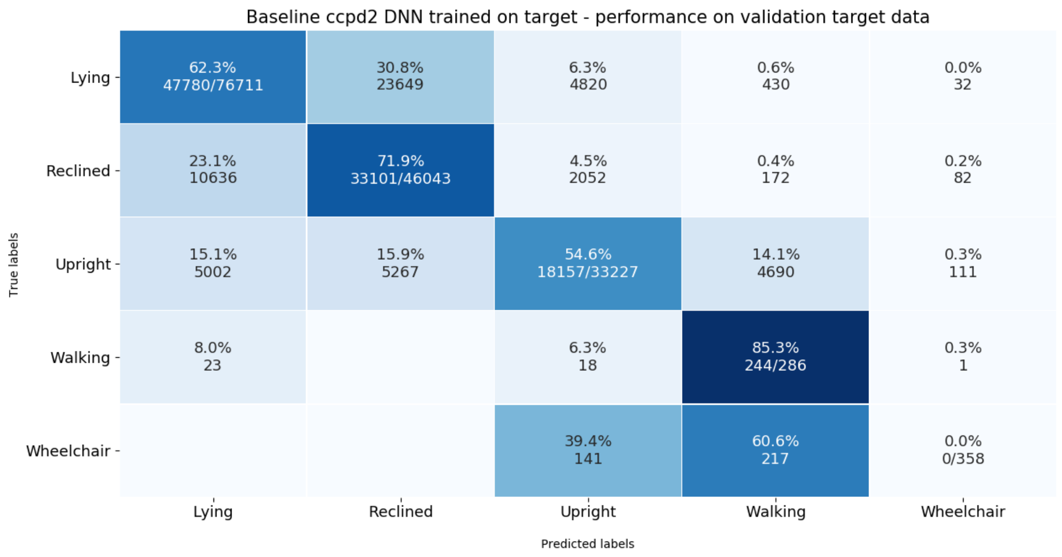


Figure 3.15: F1-score distributions for "different domain" kernel transfer analysis

without adaptation (i.e. $\lambda_d = 0$). The baseline results are represented in Fig. 3.16a and 3.16b, respectively. The WALKING detection accuracy is quite high in Fig. 3.16b due to the *Oversampling* strategy. However this improvement is at the expense of detection accuracy on the other static postures.
The first set of tests were conducted in a *unsupervised learning* setting. This implied that the *label predictor* module (see DANN structure Fig. 2.19) has been trained on SHS *source* activity data only. Within this context, an alternative strategy to the *Oversampling* has been followed to mitigate its side-effects reported above.
To compensate the label distribution within training batches, a portion of target RHS activity data has been fed to the *label predictor* block. This configured a *semi-supervised learning* setting. Throughout this stage, labeled activity data from a specific RHS patient were (partially) revealed during the training procedure. The best result has been obtained by partially including (i.e. 50%) P03 activity data within the training set.
However, "cherry-picking" a specific RHS patient might introduce several biases within the whole procedure. In addition, the a-priori knowledge on *target* data labels distribution might often be limited. Thus, it has been investigated if results comparable to the ones obtained in the previous setting could be produced by including small portions of labeled activity data from multiple random RHS patients to the DANN *training* set. The optimal configuration has been achieved by revealing 10% of labeled activity patterns for each of the 5 randomly-picked RHS patients: P03, P04, P08, P10 and P11. The performance obtained on *target validation* subset with this DANN setting has been represented in Fig. 3.17. In addition, the accuracy and loss trends for both *label predictor* and *domain discriminator* modules have been depicted in Fig. 3.18a and 3.18b, respectively. It can be appreciated that the improvement of both classification loss and accuracy scores for all the involved data subsets (see legend in Fig. 3.18a) was slow but constant. It has to be reminded that the on the aims of DANN is to increase the *domain discriminator* module loss as much as possible. This can be verified in Fig. 3.18b.
The effectiveness of the chosen optimal DANN configuration can be verified under a different point of view. As performed in [16], it might be possible to visually quantify the success of the DANN "feature alignment" task. This can be carried out using "t-Distributed Stochastic Neighbor Embedding" (t-SNE) [44]. In particular, the *label predictor* last hidden layer output obtained by either feeding *source* or *target* samples to the DANN have been embedded into a 2D t-SNE representation. Fig. 3.19a and 3.19b, show a side-to-side comparison of the above-mentioned 2D t-SNE embeddings obtained before and after DANN training. Each activity class has been coded as a digit: [0: *Lying*, 1: *Reclined*, 2: *Stair ascent*, 3: *Stair descent*, 4: *Upright*, 5: *Walking*, 6: *Wheelchair*]. Red and blue numbers refer to *source* (SHS) and *target* (RHS) samples, respectively. The represented



(a) Baseline "source-only" domain-adversarial neural network performance



(b) Baseline "target-only" domain-adversarial neural network performance

Figure 3.16: Performance comparison between two baseline domain-adversarial neural networks

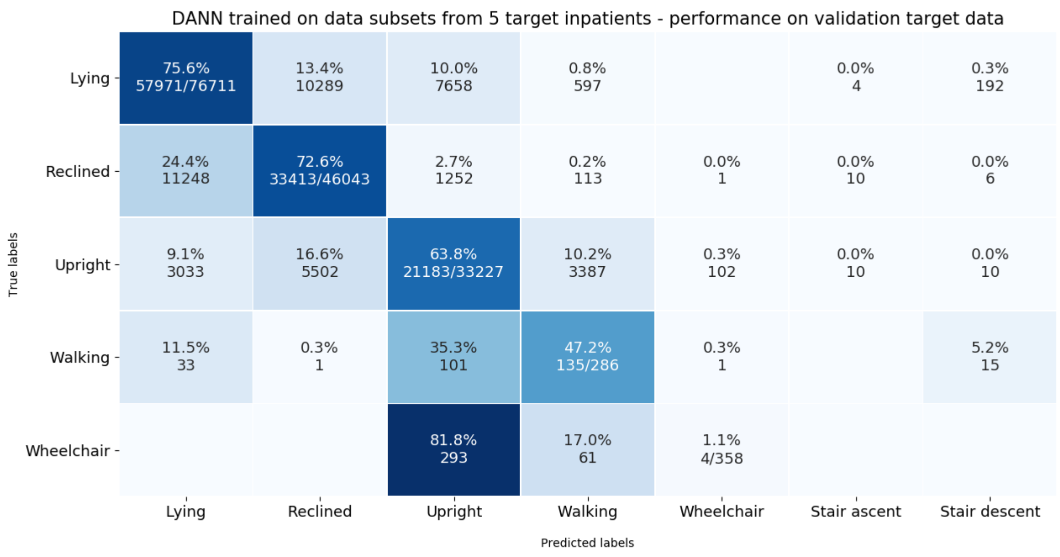
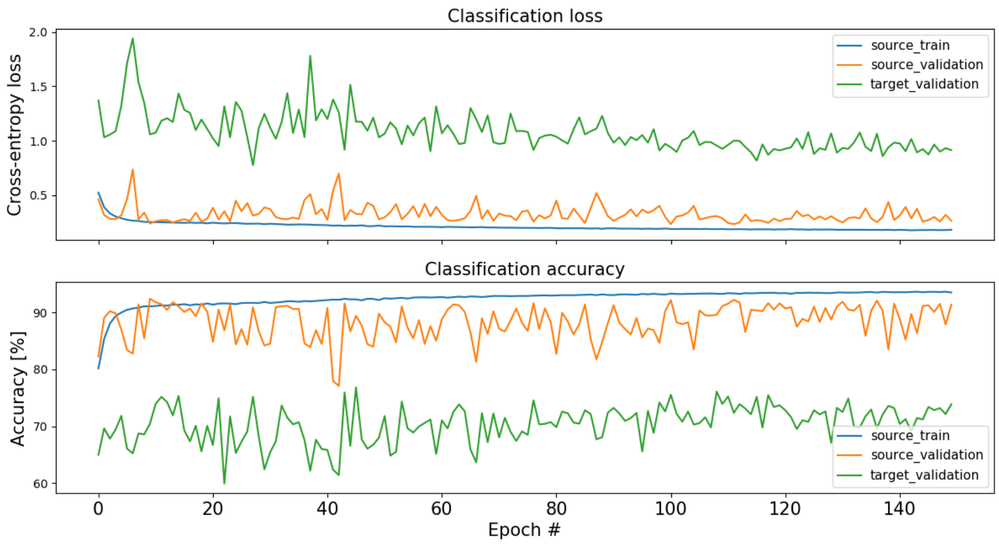
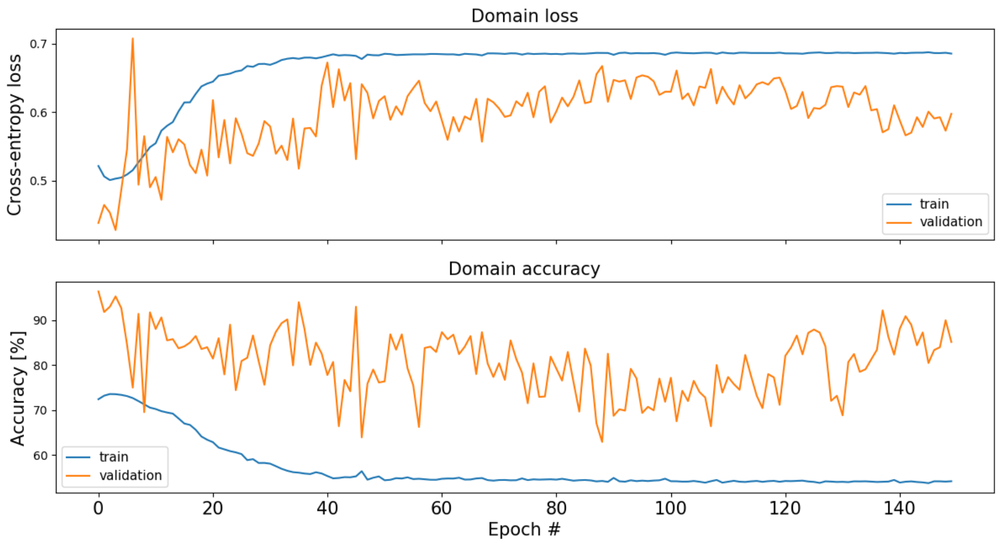


Figure 3.17: Optimal domain-adversarial neural network performance

clusters appeared a bit more compact and intermixed between *source* and *target* samples (especially [0: *Lying*]) after the *domain adaptation* process (see Fig. 3.19b). However, some of those clusters were still isolated (e.g. SHS [0: *Walking*]. On the other hand, moderate-to-low *Walking* speeds should result quite intermixed at the end of the *domain adaptation* process. Fig. 3.20 shows a zoomed-in representation of what represented in Fig 3.19b. The dark-orange rectangle is mainly filled with both red and blue [5: Walking] dots. This indicates that the DANN was able to align the slow-paced *Walking* feature representation for both RHS and SHS.

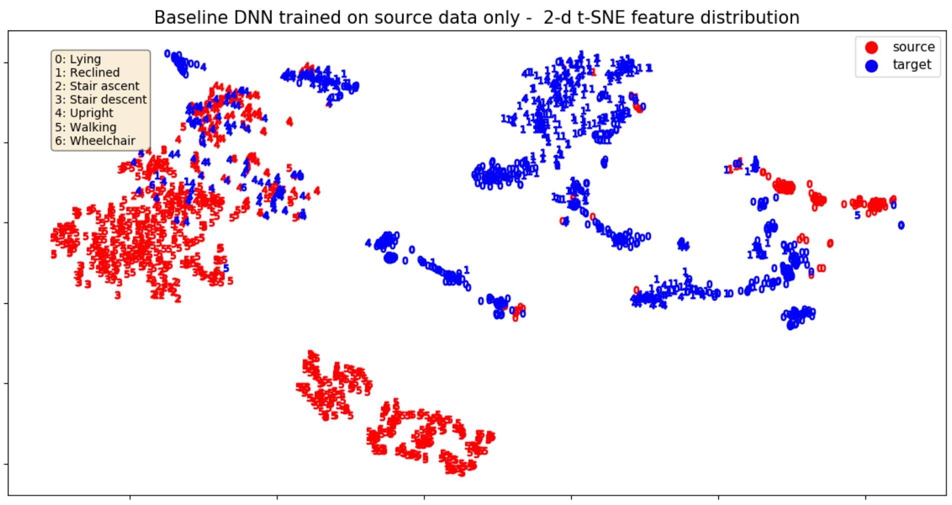


(a) Optimal domain-adversarial neural network label predictor metrics history

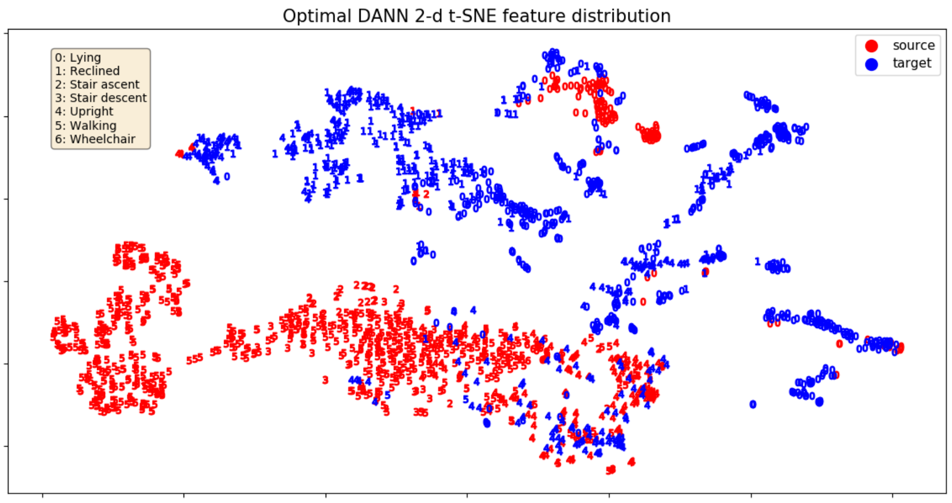


(b) Optimal domain-adversarial neural network domain classifier metrics history

Figure 3.18: Optimal domain-adversarial neural network domain classifier and label predictor metrics history



(a) Baseline "source-only" domain-adversarial neural network feature alignment



(b) Optimal domain-adversarial neural network feature alignment

Figure 3.19: Feature alignment before and after domain adaptation

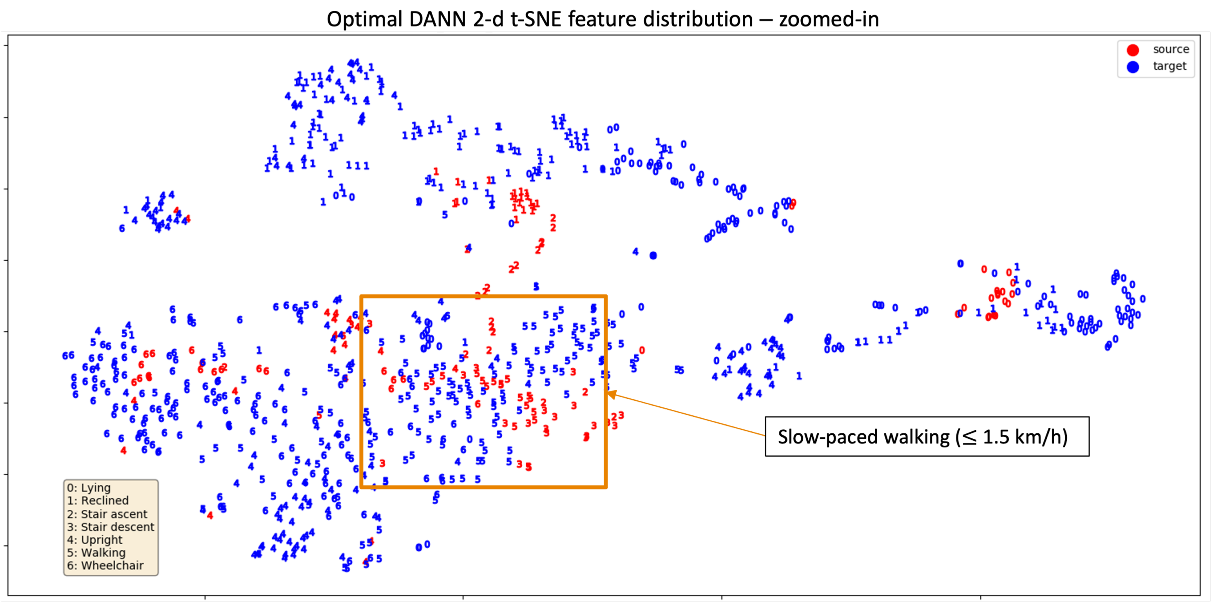


Figure 3.20: Alternative visualization of Fig. 3.19b

4 | Discussion

Data labelling

Although each of the 28 labels had a precise definition (see label set description table 2.2), the annotation procedure resulted challenging due to multiple contaminating factors.

The first consideration concerns static postures (i.e. UPRIGHT, RECLINED, SUPINE)). Eventual bed or (wheel)chair tilt angle has been estimated by inspecting it from camera point of view without any additional support. The worst-case scenario, in terms of annotation effort, consisted in the camera being placed frontally and quite far away from the patient (as depicted in the sample *Barista* snapshot in Fig. 2.5). In addition, some edge-cases constituted an additional layer complexity (i.e. 30°→ SUPINE/RECLINED, 60°→ RECLINED/UPRIGHT). Plus, pillows may have occasionally been placed under patient's back or head for additional comfort. Those objects could interfere with ECGMove4 3D acceleration readings, making the patient result more tilted than how the bed (or chair) actually was. Furthermore, nighttime recordings often showed patients covering with blankets while sleeping. Those segments have been either labeled according to the visible previous posture or discarded in case this clue was unavailable.
Indeed, all the above-mentioned factors could have been mitigated by means of an adequate tool to support the labelling process. The adopted camera was able to acquire point cloud data. Such information could have been used to perform bed tilt angle detection as performed in [29]. Actually, a R&D Philips team with which we had strict cooperation had access to those data. However, their project solely consisted in the detection of bed position for patient monitoring.

Moving to WALKING activity, the most cumbersome information to annotate related to `Step` occurrence. The HAR models adopted in this study aimed at recognizing a general WALKING activity. Thus, a compromise between precision in STEP annotations and time spent on the overall labelling procedure has been made. Anyway, the annotation platform (see annotation setup subsection 2.2.2) did not allow to label timestamp events. Consequently, timestamp events (indicated in the compact label set table 2.10) underwent an additional pre-processing step. Moreover, in case the patient used a walking aid while

Walking, both its type and the starting timestamp for that activity were reported in a separate `.csv` spreadsheet.
In general, Walking activities are one of the most important indicators for patients' discharge readiness and readmission rate [13]. Being able to preemptively predict these two information by analyzing patient's Walking while in hospital represents an important objective. However, the RHS features Walking patterns that might not be suited for this aim. In fact, those are usually short-lasting and slow-paced since patients are mostly moving around the room. Indeed, the sustained Walking could have occurred outside hospital room, but this has not been annotated because outside of camera field of view. In addition, as mentioned in [15], the DNN used in that work struggled in recognizing Walking when using *walking aids*. This constitutes a current and major challenge in the HAR field [10]. However, only 3 out of 12 patients consistently used *walking aids*. This aspect contributed to not perform an in-depth analysis on such activity on RHS. In conclusion, the RHS Walking activity class has not been considered as high-priority when evaluating HAR models. At the same time, the optimal DANN configuration provides a modest recognition performance for such activity.
The additional remark concerned the `Patient care` label indicating physical interaction between the nurse (or clinician) and patient. The guideline was to keep the segments in which `Patient care` followed a standard clinical routine (e.g. bandages removal, auscultation, blood pressure measurement). On the other hand, such samples have been discarded in case of ECGMove4 devices removal (e.g. before the patient took a shower or underwent diagnostic exams).

As a general conclusion, the uncertainty due to the contaminating factors during the annotation process might have played a consistent role in the performance of HAR models developed throughout this work.

Data pre-processing

One of the inconsistencies between SHS and RHS accelerometers consisted in their sampling frequency (i.e. 100 Hz and 64 Hz, respectively). We found that both upsampling (to 100 Hz) and downsampling (up to 16 Hz) input acceleration had little to none effects in terms of classification accuracy. Thus, for the reasons outlined in the baseline DNN subsection 2.7.2, RHS acceleration data have been downsampled to 16 Hz. This decision was supported by the findings of a previous study investigating the optimal acceleration frequencies for different human activity recognition tasks [23]. Furthermore, the majority of RHS activity was composed of *static* postures which, typically, do not feature complex acceleration patterns.

Following the example of [15], the segment length was chosen to make sure that only relevant information was captured in each 6-seconds data window, including slow WALKING and WHEELCHAIR transport.

Data analysis

As already mentioned shown in the label distribution analysis in Fig. 3.1, the RHS activities were mostly composed of *static* postures. This is in agreement with the expected behaviour of general ward patients and has been supported by the `In_bed` contextual label analysis. The latter showed that RHS patients spent ≈80% of the recording session lying in bed.
On the other hand, SHS mainly featured WALKING activities at sustained and different paces (i.e. by setting a treadmill). Thus, this setting was remarkably different from the under-exam RHS, implying a consistent domain shift between the two scenarios.

The scarcity of *posture transitions* obtained after the corresponding analysis showed that the implemented window length was long-lasting enough to avoid "transition-only" segments.

The correlation analysis between 3D acceleration and BMI brought some preliminary speculations confirmed in successive experiments. A remarkable heterogeneity of acceleration baseline values between patients taking the same posture (e.g. `Standing/Held upright` in Fig. 3.2) has been found. This has been identified as a major culprit for the impairment of human activity recognition models based on learning. Although nurses have been trained to correctly attach wearables as close as possible to the intended chest locations, some factors (e.g. bandages, wounds, laparoscopic holes and already-in-place telemetry devices) interfered with this procedure. In addition, BMI additionally contributed to the divergence of 3D acceleration readings from expected values.

To summarize, even limited incorrect sensor placements or orientations could have impaired the activity detection accuracy of models used in this work.

CPA

The determinism that characterized CPA has been leveraged to double-check the correctness of annotations. The whole annotation procedure resulted time-consuming and, in turn, prone to errors. It is of key importance to point out that annotations have not been reworked on the basis of the results obtained with CPA. Instead this algorithm served as a tool to fix major annotation mistakes (e.g. `Lying right` instead of `Lying left`). This process has been beneficial to spot subtle scenarios that were not grasped during video playback when making annotations. For instance, some patients removed one or more movisens® devices by themselves. In such cases, the corresponding annotated acceleration segments have been labeled as `Other`. A similar example consisted in patients undergoing therapies involving devices that interfered with ECGMove4 readings.

The outcome of the CPA *Calibration* module was not successful. In fact, the entire procedure is built around the assumption of having sufficient and uninterrupted *Walking* samples. Both aspects are in contrast with what outlined in the previous Data Analysis discussion. On top of that, some elderly patients consistently used walking aids (e.g. rollator) that made them hunch over while moving around the room. The latter statement contradicts the CPA *Calibration* assumption of being `Upright` while `Walking`. Eventually, if the *Upper* ECGMove4 sensor had been relocated after a successful *Calibration* routine, the latter needed to be performed again. This behaviour has been encountered within RHS (see next paragraph). However, the implemented *Calibration* module is a one-time process not envisaging a sensor repositioning by design.
This issue is closely linked to the consistent heterogeneity in sensors orientations across RHS patients, as previously discussed. This concern could be tackled by adopting a calibration procedure that bases on making the subject repeat a specific posture [22]. However this routine should be a-priori defined, contrasting with the free-living RHS setting. Instead, a potential reworking of the CPA *Calibration* module according to what outlined in the recommendations (see next paragraph) could be envisaged. Hereafter some good practices for using the CPA algorithm in a similar future scenario have been outlined.

Clearly, the *Calibration* process should be used whenever possible. In case of a dataset with scarcity of *Walking* segments, a different trigger event should be devised to start the *Calibration*. For instance, it can be assumed that a subject should take an *Upright* posture a few seconds after a *Sit-to-Stand* transition. The additional benefit of using the proposed *Calibration* trigger is that the orientation matrix (R_{orient}) can be updated whenever a *Sit-to-Stand* transition occurs. In this way, the *Calibration* process would no more be a one-time process but rather become robust to sensor repositioning. However, all the speculations made so far should be supported and verified by an adequate feasibility study.
Following the strategy used to validate CPA in the testing subsection 3.2.1, it is advisable to select a range of candidates α orientation tilt angles in case the *Calibration* procedure is unavailable. According to the aim of the study and label set distribution, the (sub)optimal α and parameters can then be selected.

Deep neural network

DNNs composed of 1D convolutional layers have been developed to usually compare to a baseline feature-based machine learning model [27], [20]. However, shallow ANNs are still capable to obtain fairly good results it the dataset is limited or does not feature complex activity patterns [41]. Indeed, RHS entails a vast dataset suitable to be fed to deep-learning models. At the same time, the featured activities are not complex. In fact, even the rule-based CPA model deprived of the *Calibration* routine was able to achieve an acceptable performance.

The *ccpd2* DNN trained on SHS and validated on RHS provided modest performance on the target activity classes. The misclassifications concerning static postures have been produced by multiple causes: ambiguous annotations, wide range of variations and differences in sensor placement. Such factors were not present in the SHS. Thus, the validation procedure actually consisted in a robustness assessment of the previously-developed model. At the same time, the dynamic activities on which SHS DNN has been prominently trained for did not occur so frequently within RHS. Considering Walking, it is important to note that SHS Walking speeds ranged from 0.4 km/h to 4.0 km/h by pre-setting a treadmill. Indeed, the above-selected walking paces range includes a vast group of ages and conditions among subjects [3]. However, the average Walking RHS speed within the hospital study (although not annotated) might even be slower than the lowest one featured in the SHS study (due to the previously-mentioned reasons).
Examining the *Wheelchair-as-Upright* case, almost the same observations made in the Walking case hold in this context. In fact, it can be expected that general ward patients might use the wheelchair to merely move around, implying a very slow and discontinuous wheelchair propelling.

The next consideration concern the *ccpd2* retraining on RHS. In this case the main issue consisted in the label distributions heterogeneity between train, validation and pre-defined test patients. We noticed that the baseline DNN failed in extracting general features associated to patient activities and postures. This has been confirmed by the L1SO cross-validation results (see table 3.2). The poor F1-score obtained for specific emphasized how the RHS was diverse. Other than the motivations outlined above, patients mannerism for static posture and dynamic activities added up to the overall RHS heterogeneity.

Hybrid ensemble method

Conventional machine learning ensemble methods have been successfully and widely used in several HAR studies [12], [40]. However the proposed hybrid *Stacking* ensemble method introduces the novelty of relying on the outputs of DNN and rule-based model (CPA). The results obtained with this method are promising and could be validated on similar future works.

Referring to the current use-case, the CPA algorithm brought meaningful information for classifying Lying-like postures and overall performance stability. In fact, due to the intrinsic *black-box* nature of the DNN model it was uncertain if it was able to give enough importance to 3D acceleration axes orientation. The CPA algorithm represented a strongly deterministic approach uniquely relying on that information and able to disambiguate some uncertain predictions.

In general, by inspecting the regression coefficient (in Fig. 3.9) we found that the *Stacking* ensemble model preferred the information content contained in the CPA *deltas* rather than DNN probability outputs. Since 3 out of 5 RHS activity classes consists in static postures, it could be expected that the CPA determinism proved highly reliable. As a future work, it might be interesting to explicitly force the *Stacking* ensemble model to use the CPA *deltas* information content in case of uncertainty when classifying specific postures (e.g. Reclined).

The last set of experiments based on including "past" information was not remarkably beneficial for the RHS. Moreover, by also adding the previous probability outputs obtained from the DNN prediction in place of the CPA *deltas* no performance improvements have been encountered. However, it has been found that replicating the same sets of experiments on the dataset acquired from the SHS, the *Meta-Model* trained on DNN probability outputs and CPA deltas outperformed the DNN-only model performance. Plus, by additionally introducing the previous DNN probability outputs to the above-mentioned setting, further overall improvement in the detection of the SHS DNN activity classes has been achieved (especially for *Stair ascent* , *Stair descent* and *Walking*). This might suggested that the rationale of introducing additional past information might be useful to enhance the classification of activities that involve repetitive accelerations patterns. This speculation should verified in future works adopting this technique in a human activity recognition task.

Eventually, this last set of experiments served as a catalyst for some initial testing and future works on hidden markov models (HMM) and Viterbi algorithm (not reported in this document).

Deep transfer learning

Extensive research has been carried out on transferring previously-learned knowledge on a target domain in the context of HAR. The *transfer learning* settings presented in this work are considered standard implementations of such approach.

Using such techniques usually implies a degradation of the detection accuracy on activities classes related to the target domain. However, such impairment is often limited making *transfer learning* suitable to be adopted in several scenarios. Nevertheless, the results obtained using this approach to transfer activity feature representation from SHS to RHS were unsatisfactory. By observing Fig. 3.11a and 3.11b, it is quite clear that the represented *transfer learning* settings failed in adapting the features extracted from SHS to the hospital activity data. The enhancement obtained after *fine-tuning* was almost irrelevant.

A possible explanation for this behaviour, might be attributed to consistent domain shift between SHS and RHS. To be fair, many of the activity classes are shared between the two. However, some of them (e.g. Walking and Wheelchair) are inherently different for the reasons outlined above. In addition, the large heterogeneity of static postures activity samples within RHS and compared to SHS might have severely impaired the learning and *fine-tuning* processes.

Kernel transfer analysis

The results obtained at the end of the *kernel transfer analysis* procedure have been compared to what reported in [31].

We found that by revealing a small percentage (as low as 25%) of labeled activity data performed by the left-out patient (i.e. "same domain" scenario in subsection 3.4.2) the F1-score obtained for the respective *test* subset sharply increased. Indeed, recalling that most of the RHS activities consisted of static postures, it should not be challenging for the DNN to classify them after receiving some "clues" on activity data carried out by test patient itself.

By carrying out the latter analysis on all boxplots obtained in Fig. 3.13, two general behaviours have been identified.
The first relates to an overall decrease in F1-score values when transferring more kernels, independently from the used percentage of *adaptation* subset. This might be related to more user-related features captured by deeper kernels. At the same time, it is important to note that the last convolutional layer alone contains over 50% of the overall *ccpd2*

parameters. As a consequence, the general performance decrease might also be due to a reduced optimization space. Except for the last consideration, this finding is in line with what outlined in [31].
The second trend directly relates to the analysis of the *p-values* obtained in conjunction with the Wilcoxon test. In fact, it has been noticed that the F1-score populations obtained by transferring one or two kernel are not significantly different, independently from the revealed percentage of *adaptation* subset. This meant that features extracted up to the second convolutional layers were user-independent. Also this results matched with what reported in the reference study
A final consideration can be made on the RHS "same domain" *kernel transfer analysis*. By observing the high F1-score values obtained by accustoming the DNN to a small percentage of activities samples of the test set (see Fig. 3.13), the annotation and acquisition process entailed in a similar future use-case can be drastically unburdened by adopting this strategy. On a practical side, the already trained DNN model might only adapt to few hours of activity data carried out by the test patient to provide fairly good results.

The "different domain" *kernel transfer analysis* scenario was quite more complex to interpret. Indeed, transferring knowledge learned from a protocol laboratory activity data environment (SHS) to a free-living one (RHS) represents a challenging task. However, although not as linear as in the previous scenario, also in this case the F1-score values generally decreased by transferring more kernels trained on SHS. On top of that, by increasing the used percentage of the *adaptation* subset, the boxplots distributions spreading fairly decreased. Also this result is in line with what expected since the DNN has the chance more information on the target RHS setting.

Differently from the statistical analysis performed in the previous case, now was way more difficult to assess if two F1-score populations (see boxplots in Fig. 3.15) were significantly different due to the their wide-spread distributions. As a general finding, the F1-score populations associated to transferring 3 or 4 convolutional layers were almost always significantly different under a statistical point of view. In this case, however, due to the uncertainty linked to the statistical analysis, it was trickier to determine which kernels contain feature representations associated to a specific domain and which are user-independent. Thus, no direct comparison could be performed with the corresponding scenario described in [31].

Domain-adversarial neural networks

One of the main issues encountered during the validation of DNN models on the hospital study setting was the consistent data heterogeneity due to sensor placements (i.e.

orientations) and diverse patient-specific activity patterns. Hence, it was indeed valuable to obtain activity feature representations agnostic to the above-mentioned factors. Techniques based on *adversarial learning* might constitute a valid solution to address this issue. Indeed, latest advancements of this approach entail quite sophisticate DNN architectures [39]. However, few studies directly implemented DANNs in the context of HAR. This project aims at investigating this knowledge gap by validating DANNs on RHS.

The first insight obtained after the *domain adaptation* procedure from *source* SHS to *target* RHS concerned the label discrepancies between the two settings. In fact, we found that the *domain discriminator* module fairly easily identified the membership of the *Stairs Walking* activity to the SHS. This has been verified by observing that the DANN almost never predicted those classes when validated on RHS. This concept generally refers to *open set recognition.* The DANN implemented in this work was not purposely designed for this task, but it can be accustomed to it in future related works.
Similarly, the "target-only" DANN (see Fig. 3.16b) *domain classifier* had no difficulties in discriminating which extracted features belonged to the *source* or *target* domain (i.e. WALKING and static postures, respectively). This has been confirmed by noticing a very low *domain classifier* binary cross-entropy loss occurring during the initial training epochs.

The first set of DANN testings were conducted in a fully *unsupervised learning setting* and implementing an *Oversampler* to generate balanced batches.
However, such learning setting implies that no a-priori knowledge on *target* labels is available. Thus, according to this reason and the modest results obtained for the *unsupervised*, it has been decided to include RHS labelled samples to the training batches. In this way, the differences in SHS labels distribution were compensated by RHS ones.

The best result for the "single-patient" *semi-supervised setting* were obtained by revealing 50% of P03 labelled activity data. In fact, by analyzing P03 labels distribution it has been verified that it almost perfectly matched with the overall RHS HAR labels distribution (see Fig- 3.1). In other words, P03 consisted in the "most-representative" patient for the *target* hospital dataset. The performance obtained for this DANN *semi-supervised learning* setting outperformed the results obtained for the fully *unsupervised* one.

The successful results obtained for the optimal "multi-subject" DANN configuration suggested that such model could be able to rapidly adapt to future activity recognition tasks by annotating activity patterns in a spot-check fashion (e.g. by a trained nurse) and revealing them during the DANN training process. Such remark was corroborated with what found at the end of the "same-domain" *kernel transfer analysis* (see end of subsection 3.4.2).

Concerning feature alignment, it was expected that some fast-paced ($\geq$ 4 km/h) SHS *Walking* activities were too much different from RHS WALKING patterns. DANN did not manage to align the related features that appeared as an isolated [5: Walking] (in red) cloud (see bottom-left of Fig. 3.19b).

Research question and optimal model

Upon the considerations made so far, a comprehensive answer to the research question outlined in the section 1.2 can finally be outlined-

Although the SHS and RHS may share similarities in label set and sensor modalities and placement, several above-mentioned causes made the domain shift quite remarkable. Thus, the knowledge previously acquired from SHS data can be **limitedly** directly transferred to the hospital study setting.
In fact, it was necessary to *fine-tune* the SHS feature representations to the target activity data carried out by inpatients. This task was not accomplished successfully by using standard *transfer learning* techniques. Instead a more refined *domain adaptation* approach based on DANNs was required to gain substantial improvement in activity recognition.

The optimal model to perform this task was identified in DANN leveraging knowledge on a partial amount of RHS labelled data from multiple patients. Table 4.1 summarizes the F1-scores obtained on RHS data for each of the optimal configurations found for each investigated approach. Although using the DANN (50% P03) model provided the best F1-score, the "multi-subject" DANN (score in bold in table 4.1) was identified as optimal. The rationale behind this is that the cross-entropy loss scores obtained with this setting was the lowest among all models tested out. This suggested that the "multi-subject" DANN might be more robust to target domain variation for potential future uses. In addition, it is worth to recall that DANN models might be in general preferred because they drastically reduce the need of labeled target RHS data.

Model	**(Weighted) F1-score**
DANN (50% P03)	0.76
DANN (multi-subject)	**0.73**
DANN (unsupervised)	0.69
DNN validation	0.69
CPA (optimal *alpha*)	0.65
DTL (unfreeze all)	0.61

Table 4.1: Summary of main models performance

5 | Conclusion

In this project we tackled a acceleration-based HAR task entailing the evaluation a DNN developed on simulated hospital setting (SHS) on a real one (RHS). Due to the inherent discrepancies between SHS and RHS, a slight variation of the original DNN architecture has been designed (defined as *ccpd2*). During the validation procedure, we proved that the features extracted from the SHS domain were not representative enough for the RHS. In addition, the RHS dataset itself, although mainly composed of trivial static postures, resulted be quite diverse. This facet consistently impaired the retraining of the *ccpd2* model on RHS activity data.

Furthermore, several alternative approaches and techniques have been tested among which the rule-based CPA algorithm and a novel hybrid *Stacking* ensemble machine learning model. However, the former suffered from the lack of a sensor calibration procedure. The latter consistently improved the detection of specific postures but at the expense of others.

Due to the consistent domain shift between SHS and RHS, *transfer learning* techniques have been adopted to *fine-tune* the SHS feature representations to the target hospital activity data. However, no substantial improvements were obtained by using this approach.

Thus, we implemented a more refined *domain adaptation* technique based on DANNs. After extensive testing, we found that the optimal DANN configuration entailed a *semi-supervised learning* setting by revealing small portions of labeled activity data from multiple RHS patients. Such optimal model offered a 0.73 weighted F1-score. We considered this model as the best compromise between annotation burden and performance scores.

As future works, it might be valuable to investigate on a different CPA calibration routine as suggested in this work. In addition, it might be interesting to accustom the developed models to include the information acquired from both ECGMove4 sensors (i.e. sensor fusion). Finally, the optimal identified DANN configuration should be validated to assess if the percentage of target labeled data should be varied according to the HAR task.

Bibliography

[1] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(12):31314–31338, 2015.

[2] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

[3] R. W. Bohannon. Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants. *Age and ageing*, 26(1):15–19, 1997.

[4] N. A. Capela, E. D. Lemaire, and N. Baddour. Feature selection for wearable smartphone-based human activity recognition with able bodied, elderly, and stroke patients. *PloS one*, 10(4):e0124414, 2015.

[5] P. J. Carter and S. Lewsen. *Lippincott's textbook for nursing assistants: a humanistic approach to caregiving*. Lippincott Williams & Wilkins, 2005.

[6] P. Casale, O. Pujol, and P. Radeva. Human activity recognition from accelerometer data using a wearable device. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 289–296. Springer, 2011.

[7] S. Chernbumroong, S. Cang, A. Atkins, and H. Yu. Elderly activities recognition and classification for applications in assisted living. *Expert Systems with Applications*, 40 (5):1662–1674, 2013.

[8] I. Cleland, B. Kikhia, C. Nugent, A. Boytsov, J. Hallberg, K. Synnes, S. McClean, and D. Finlay. Optimal placement of accelerometers for the detection of everyday activities. *Sensors*, 13(7):9183–9200, 2013.

[9] D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36(3):537–556, 2013.

[10] R. De Ridder and C. De Blaiser. Activity trackers are not valid for step count registration when walking with crutches. *Gait & posture*, 70:30–32, 2019.

[11] R. Ding, X. Li, L. Nie, J. Li, X. Si, D. Chu, G. Liu, and D. Zhan. Empirical study and improvement on deep transfer learning for human activity recognition. *Sensors*, 19(1):57, 2019.

[12] Z. Feng, L. Mo, and M. Li. A random forest-based ensemble method for activity recognition. In *2015 37th annual international conference of the ieee engineering in medicine and biology society (embc)*, pages 5074–5077. IEEE, 2015.

[13] S. R. Fisher, Y.-F. Kuo, G. Sharma, M. A. Raji, A. Kumar, J. S. Goodwin, G. V. Ostir, and K. J. Ottenbacher. Mobility after hospital discharge as a marker for 30-day readmission. *Journals of Gerontology Series A: Biomedical Sciences and Medical Sciences*, 68(7):805–810, 2013.

[14] A. Fleury, M. Vacher, and N. Noury. Svm-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. *IEEE transactions on information technology in biomedicine*, 14(2):274–283, 2009.

[15] E. Fridriksdottir and A. G. Bonomi. Accelerometer-based human activity recognition for patient monitoring using a deep neural network. *Sensors*, 20(22):6424, 2020.

[16] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[17] L. Gao, A. Bourke, and J. Nelson. Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical engineering & physics*, 36 (6):779–785, 2014.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[19] I. C. Gyllensten and A. G. Bonomi. Identifying types of physical activity with a single accelerometer: evaluating laboratory-trained algorithms in daily life. *IEEE transactions on biomedical engineering*, 58(9):2656–2663, 2011.

[20] A. Ignatov. Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915–922, 2018.

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[22] M. Jiang, H. Shang, Z. Wang, H. Li, and Y. Wang. A method to deal with installa-

tion errors of wearable accelerometers for human activity recognition. *Physiological measurement*, 32(3):347, 2011.

[23] A. Khan, N. Hammerla, S. Mellor, and T. Plötz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73: 33–40, 2016.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[27] S.-M. Lee, S. M. Yoon, and H. Cho. Human activity recognition from accelerometer data using convolutional neural network. In *2017 ieee international conference on big data and smart computing (bigcomp)*, pages 131–134. IEEE, 2017.

[28] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[29] L. Liu and S. Mehrotra. Bed angle detection in hospital room using microsoft kinect v2. In *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 277–280. IEEE, 2016.

[30] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[31] F. J. O. Morales and D. Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 92–99, 2016.

[32] B. Najafi, K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. J. Bula, and P. Robert. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *IEEE Transactions on biomedical Engineering*, 50(6):711–723, 2003.

[33] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[34] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile

phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):1–27, 2010.

[35] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[37] J. Shafi, A. Waheed, and P. V. Krishna. Analysing human activity patterns by chest-mounted wearable devices. In *Emerging Research in Data Engineering Systems and Computer Communications*, pages 389–401. Springer, 2020.

[38] E. Soleimani and E. Nazerfard. Cross-subject transfer learning in human activity recognition systems using generative adversarial networks. *Neurocomputing*, 2020.

[39] E. Soleimani and E. Nazerfard. Cross-subject transfer learning in human activity recognition systems using generative adversarial networks. *Neurocomputing*, 426: 26–34, 2021.

[40] A. Subasi, D. H. Dammas, R. D. Alghamdi, R. A. Makawi, E. A. Albiety, T. Brahimi, and A. Sarirete. Sensor based human activity recognition using adaboost ensemble classifier. *procedia computer science*, 140:104–111, 2018.

[41] J. Suto and S. Oniga. Efficiency investigation from shallow to deep neural network techniques in human activity recognition. *Cognitive Systems Research*, 54:37–49, 2019.

[42] C. E. Tudor-Locke and A. M. Myers. Challenges and opportunities for measuring physical activity in sedentary adults. *Sports medicine*, 31(2):91–100, 2001.

[43] P. Urwyler, L. Rampa, R. Stucki, M. Büchler, R. Müri, U. P. Mosimann, and T. Nef. Recognition of activities of daily living in healthy subjects using two ad-hoc classifiers. *Biomedical engineering online*, 14(1):54, 2015.

[44] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[45] J. Wang, V. W. Zheng, Y. Chen, and M. Huang. Deep transfer learning for cross-domain activity recognition. In *proceedings of the 3rd International Conference on Crowd Science and Engineering*, pages 1–8, 2018.

[46] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[47] M. Zubair, K. Song, and C. Yoon. Human activity recognition using wearable accelerometer sensors. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–5. IEEE, 2016.

List of Figures

List of Tables

List of Acronyms

Acronym	**Full name**
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Convolutional Neural Network
GAP	Global Average Pooling
HAR	Human Activity Recognition
MLP	Multilayer Perceptron
LSTM	Long Short-Term Memory
SHS	Simulated Hospital Study
RHS	Real Hospital Study

Acknowledgements

This thesis project has been carried out during a tough historical moment. More than once I experienced on my skin how the lack of community life could represent an insidious danger for mental well-being. For this reason, it is more important than ever to express gratitude to people that were there for me in spite of everything.

First of all, I would like to thank all the Philips colleague with which I had the opportunity to share a wonderful working experience: Guido, Gabriele, Alberto, Carlijn, Jens, Aline and René. Last but not least, I owe my deepest thanks to Lieke. She was always able to make brilliant observations on my work and guide me towards the best solutions. Still, I always felt on the "driving seat" of my internship project. I think that, thanks to her supervision, I really matured my awareness and sense of initiative not only as a researcher but also as a person.

In addition, I am grateful to my advisor Prof. Caiani who made me aware of the internship opportunity there in Philips. His feedbacks and corrections during the revisions of this thesis were always swift.

Although in a short time, I had the opportunity to establish solid relationships with my flatmates in Eindhoven. Thus, I would like to say thank you to: Derya, Salomé and Stijn. A special mention goes to my "brother from another mother" Niklas. Really looking forward to meet you again in the future and wishing you the best of luck for your career.

Naturally, I cannot fail to express my gratitude towards to my University colleagues: Ilaria, Andrea, Ludovica and many others. Also in this case, I have to make a special mention: my lifelong friend Veronica. Although we lost touch a bit, I will not ever forget the piece of life I had to luck to share with you.

Eventually, I would like to thank my family for being always supportive and caring to me. Cheers also to my "John" second family for making me welcome whenever I came back home.

Thank you too, Giulia. We went through a lot of stuff but we were always able to make our love prevail over everything.