Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Eskil Reitan Riibe

Self-Supervised Human Activity Recognition on Free-Living Sensor Data Streams

Master's thesis in Engineering and ICT
Supervisor: Kerstin Bach
Co-supervisor: Aleksej Logacjov
June 2023

NTNU
Norwegian University of
Science and Technology

Eskil Reitan Riibe

Self-Supervised Human Activity Recognition on Free-Living Sensor Data Streams

Master's thesis in Engineering and ICT
Supervisor: Kerstin Bach
Co-supervisor: Aleksej Logacjov
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Quantifying the state of physical activity among the population is traditionally done through questionnaires, a method subject to inherent biases. As an alternative, Human Activity Recognition (HAR) from sensor data streams can potentially offer a more accurate approach to assessing physical activity. This thesis aims to develop a system for accurately recognizing activities under free-living conditions - a notably more complex task than recognizing activities recorded in-lab due to the diversity and unpredictability of real-world environments. One significant challenge with free-living data is the costly process of collecting labeled data. This leads us to explore whether self-supervised learning methods (SSL) can be utilized for free-living HAR as a way to reduce the reliance on these labels by creating auxiliary tasks from raw unlabeled data. This is a methodology that has achieved significant success in various other domains like vision and speech recognition. For this purpose, we mainly utilize the unlabeled free-living HUNT4 dataset for self-supervision, which consist of accelerometer recordings from around 35 000 participants that were recorded for roughly a week. For supervised fine-tuning, the free-living and labeled HARTH dataset is used, which consists of accelerometer recordings from 22 participants recorded for around 1-2 hours.

We examine and implement the two prominent SSL methods SimCLR and SimSiam that represent two domains of self-supervised learning, namely contrastive and non-contrastive learning. We find that the pretraining stage successfully identifies features valuable for the classification of certain activities. However, despite the apparent promise of these methods, purely supervised learning still outperform these methods, with an improvement of as much as 21 % from the SimCLR. Our findings indicate that certain everyday activities present a much greater challenge in classification than others. Moreover, SimCLR consistently exceeds SimSiam's performance in all experiments, emphasizing the importance of contrasting negatives in SSL for free-living HAR. We also address the issue of sampling bias in contrastive self-supervised learning by implementing false negative detection using clustering. However, our findings suggest that this approach does not enhance the training process compared to doing random sampling. Overall, while SSL demonstrates potential in HAR, our results show there still is considerable room for enhancement and refinement of its application in free-living environments. We also find that there is only a minor drop in performance of both SimCLR and SimSiam when pre-training the model on the much smaller UCI-HAR dataset which consists of a different sensor setup and recording environment than HARTH. Lastly, we find that the supervised version outperforms the best self-supervised method, SimCLR, already at below 1 % of available labeled data.

Sammendrag

Å kvantifisere befolkningens fysiske aktivitetstilstand gjøres tradisjonelt gjennom spørreskjemaer, en metode som er underlagt iboende skjevheter. Som et alternativ kan menneskelig aktivitetsgjenkjenning (HAR) gjennom datastrømmer fra sensorer potensielt tilby en mer nøyaktig tilnærming til vurdering av fysisk aktivitet. Denne oppgaven tar sikte på å utvikle et system for nøyaktig gjenkjenning av aktiviteter under frie leveforhold - en merkbar mer kompleks oppgave enn å gjenkjenne aktiviteter som har blitt utført i lab-omgivelser på grunn av mangfoldet og uforutsigbarheten under frie leveforhold. En betydelig utfordring med frie leveforhold i HAR er den kostbare prosessen med innsamling av merkede (labeled) data. Dette fører oss til å utforske om selv-veiledede læringsmetoder (SSL) kan brukes til HAR under frie leveforhold som en måte å redusere avhengigheten av disse merkene (labels) ved å lage hjelpeoppgaver fra rå umerkede data. Dette er en metodikk som har oppnådd betydelig suksess i forskjellige andre domener som datasyn og talegjenkjenning. For dette formålet bruker vi hovedsakelig det umerkede frittlivs HUNT4-datasettet for selv-veiledning, som består av akselerometeropptak fra rundt 35 000 deltakere som ble registrert i omtrent en uke. For veiledet finjustering brukes det fri-livs og merkede HARTH-datasettet, som består av akselerometeropptak fra 22 deltakere registrert i omtrent 1-2 timer.

Vi undersøker og implementerer de to fremtredende SSL-metodene SimCLR og SimSiam som representerer to domener av selv-veiledet læring, nemlig kontrastiv og ikke-kontrastiv læring. Vi kommer frem til at forhåndstreningstrinnet (pre-training stage) identifiserer egenskaper som er verdifulle for klassifiseringen av visse aktiviteter. Til tross for dette, overgår ren veiledet læring fremdeles disse metodene, med en forbedring på så mye som 21 % fra SimCLR. Funnene våre indikerer at visse hverdagsaktiviteter utgjør en mye større utfordring i klassifiseringen enn andre. Videre overstiger SimCLR konsekvent SimSiams prestasjoner i alle eksperimenter, noe som understreker viktigheten av å kontrastere negativer i SSL for fri-livs HAR. Vi adresserer også problemet med utvalgsskjevhet i kontrastiv selv-veiledet læring ved å implementere falsk negativ deteksjon ved hjelp av clustering. Imidlertid tyder funnene våre på at denne tilnærmingen ikke forbedrer treningsprosessen sammenlignet med å gjøre tilfeldig utvalg. Samlet sett, selv om SSL viser potensiale i HAR, viser resultatene våre at det fremdeles er betydelig rom for forbedring av anvendelsen i fri-livsmiljøer. Vi finner også at det bare er en mindre nedgang i ytelsen til både SimCLR og SimSiam når vi forhåndstrener modellen på det mye mindre UCI-HAR-datasettet som består av en annen type sensorsoppsett og registreringsmiljø enn HARTH. Til slutt finner vi at den overvåkede versjonen overgår den beste selv-veilede metoden, SimCLR, allerede ved under 1 % av tilgjengelige merkede data.

Preface

This master's thesis marks the culmination of my five-year journey toward a Master of Science (MSc) degree at the Norwegian University of Science and Technology (NTNU), which concludes in June 2023. I want to thank my supervisor Kerstin Bach for guiding me in the right direction throughout the year and for being highly available whenever I had any questions regarding the HUNT4 study, previous HAR studies at NTNU, or other important technical matters. I also want to thank Kerstin for letting me write this thesis for the Norwegian Research Center for AI Innovation (NorwAI) under her supervision. I very much want to thank my co-supervisor, Aleksej Logacjov, for his expertise in self-supervised learning, for the introduction to the HAR framework used for HAR studies at NTNU, and for his availability whenever I had any technical questions. Lastly, I want to thank NorwAI for letting me present my work at their seminar to other AI students and employees.

Eskil Reitan Riibe
Trondheim, June 11, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human activity recognition (HAR) from sensor data streams has gained popularity in recent years due to its potential in domains like healthcare, sports, and entertainment. One critical application of HAR is monitoring and promoting physical activity, which is important in maintaining good health and preventing chronic diseases. Inactivity has been linked to various health risks, which include obesity, cardiovascular diseases, and diabetes [Warburton et al., 2006] emphasizing the importance of both understanding and promoting physical activity within the population. Traditionally, questionnaires have been the main source of information to map out the level of physical activity among individuals. However, these self-reporting methods suffer from several drawbacks. This includes subjective interpretation, recall bias, and limited accuracy in capturing the true extent of physical activities [Prince et al., 2008]. For this reason, there is a growing need for more objective and reliable approaches to analyzing human activities.

The HUNT4 study is a large-scale population-based health study that aims to investigate various aspects of health, including physical activity [Åsvold et al., 2023]. By leveraging modern technology and data analysis, the HUNT4 study seeks to understand physical activity patterns and their implications on public health. This provides a strong motivation for the exploration of novel techniques in HAR. This is particularly relevant when leveraging sensor data streams from individuals in free-living scenarios, as they enable continuous monitoring of activities and provide objective sensor data on how these activities are performed in real-life settings.

Artificial intelligence has come a long way since the term was introduced in 1956 at Dartmouth college and has in recent years shown tremendous potential in a wide range of applications, including healthcare, finance, and transportation [Russell, 2010]. In the context of HAR, machine learning and neural networks have emerged as powerful tools for modeling complex patterns in sensor data and inferring underlying activities. These techniques have already demonstrated success in various supervised learning settings [Logacjov et al., 2021; Wang et al., 2019] where sufficient labeled data is available to train the models. However, collecting labeled data for free-living HAR datasets can be expensive, time-consuming, and labor-intensive, as it often requires manual annotation of activities by human experts [Bulling et al., 2014]. This poses a significant challenge in scaling HAR models to real-world scenarios where labeled data may be limited or unavailable.

Self-supervised learning, an emerging paradigm in machine learning, offers a promising solution to this challenge in various fields like computer vision [Chen et al., 2020; He et al., 2020]

and NLP [Devlin et al., 2018]. By exploiting the inherent structure and relationships within the data, self-supervised learning enables models to learn useful representations without relying on explicit labels. This approach can potentially alleviate the need for costly data annotation and facilitate the development of more robust and scalable HAR models for free-living sensor streams.

This thesis aims to investigate the potential advantages of self-supervised learning for human activity recognition in free-living sensor streams. We will focus on contrastive- and non-contrastive self-supervised learning techniques and their applicability to HAR tasks, with the goal of developing more effective and efficient models for real-world applications. By bridging the gap between AI research and public health initiatives such as the HUNT4 study, we hope to contribute to a better understanding of physical activity patterns.

1.1 Goals and Research Questions

Goal To evaluate the effectiveness and performance of self-supervised learning methods on free-living sensor data streams for Human Activity Recognition (HAR).

In this research, the primary objective is to assess the potential and performance of self-supervised learning techniques for Human Activity Recognition tasks specifically with free-living sensor data streams. This represents a complex multi-class classification challenge occurring in dynamic, real-world environments. The results of this investigation will shed light on the strengths and weaknesses of self-supervised learning techniques in this domain, contributing to the broader goal of building more robust, effective models for real-life HAR tasks.

Research question 1 *What are the latest advances and key characteristics of state-of-the-art supervised- and self-supervised HAR systems?*

This question serves to provide a comprehensive overview of the current state of supervised and self-supervised models used in HAR systems. The intention is to identify the most effective strategies and methods in practice today and compare them based on their ability to classify a variety of different activities and robustness to data variations. The study will inform the choice of self-supervised methods to be used and evaluated in this study.

Research question 2 *Can existing contrastive and non-contrastive self-supervised methods like SimCLR and SimSiam be exploited or further developed for human activity recognition on free-living data like HUNT4 and HARTH?*

Building upon the insights gained from the first research question, this question explores the applicability of specific contrastive and non-contrastive self-supervised techniques, namely SimCLR and SimSiam, to HAR tasks using free-living data from the HUNT4 and HARTH datasets. Additionally, we want to explore whether the SimCLR can be modified with false negative detection through clustering and what impact this will have on the results.

Subquestion 2.1 *What is the comparative performance of the selected self-supervised methods, and how well do they perform against purely supervised models?*

Subquestion 2.2 *How robust are the selected self-supervised methods in scenarios with limited labeled data, and to what extent do they demonstrate successful transfer learning between different types of HAR contexts?*

Subquestion 2.3 *How well do the experimental results align with the findings reported in the literature study?*

The first subquestion directly compares the performance of the selected self-supervised learning methods. The comparison is based on standard metrics such as precision, recall, and f1-score, providing quantitative insights into their respective strengths and weaknesses in the context of HAR with free-living sensor data. The second subquestion investigates their performance and adaptability in less-than-ideal situations, such as when there is a limited amount of labeled data available and where the environments of the trained model differ from the environment of the HAR problem at hand. Such variations can include types of sensors, sensor positions, and types of subjects. Robustness to variation is a critical feature for models deployed in diverse real-world scenarios. The last question aims to compare the experimental results of this thesis with findings from previous research where self-supervised learning mostly have been applied to data collected in laboratory conditions.

1.2 Research Method

The first stage of the research method involves a comprehensive literature study. The aim is to get an overview of the current state-of-the-art in self-supervised learning (SSL), particularly its application in HAR and various other methods explored within HAR. The goal of the study is to identify the successful techniques and gaps that exist in the current literature.

The second stage, following the literature study, involves employing an experimental methodology based on the presented research questions above and findings from the literature study. The primary focus of the experimental stage is based on two state-of-the-art self-supervised learning methods developed in recent years, namely the SimCLR [Chen et al., 2020] and SimSiam [Chen and He, 2021]. However, the methods will be modified and optimized using a different set of (hyper-)parameters to fit the context of HAR. The methods will also be evaluated in a low-(labeled)data regime and transfer learning environments to assess their general robustness. Multiple models and setups will be constructed and evaluated using the same datasets for a fair comparison. The constructed models will also have high similarity in setups and architecture. This also makes SimCLR and SimSiam a reasonable choice for contrastive and non-contrastive methods, as they mostly differ in how they learn from examples while keeping architectures and a number of trainable parameters similar.

1.3 Thesis Structure

- **Chapter 2: Background theory** In Chapter 2, we introduce the theories and previous research that we directly use or that guide our work in this thesis. It also includes a section on previous research that are similar or related to the methodology used in this thesis.

- **Chapter 3: Methodology** In Chapter 3, we provide an in-depth overview of the datasets that are being used for our study, as well as a detailed description of the architectural design for the models accompanied with figures. A description of the training- and validation process will also be given.

- **Chapter 4: Experiments** This section clearly describes how we carry out our experiments. It includes the step-by-step plan, the setup of the experiment with necessary

preprocessing, and hyperparameter optimization. Lastly, the results from the conducted experiments are presented.

- **Chapter 5: Discussion** Highlights the key findings of the results, what insights they give, trends, and some limitations with the conducted experiments.
- **Chapter 6: Conclusion and future work** Finally, in Chapter 6, we wrap up our findings and point out possible areas for more research.

Chapter 2

Background Theory & Related Work

This chapter gives an overview of the key theories and research relevant to our work on self-supervised human activity recognition. First, we'll cover the main ideas and theories that our methods rely on. Understanding these will give you a clear picture of how our methods work. Then, we'll look at what other researchers have found in this area. By seeing where our work fits in with other studies, you'll get a better idea of how our methods could be useful and how it relates to previous studies. Certain studies uncovered during the literature review process will not only influence our choice of methods but are also important in order to address the third subquestion of research question two above later in this thesis. This chapter presumes that the reader has a strong technical foundation, including a solid understanding of calculus and linear algebra.

2.1 Background Theory

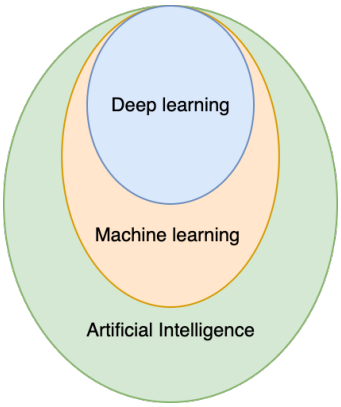
2.1.1 Machine Learning

Machine learning is as shown in figure 2.1a a sub-field of artificial intelligence (AI) that focuses on creating computer algorithms that can gradually improve from experience [Mitchell et al., 2007]. In machine learning, the computers are trained on available data to make predictions on some task, without being explicitly programmed to do so. Deep learning is again a subgroup of machine learning that uses artificial neural networks with three or more layers to make predictions. These layers include one input layer, one hidden layer, and one output layer with more details in section2.1.2 below. Machine learning is typically divided into three categories: Supervised learning, unsupervised learning, and reinforcement learning. The subgroups of machine learning are visualized in fig 2.1b. In supervised learning, the algorithms rely on labels to improve their decision-making. These labels represent the truth and are ideally the output predictions we want the algorithm to make.

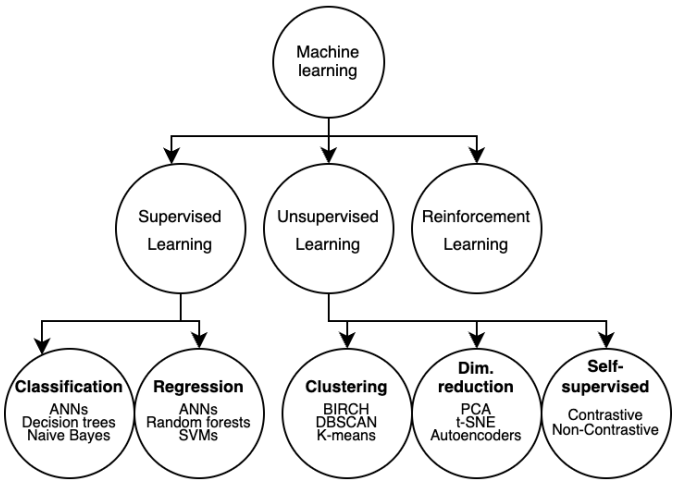
In unsupervised learning, the algorithms must find structure and correlation within the data and typically group the data without having access to the labels. Clustering is a type of unsupervised learning that focuses on creating subgroups within the data so that each observation in that group is more similar to each other given some similarity criteria. In dimensionality reduction

algorithms, the goal is to map the data from a high-dimensional space into a low-dimensional space by maintaining the most important properties of the data. High dimensional data can make it hard to train a machine learning model as many dimensions might not be valuable for the model and can therefore be considered as noise. In self-supervised learning, the model is given an auxiliary task to solve where labels are created from the data itself. Self-supervised learning is normally combined with supervised training. For this reason, the information learned when solving the self-supervised task should be valuable for downstream supervised training. The situation where unsupervised learning is combined with supervised learning is called semi-supervised learning. This is especially advantageous in situations where a large amount of data is labeled, but only a small fraction is labeled. A model can first be trained on the large unlabeled dataset to find patterns or clusters in the data without access to labels.

In reinforcement learning, an agent in the form of a computer program engages in a dynamic environment and learns optimal decisions through trial and error. The agent receives feedback on the performed actions and adjusts its actions based on this feedback. This method is mostly used in situations that include a complex environment, an agent, and a goal and there exist multiple sub-optimal paths to achieve that goal. The method is therefore widely used within games and robotics.



(a) AI subgroups



(b) Subgroups of machine learning

2.1.2 Multilayered perceptron

A multilayer perceptron (MLP) is a neural network consisting of several layers of interconnected neurons. Since all the information goes in the forward direction from the input layer to the output layer, it is also known as a feedforward neural network. This is distinct from the recurrent neural networks and LSTMs where the layers contain loops. The most simple MLP consists of an input layer, a hidden layer, and an output layer. However, an MLP can also contain multiple hidden layers. The first layer in the network, the input layer, takes in raw data and sends its output to the first hidden layer of the network. Each hidden layer contains multiple neurons and each neuron in one layer receives input from all neurons in the preceding layer and forwards its output to all neurons in the next layer, given that the network is fully connected. Most calculations are carried out in the hidden layers. These layers apply weights, biases, and activation functions to extract features that are useful for making accurate predictions. The last layer, the output layer, makes the final prediction based on the features created from the hidden layers. MLPs can be used both for regression tasks and classification tasks. In regression tasks, the output layer of the MLP outputs the predicted value(s) for the task. In classification tasks, the MLP outputs a probability distribution over all the possible classes. In multi-class classification, there is only one correct class and the predicted class is therefore the class that has been assigned the highest probability by the model. Thus, the assigned probabilities for each class should normally sum up to 1. In multilabel classification, however, there can be more than one correct class.

A multilayer perceptron can mathematically be expressed as functions that map input data to output predictions. The parameters of these functions are a set of weights and biases for each layer that are learned from training data. Given an input vector x and a set of weights w and biases b for the first layer, the output from one neuron from the multilayered perceptron can be calculated as

$$z_{i1} = f\left(\sum_{i \in I}(x_i * w_{i1}) + \theta_1\right), \tag{2.1}$$

Where i denotes input i from the input layer. F is called the activation function and is applied to the sum of the weighted inputs and one bias term theta. The job of the activation function is to introduce a non-linearity to the network. Common activation functions are the tanh activation function, the sigmoid activation function and the rectified linear unit activation function (ReLu) with their respective equations shown below.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.4}$$

All the activations from the first layer are then passed as inputs to all neurons in the following layer where similar operations occur, including summing weighted inputs and applying an activation function. The function for calculating the output of one neuron is similar to equation 2.1:

$$z_{i2} = f\left(\sum_{i \in I}(z_{i1} * w_{i2}) + \theta_2\right), \tag{2.5}$$

where z_{i_2} denotes the output from neuron i in the current layer (layer 2) and z_{i_1} denotes the output from neuron i in the previous layer (layer 1). Note that there is only one bias term added to the sum of the weighted inputs, denoted by θ_1 for layer 1 and θ_2 for layer 2.

This process is repeated for each additional layer in the network until the propagation has reached the output layer of the network. In this layer, the output is passed through an activation function that produces a vector of predictions or class probabilities. For a classification task with only one correct class, the softmax activation function is normally used with the formula

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}, \quad i = 1, 2, \ldots, n. \tag{2.6}$$

As seen from the formula, the standard exponential function is applied to each element Z_i which is divided by the sum of these exponentials. This ensures that the sum of the components for the output vector is 1. During the training of the network, the weights and biases of the network are adjusted in a way that minimizes the output of the loss function. The loss function calculates how far off the predictions are from the true values. There are multiple loss functions, but a common loss function used for classification tasks is the cross entropy loss with the formula

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2.7}$$

where $y_{c,o}$ is a binary indicator of whether label c is the correct classification for observation o. For multiclass classification where the binary indicator is only 1 for the correct class, the formula now becomes $log(p_{o,c})$ where $p_{o,c}$ is the predicted probability of the model for the correct class. An important characteristic of the cross entropy loss is that the function explodes rapidly when $p_{o,c}$ is close to zero. In backpropagation, the weights are adjusted in a way that minimizes the loss. By calculating the partial derivative of the loss with respect to the weights, we can adjust the weight of node i in layer j by the following formula

$$w_{i,j,new} = w_{i,j} - \alpha \frac{\partial E}{\partial w_{i,j}}. \tag{2.8}$$

where $w_{i,j,new}$ is the updated weight, $w_{i,j}$ is the old weight and α is a learning rate. When the derivative of the last term is positive, increasing the weights will increase the loss which is not what we want. Conversely, if the derivative is negative, increasing the weights will decrease the loss. Subtracting the last term will therefore give us the desired result. From the chain rule, the derivative of the loss with respect to the weights involves taking the derivative of the loss with respect to the output multiplied by the derivative of the output with respect to the weights by the following formula

$$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial w_{i,j}} \tag{2.9}$$

The first derivative is the derivative of the loss concerning the output. As the cross entropy -log(z_j) explodes when the z_j becomes small, the derivative of the loss with respect to z_j becomes large. The result is that the weights will be heavily adjusted when z_j is very small, i.e. when the prediction is bad.

2.1.3 Convolutional Neural Network

A convolutional neural network is a neural network that consists of one or more convolutional layers. A convolutional layer is a layer that consists of a matrix of interconnected neurons that

most often are arranged in a two-dimensional structure. As with the multi-layered perceptron, the convolutional neural network uses the backpropagation algorithm to update its weights. However, instead of having a structure of neurons where each neuron in a hidden layer is fully connected with all neurons in the previous and preceding layer, the CNN uses a convolutional filter to extract valuable features. The convolutional filter is applied over the input from the previous layer by multiplying the weights of the filter with its input and then summing these products. This is especially advantageous for two reasons: Firstly, there are far fewer weights associated with a convolutional layer as the same filter is used over again over the entire input. In comparison, the number of weights to be updated in a fully connected layer grows large quickly as each connection between two layers is associated with a weight. Secondly, since each neuron is connected to only a small local region of the data, it allows each layer to focus on specific spatial regions of the data, which is a desirable characteristic for data like images.

One convolutional layer takes an input I and applies the convolution C = I * K for one area of the input depending on the filter size. For a two-dimensional image I, the convolutional output C[i,j] for one specific area of the input image is mathematically defined as

$$C[i,j] = \sum_{u=-k}^{k} \sum_{u=-k}^{k} I[u,v]K[i-u,j-v], \tag{2.10}$$

where K is the kernel (filter) for that layer. This operation is applied over the whole image by moving the filter S strides to the side after each repetition. As with the MLP, it is normal to use an activation function after applying the convolutional filter to the input.

2.1.4 Self-Supervised- and semi-supervised learning

Self-supervised learning (SSL) is, as seen from 2.1b, a subgroup of unsupervised learning that focuses on exploiting large amounts of unlabeled data to train a model and learn valuable features. In SSL, the model is trained by solving an auxiliary task that can be created from the already existing unlabeled data. The trained model can afterward be used in a semi-supervised setting as a pre-trained feature-extractor (encoder) for a downstream model that can be finetuned on a smaller labeled dataset available. During this transfer, the weights of the encoder are usually *frozen* which means that they are not adjustable during backpropagation in supervised training. This reduces the dependency on a large-scale labeled dataset that can be costly to create. Sometimes, specific parts of the encoder are *unfrozen* during supervised fine-tuning, which means that the weights of certain layers will be trainable.

Self-supervised learning has multiple advantages which have contributed to a growing interest within the field. The most important advantages of using self-supervised learning are

- **Cost-effective:** The first and most obvious advantage is the reduced need for large-scale labeled datasets as valuable features. This is important because data can be easy to acquire, but labeling can be a costly process that involves a human expert for annotation. Self-supervised learning methods can perform extremely well in a few-shot learning setting where the model is pre-trained on a large-scale unlabeled dataset and afterward finetuned on just a few labeled examples [Chen et al., 2020].
- **Scalable:** Another advantage is the ability to scale the models. In general, more data is needed as the model-size increases due to the increase in optimizable parameters. A large amount of unlabeled data allows for bigger models with more parameters that can

solve more complex tasks. When the performance of supervised methods declines with an increase in the number of parameters for the backbone encoder, Balestriero et al. [2023] find that the performance of self-supervised learning (SSL) methods often increases with the same increase in parameters.

- **Generalizable and transferable:** Self-supervised models trained on unlabeled data extract features that can be valuable for a large range of supervised tasks. This is because it learns the underlying structures of the data [Balestriero et al., 2023]. Related to being generalizable is the robustness of SSL models in transfer learning settings where the representation learned from one task is applied to a different task. More specifically, a model trained on a specific dataset might be used to solve related tasks on a different but related dataset [Ericsson et al., 2021]. For HAR, having these properties mean that we can more easily use knowledge of previous sensor streams to recognize new activities that share similar characteristics without collecting a completely new and large labeled dataset.

Generally, self-supervised methods can be either contrastive or non-contrastive. A term that frequently occurs in self-supervised techniques is *energy*, which is a single number that reflects the distance or compatibility between two samples x and y. The energy function is the function used to compute this compatibility. In contrastive methods, the goal is to produce high energy for samples of data that are incompatible with each other and at the same time produce low energy for samples of data that are compatible. In Natural Language Processing (NLP), a common technique is to mask out certain words of a sentence x to produce a corrupted version y of that sentence. The pre-training task is to reproduce the original input x by predicting the corrupted parts of the sentence. Uncorrupted text will have a large reconstruction error while corrupted text will have a low reconstruction error. Interpreted as energy, the uncorrupted text will thereby have low energy (similar samples) while corrupted text will have large energy reflecting the difference between x and y due to the corruption. The model outputs the probability distribution over the words in the vocabulary for each masked word.

However, these types of techniques are not as easily applicable in fields like vision because there exists almost an infinite number of images. This also applies to time-series data from sensors where a signal can exhibit infinite variations. However, siamese networks are a method that has gained popularity in vision in recent years for its performance when used in self-supervised learning. Siamese networks are usually composed of two neural networks with similar architectures that share their weights and biases and are applied to two or more inputs. The output of these two branches of networks are compared and a similarity is calculated. The input to a Siamese network is different alterations of the same input. The alterations that originate from the same sample are called *positives*, and the pretextual task is to maximize the similarity of these pairs, i.e. decrease the energy for positive pairs. A phenomenon frequently occurring when applying this technique directly is collapsing. This happens when the model always outputs the same constant trivial solution resulting in the same energy for non-compatible input pairs as for compatible, but altered input pairs [LeCun, 2021]. Various implementations of the Siamese architecture have been developed that try to address the problem of dimensional collapse. This includes SimCLR [Chen et al., 2020], BYOL [Grill et al., 2020], SWAV [Caron et al., 2020] and SimSiam [Chen and He, 2021].

2.1.5 Siamese architectures

SimCLR

In SimCLR, the issue of collapsing is addressed by including *negatives* in the calculation of the loss. Negative pairs are augmented pairs that originate from different samples. The learning objective of SimCLR is to attract positive pairs (produce low energy) and at the same time separate negative pairs (produce high energy) in the latent space. This is called *contrastive learning* because both negatives and positives are utilized in the training process. And since the cost function now includes the similarity of negative pairs, collapsing negative pairs to the same constant will result in a higher cost which will be penalized during backpropagation. The SimCLR is visualized in figure 2.3a, and the pseudocode for the method can be found in the appendix. SimCLR utilizes the infoNCE loss (NCE is short for Noise-Contrastive Estimation) developed by Oord et al. [2018]. For one positive pair (i,j) in an input batch of size N, the InfeNCE loss $\ell(i, j)$ is defined as

$$\ell(i,j) = -\log\left(\frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau)}\right) \tag{2.11}$$

where $sim(z_i, z_j)$ is the calculated similarity of the encoded positive pair (i,j) and τ is called the temperature which is a constant between 0 and 1. Chen et al. [2020] suggests that choosing a suitable temperature value can assist the model to effectively learn from difficult negative examples (hard negatives). Hard negatives are negative samples that might have characteristics that are related to the positive pair. As seen from equation 3.4 the numerator consists of the similarity of the positive pair (i,j) while the denominator consists of the sum of the similarities between sample i and all other pairs. With everything else unchanged, bigger similarities of positive pairs yield a fraction closer to 1 resulting in a log-loss closer to zero. Conversely, smaller similarities of negative pairs with else left unchanged also yield a smaller loss due to a decreasing denominator, such that the fraction also becomes closer to 1. The ideal situation is therefore big similarities between positive pairs and small similarities between negative pairs. This loss is calculated for all positive pairs (i,j), and the total loss is the sum of the loss for all positive pairs. However, since the same sample is augmented in two different ways, we end up calculating the similarity of two samples twice; i with j, and j with i. This detail can be seen from the similarity matrix in 2.2 where a small mini-batch of three samples are augmented in two different ways; $[x_1, x_2, x_3]$ and $[y_1, y_2, y_3]$. The infoNCE begins by concatenating the two augmentations into one vector of size 2N. The similarity of each combination of two entries in the vector produces the similarity matrix. The bold entries are the positive samples used in the numerator of equation 3.4 and every other entry on that row are the negative samples used in the denominator, except for the entries on the diagonal. These similarities between the same augmentation of the same sample are always 1 and are therefore omitted when calculating the loss. Note how the similarities below and above the long diagonal are mirrored.

Cosine simularity

There are various options for similarity functions when comparing two vectors. However, cosine similarity is commonly used in machine learning when calculating the similarity between two vectors. In cosine similarity, the cosine of the angle between the vectors is used to reflect the similarity of these vectors. This means that aligned vectors pointing in the same direction produce a maximum cosine similarity of 1 reflecting that the samples are similar, while vectors pointing in the opposite directions produce a minimum cosine similarity of -1 reflecting the dissimilarity of

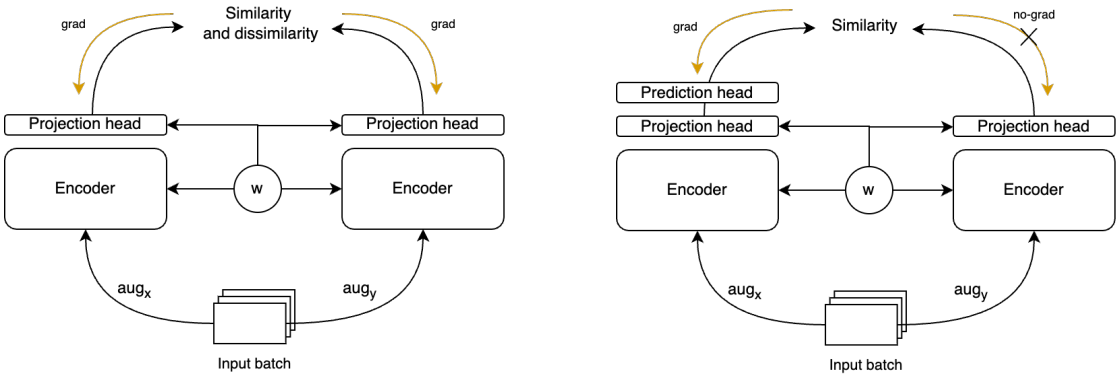
	x_1	x_2	x_3	y_1	y_2	y_3
x_1	1	$s(x_1, x_2)$	$s(x_1, x_3)$	$\mathbf{s(x_1, y_1)}$	$s(x_1, y_2)$	$s(x_1, y_3)$
x_2	$s(x_2, x_1)$	1	$s(x_2, x_3)$	$s(x_2, y_1)$	$\mathbf{s(x_2, y_2)}$	$s(x_2, y_3)$
x_3	$s(x_3, x_1)$	$s(x_3, x_2)$	1	$s(x_3, y_1)$	$s(x_3, y_2)$	$\mathbf{s(x_3, y_3)}$
y_1	$\mathbf{s(y_1, x_1)}$	$s(y_1, x_2)$	$s(y_1, x_3)$	1	$s(y_1, y_2)$	$s(y_1, y_3)$
y_2	$s(y_2, x_1)$	$\mathbf{s(y_2, x_2)}$	$s(y_2, x_3)$	$s(y_2, y_1)$	1	$s(y_2, y_3)$
y_3	$s(y_3, x_1)$	$s(y_3, x_2)$	$\mathbf{s(y_3, x_3)}$	$s(y_3, y_1)$	$s(y_3, y_2)$	1

(2.12)

Figure 2.2: Similarity matrix for the InfoNCE loss using a batch size of three for simplicity. Positives are marked in bold.

the samples. The cosine similarity between two vectors v_1 and v_2 is calculated by the dot-product of the vectors divided by their absolute lengths as shown in formula 2.13 below.

$$\cos(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}, \tag{2.13}$$



(a) SimCLR (b) SimSiam

Figure 2.3: Siamese architectures. The input batch is augmented in two different ways, the data is encoded and in the end, the loss is calculated based on the similarity (and dissimilarity) of positive (and negative samples) samples. Gradient calculations based on this loss are propagated backward.

SimSiam

SimSiam (Simple Siamese) [Chen and He, 2021] and BYOL (Bootstrap Your Own Latent) [Grill et al., 2020] are examples of *non-contrasive* self-supervised learning methods based on siamese architectures. BYOL avoids the collapsed solutions by using a momentum encoder. In BYOL, there is no direct weight sharing between the two Siamese networks. However, one network called the target network, updates its weights based on a moving average of the online network parameters. In this way, the network representations will always stay different ensuring that collapsed solutions are avoided. Chen and He [2021] argues that the momentum encoder in BYOL is not necessary to avoid collapsed solutions. In SimSiam, collapsed solutions are avoided

by introducing asymmetric operations to the network. More specifically, the prediction head ensures that the resulting representation created from both sides of the network is different and will therefore not just collapse to the same solution. Additionally, Chen and He [2021] show that the stop-grad operation is essential for the network to not just find a degenerated solution caused by collapsing. The stop-grad operation prevents gradients from flowing backward and between the two branches of the network and thus prevents the network to learn identical functions. The general SimSiam architecture is visualized in figure 2.3b and the corresponding pseudocode is attached in the appendix. Unlike the SimCLR, the SimSiam uses the similarity between the positive samples directly within the loss function, represented by the following formula:

$$L = \frac{1}{2}D(p_1, \text{stopgrad}(z_2)) + \frac{1}{2}D(p_2, \text{stopgrad}(z_1)) \tag{2.14}$$

In this formula, p represents the vector output of the prediction head, z represents the projection output, their indices represent the branch, D is the similarity function and *stopgrad* means that gradients from these variables will not be used to update the model weights and therefore be treated as constants. The similarity function D used by Chen and He [2021] is the cosine similarity from formula 2.13

Sampling bias

One issue with contrastive learning is that if the selection of negative pairs is done randomly, it can create negative pairs of samples that originate from the same class and thereby push these samples away from each other in the latent space. In a non-contrastive architecture like SimSiam, this is not a problem as only the positive samples are used during loss calculation. This problem is highlighted in Chuang et al. [2020], where the sampling bias leads to large performance drops compared to a completely unbiased negative sampling. For this reason, Chuang et al. [2020] developed a debiased contrastive objective that accounts for the sampling of data points with the same labels, even when the true labels are not available. Robinson et al. [2020] investigates the sampling strategy and shows that sampling hard negatives are the most informative samples for the self-supervised model. Hard negatives are the samples that have the biggest similarities to the anchor and ideally belong to a different class. As formulated by the author, the hard negatives are the ones that the encoder is currently wrong on. For this reason, Robinson et al. [2020] proposes a strategy that samples hard negatives in combination with debiasing, and shows that this can increase the performance of the self-supervised model.

2.1.6 The Human Activity Recognition Chain

In response to the rapid advancements in the field of Human Activity Recognition, Bulling et al. [2014] introduced the concept of the human activity recognition chain in 2014. This paper discusses multiple challenges that are associated with HAR and is one of the most cited papers within the field due to early providing a thorough description of the proposed HAR chain. Among the challenges mentioned are intraclass variability and interclass similarity. Intraclass variability underlines the fact that the same activity may be performed differently by different individuals, while interclass similarity suggests that distinct activities could be executed in similar ways or result in comparable sensor outputs. Another notable challenge is the null class, which is all activities not included in the predicted classes. Due to the potentially infinite number of irrelevant activities, tackling the null class can be difficult. Furthermore, HAR datasets tend to be highly imbalanced, with a limited number of classes, such as sleeping or sitting, recurring frequently. As Bulling et al. [2014] suggests, potential solutions to this imbalance might involve oversampling,

creating artificial data, or accumulating additional data. The requirement for expert annotation of HAR datasets with ground truth labels is another considerable challenge, as this process is both costly and time-consuming. While post hoc labeling based on video recordings can facilitate annotation for stationary or lab-based activities, the complexity increases in free-living settings, where accelerometers collect the data. Bulling et al. [2014] proposes various techniques to counter these challenges, including active and semi-supervised learning, the latter of which will be examined in section 2.2. The paper also points out that, at the time of its writing, HAR was a developing field lacking standardized, high-quality datasets. However, the situation has since improved, with a variety of quality datasets now available, such as MotionSense [Malekzadeh et al., 2018a], HARTH [Logacjov et al., 2021], and WISDM [Weiss, 2019], although most are recorded in a laboratory setting.

2.1.7 Clustering

Clustering is a fundamental unsupervised learning technique in machine learning and data mining. It aims to group data points (also called instances or samples) based on their similarities or relationships, such that data points within the same cluster are more similar to each other than to those in other clusters. Clustering can help reveal the underlying structure of the data, discover patterns, and reduce the dimensionality of high-dimensional datasets.

Two examples of popular clustering algorithms are BIRCH and K-means clustering which represent two different clustering methodologies, namely hierarchical clustering, and partitioning. In hierarchical clustering, the clusters are created by iteratively merging or dividing clusters based on the similarity of the data points. The method creates a dendrogram with different hierarchies of clusters, making it flexible to different cluster sizes as the dendrogram can by cut at any level to create clusters of different sizes. In non-hierarchical clustering or partitioning clustering, the data points are divided into a fixed number of clusters by using a distance measure.

In k-means clustering the dataset is partitioned into K distinct non-overlapping clusters based on the Euclidean distance between the data points. Iteratively, the algorithm adjusts the cluster centroids, which is the mean of the data points, until convergence. The algorithm for K-means clustering proceeds as follows:

1. Initialize K cluster centroids randomly or using a heuristic, $C = \{c_1, c_2, \ldots, c_K\}$, where K is the number of clusters to be created.

2. Assign each data point x_i to the nearest centroid c_j, minimizing the squared euclidean distance $\|x_i - c_j\|^2$. The objective function is given by:

$$J(C) = \sum_{i=1}^{n} \sum_{j=1}^{K} z_{ij} \|x_i - c_j\|^2, \tag{2.15}$$

   where z_{ij} is a binary indicator variable, $z_{ij} = 1$ if x_i belongs to cluster j, and 0 otherwise.

3. Update the centroids by computing the mean of all data points assigned to each centroid:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i, \tag{2.16}$$

   where C_j denotes the set of data points belonging to cluster j and $|C_j|$ is the number of data points in cluster j.

4. Repeat steps 2-3 until the change of the position of the centroid is below a predefined threshold ϵ, i.e. the algorithm converges.

2.1.8 Metrics of performance

Regarding the assessment of a machine learning model, there is a wide array of metrics to choose from and it is essential to pick a metric that is compatible with the intended results of the model. Accuracy is the most straightforward metric and reflects the number of correct predictions that a model makes. Accuracy is quite easy to understand and is suitable for some situations. However, it does not take into account the associated costs of false positives and false negatives, i.e. precision and recall, which can be particularly crucial for specific types of problems.

Precision measures the proportion of true positives among all positive predictions made by a model. Precision is particularly useful for imbalanced datasets, as it takes into account the rarity of the positive class. However, it does not account for false negatives. Recall measures the proportion of true positives among all actual positives in the dataset. The recall is also useful for imbalanced datasets, as it takes into account the rarity of the positive class. However, it does not account for false positives. Most often, precision and recall are reported together in order to see the full picture of false positives and false negatives.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

In an ideal scenario, we would aim for both precision and recall to be as high as possible. However, there is usually a counterbalance between these two parameters. Enhancing precision could unintentionally result in a decrease in recall. The F_β score takes into account both precision and recall and can weight either of the two higher according to their importance by assigning a weight β:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}} \tag{2.17}$$

For example, for extreme minority classes, it can be of high importance to find the ones that exist (high recall) at the expense of getting some extra false positives (low precision). The F1 score is a special case of F_β where precision and recall are considered to be of equal importance by assigning β to be 1. The F1 score now becomes

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{2.18}$$

and is the harmonic mean of the two. Due to the nature of a harmonic mean, the F1 score will always be closer to the lowest value of precision and recall and therefore punish extreme values. For example, a classifier that only predicts one of the classes will result in an F1 score of zero. F1-score can both be calculated for each class and or calculate the average of all classes. The average F1 score can either be weighted or unweighted (macro), where weighting means that the F1 score for each class is scaled corresponding to the frequency of that class. When unweighting the F1 score for each class, classes that do not show up very often will still be as important as classes that are very frequent. The advantage of using macro F1 is that some classes might be very important to classify correctly even though they are not very frequent. The choice between using a weighted or macro F1 score depends on the specific problem being addressed.

2.1.9 TSNe plots

A t-SNE (t-distributed Stochastic Neighbor Embedding) plot is a machine-learning algorithm used for visualizing high-dimensional data. It is a non-linear dimensionality reduction technique that creates a map of the data in two or three dimensions. The t-SNE algorithm finds patterns in the data by measuring the similarity between data points and then using that information to reduce the data into a smaller number of dimensions. The resulting plot is a visual representation of the data with clusters of data points that are close together in the high-dimensional space being represented as close together in the t-SNE plot. TSNe is an especially popular tool used to visualize embeddings created by self-supervised algorithms in order to see if the method manages to create useful features for a downstream supervised task. For SSL it is desirable for the TSNe plot to show clusters of data corresponding to the actual labels.

2.2 Related Work

A structured literature review (SLR) was conducted in the project thesis preceding this master thesis and includes details on the process. The SLR protocol followed the guidelines outlined by Kofod-petersen [2015], which involves the three phases of planning, conducting, and reporting. The main findings from the literature review are presented in the proceeding subsections.

This literature review examines a range of methods for human activity recognition (HAR) that have demonstrated impressive performance, specifically focusing on diverse self-supervised techniques. Recent years have seen a dominance of deep learning techniques such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and attention-based models in HAR classification. Recent advances suggest that leveraging self-supervised pretraining (SSL) can deliver impressive results in areas like activity recognition, speech recognition, and natural language processing. The 2018 Turing Prize winner, Yann LeCun, highlighted SSL's potential role in the future of AI, suggesting it as a promising approach in incorporating general knowledge and common sense into AI systems. Crucially, SSL has proven remarkably effective in situations where labeled data is sparse or expensive to gather, a condition that particularly applies to HAR in free-living conditions. As stated by Haresamudram et al. [2022], research in self-supervised HAR can be grouped into four main strategies: Multi-task SSL, Contrastive Predictive Coding (CPC), Masked Reconstruction, and Autoencoders. The upcoming sections will discuss the first three strategies, supplemented with an overview of supervised deep learning methods for HAR and some conventional machine learning methods, including some findings from related master's theses. To provide some context to the reviewed studies, an overview of the most relevant datasets used in the literature is provided in Table B.1 in the appendix. Furthermore, a summary of the results reported by state-of-the-art methods on these datasets is available in Table B.2.

2.2.1 Contrastive SSL

Saeed et al. [2021] explore the potential of self-supervised learning with contrastive learning to enhance the performance of deep learning models in HAR using unlabeled data obtained from accelerometer sensors. While autoencoders were among the first methods developed for learning from unlabeled data, the authors note that these models tend to use a significant portion of the network's capacity to predict every detail of the low-level input signal, which can be wasteful and unnecessary for downstream tasks.

In their research, Saeed et al. [2021] use Scalogram-Signal Correspondence Learning (SSCL) to pre-train an SCN network for binary classification, aiming to align signal-scalogram pairs. Figure 2.4 shows a high-level view of the architecture. The SCN, comprised of two sections, processes raw sensor data with convolutional neural networks. The top section utilizes each sensor modality's signal network, while the bottom uses wavelet transforms to create scalograms. The outputs serve to determine the alignment of the scalogram with its raw signal. Post-pretraining, the signal encoder, and the first fusion layer are used for supervised training on labeled data. The SCN outperforms both a randomly initialized model with the same architecture as the SCN and a pre-trained autoencoder.

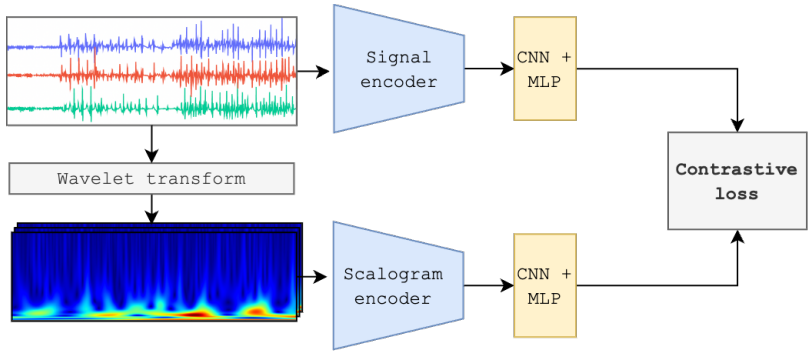


Figure 2.4: Scalogram Contrastive Network (SCN) using the wavelet transform. Each sensor modality is transformed into a 2D scalogram image.

In Rahimi et al. [2022], a similar model was experimented with. However, an additional network was added that only learns pure signal representations, resulting in two encoders: a signal encoder and a scalogram encoder. The idea is that the signal encoder will learn relevant features in the time domain while the scalogram encoder will learn features in the frequency domain. Both encoders were pre-trained separately and used in a downstream network with a trainable MLP head. The results showed that using both encoders resulted in a small improvement in F1 score on all datasets compared to using only one encoder. A drawback is that the study does not experiment with using a signal encoder on both sides of the network, making it hard to conclude whether the improvement comes from having representations of both domains or just that the model contains more trainable parameters. Rahimi et al. [2022] experiments with both adapting the SimCLR solution and the SimSiam for the encoders. As explained in more detail in section 2.1.4, SimCLR and SimSiam consist of similar setups, but differ in how the loss is calculated and propagated backward whereas SimSiam only relies on positive pairs. The authors find that the inclusion of negative samples in the contrastive pre-training significantly improves the model's performance, which is in contrast with the findings of the original paper Chen and He [2021], where only relying on positive pairs and using stop-gradient on one of the branches results in slightly improved results over the SimCLR.

Jain et al. [2022] expands on contrastive learning in Human Activity Recognition (HAR) to include a Time-Synchronous Multi-Device System (TSMDS) via a self-supervised learning model named Collaborative Self-Supervised Learning (ColloSSL). Unlike Saeed et al. [2021], which requires manual transformations, ColloSSL treats sensor data from varied positions as natural transformations of each other. Criteria for choosing suitable positive and negative samples for the anchor are given, prioritizing label similarity, temporal alignment, and device similarity.

To reduce label overlap, a large batch size is recommended. After self-supervised learning on unlabeled data, the model, like in Saeed et al. [2019], retains the weights of the convolutional layers, but allows the final layer to be fine-tuned with the classification head during downstream training. This method results in an almost 8% average F1 score improvement across all datasets compared to the supervised baseline.

Haresamudram et al. [2021] employs the unsupervised approach known as contrastive predictive encoding (CPC) which was originally proposed by Oord et al. [2018]. A core insight from CPC is that predicting multiple future timesteps based on past ones can increase the performance in subsequent tasks. Their experimental results indicate that CPC outperforms three out of four existing unsupervised benchmarks on standard HAR datasets like UCI-HAR. Furthermore, it surpasses the top-performing supervised model deepConvLSTM, developed by Ordóñez and Roggen [2016], in three out of four baseline datasets. Haresamudram et al. [2021] further demonstrates that forecasting multiple future time steps improves the model's capability to generate meaningful sensor data representations for activity recognition. This indicates the importance of carefully choosing an appropriate encoder, ensuring that the pretext task is neither too simple nor too complex to solve.

Khaertdinov et al. [2021] presents the CSSHAR model, an integration of the contrastive framework from Chen et al. [2020] and a transformer encoder. CSSHAR stands for "Contrastive Self-supervised learning approach to Sensor-based Human Activity Recognition". As detailed in the study, CSSHAR uses five signal transformations chosen at random with a probability p, such as jittering, twice on each signal to produce two unique views. The model's encoder is a transformer, which has shown to be successful in previous research like [Zeng et al., 2018] and [Mahmud et al., 2020]. According to Khaertdinov et al. [2021], CSSHAR outperforms all other SSL techniques, especially on the UCI-HAR dataset, with a mean F1 score of 91.14. It surpasses the second-best model, CPC, by 9 percent in the mean F1 score. However, it's advised to interpret these results carefully, as the original CPC authors Haresamudram et al. [2021] claim their model achieves a mean F1 score of 90.24 % on the UCI-HAR dataset.

2.2.2 Multi-Task SSL

The Transformation Prediction Network (TPN) introduced by Saeed et al. [2019] was a pioneering self-supervised model in HAR, inspired by the multi-task technique in Doersch and Zisserman [2017]. It uses eight different signal transformations, with the pretext task of identifying what transformations that were applied. The network has a shared temporal CNN (encoder) and task-specific fully-connected networks. However, only the encoder's frozen weights transfer to the downstream HAR task after pre-training.

The multi-task pre-training yields more useful features than an autoencoder's input reconstruction, with downstream training performance boosted by fine-tuning the pre-trained model's last layer. The self-supervised model learns its features in a similar way as a fully-supervised model, which is evident from TSNe plots, saliency mappings, and SVCCA analysis of unsupervised pre-training and purely supervised learning representations. Ultimately, Saeed et al. [2019] demonstrates the benefits of solving multiple tasks simultaneously with a common trunk of layers.

The multi-task approach taken by Saeed et al. [2019] is also utilized by Tang et al. [2021], who merge the technique with a self-training pipeline, naming it SelfHAR. Self-supervised tasks such as signal transformation discrimination are combined with self-training, according to the authors, to raise the variety of the data, resulting in more generalizable features created by the

model. The self-training part of the method employs the noisy student approach developed by Xie et al. [2020] which, despite its simplicity, has proven to be highly effective for computer vision tasks. Tang et al. [2021] show that combination of signal transformation discrimination with noisy student self-training yields better performances than just using one of the methods separately. Another finding is that in situations with very little labeled data available, other SSL methods can be preferable. The reason for this is the amplification of noise and uncertainty in the teacher-student paradigm when there is a scarcity of labels to learn from. Lastly, Tang et al. [2021] show that the representations learned from SelfHAR are similar to those learned from a fully supervised model. The paper also shows that pretraining on a uniformly distributed dataset that covers all activities on a larger scale is much more beneficial for downstream performance than imbalanced datasets.

2.2.3 Masked Reconstruction

The study Harish et al. [2020] employs masked reconstruction for pre-training, where a small part of sensor data is hidden and predicted using a BERT variant encoder [Devlin et al., 2018]. This encoder comprises a 1D convolutional embedding layer and a Transformer encoder with multiple attention blocks. To address the Transformer's sequential data processing limitation, sinusoidal position embeddings from Vaswani et al. [2017] is used. After the encoder, two fully connected layers make the masked area predictions. Similar to other SSL models, only the encoder is repurposed for the downstream task

Rahimi et al. [2021] also use masked reconstruction, but differs from Harish et al. [2020] by implementing convolutional layers in the pre-training encoder, instead of transformer blocks. While both methods have the pretext task of predicting masked signal regions, Rahimi et al. [2021] also masks entire consecutive samples, unlike Harish et al. [2020] which only masks a small fraction of sensor data. Moreover, Rahimi et al. [2021] introduces cross-dimensional learning, predicting masked regions in one dimension using unmasked regions from another, which is not performed in Harish et al. [2020]. They report an F1 score of 90.1 % and 91.0 % on the UCI HAR and MotionSense dataset, which is an improvement of around 2 % and 10 % respectively over the method proposed by Harish et al. [2020].

2.2.4 SSL in a low-data regime and transfer learning in HAR

self-supervised learning has recently shown promising results in both HAR and other fields like vision. However, the benefits of SSL come especially clear in situations with very little labeled data available or when you only have data for a similar but different task, and acquiring data for the new task is expensive or time-consuming. This is especially showcased for vision in Chen et al. [2020], where other supervised state-of-the-art methods are completely outperformed when only a small fraction of the data is labeled (1 % and 10 %). In the transfer learning setting when the models are trained on the ImageNet and afterward finetuned and tested on 12 other datasets, simCLR outperforms the supervised baseline on 5 datasets while the supervised baseline is only superior on 2. The rest are tied, i.e. not statistically significant.

Similar results are found within HAR by Saeed et al. [2021] where the self-supervised model outperforms the supervised baseline when transferring between the two datasets HHAR and MobiAct. This is the case both with no finetuning on the new dataset, and when finetuning on the new dataset with very limited labeled data. In the semi-supervised setting, the self-supervised

model SCN outperforms the supervised baseline on all datasets, with the difference becoming smaller with an increasing amount of labeled data available. These findings regarding transfer learning and semi-supervised approaches are also supported in other HAR studies [Rahimi et al. [2022], Khaertdinov et al. [2021], Haresamudram et al. [2021], Tang et al. [2021], Saeed et al. [2019]] and it will therefore be interesting to experiment with this in our work as well.

2.2.5 Supervised approaches

Ronao and Cho [2016] demonstrated the efficacy of regular convolutional neural networks in accurately differentiating between similar activities. They recommended the use of wider filter sizes and lower pooling sizes to retain information passed from the input to the CNN layers. Ordóñez and Roggen [2016] proposed a highly competitive supervised model called DeepConvLSTM, which comprises four convolutional layers with 64 and 128 feature maps, followed by two dense layers of recurrent units and a softmax layer. The experiments were conducted using two benchmark datasets, Opportunity [Roggen et al., 2010b] and Skoda [Zappi et al., 2008]. Due to the impressive performance of DeepConvLSTM, their method has been widely used as a supervised baseline in multiple supervised and self-supervised studies in HAR.

The works of Ordóñez and Roggen [2016] highlight the high performance achieved by both convolutional neural networks and recurrent neural networks on the Opportunity and Skoda datasets, surpassing the previous participants in the Opportunity contest. The authors found that the utilization of long short-term memory (LSTM) cells instead of the traditional fully connected layers led to an average improvement of 5% and 6% on the Opportunity and Skoda datasets, respectively. It is noteworthy that the baseline fully connected model involves six times more parameters than the proposed DeepConvLSTM model, which is attributed to the first dense layer's units needing to be linked to each value in the last feature map. The strength of LSTMs and convolutional neural networks in HAR are also supported by others, like Murad and Pyun [2017], Jiang and Yin [2015] and Guan et al. [2007], where the latter also explores using an ensemble of LSTMs. The study conducted by Hammerla et al. [2016] investigates the performance of different supervised models applied to several benchmark datasets for HAR, including the Opportunity dataset, PAMAP2 dataset, and Daphnet Gait dataset (DG). The models tested include standard DNNs, CNNs, and various uni- and bi-directional LSTMs. The study concludes that recurrent neural networks are generally recommended for short-duration activities with a natural ordering, while CNNs are more suitable for longer, repetitive activities, like every-day activities in the HARTH dataset utilized in this thesis. The author also found that adjusting the learning rates has the most significant impact on the model's performance, with regular DNNs showing the biggest spread in recognition performance and performing the worst.

Yao et al. [2019] focus on exploring the efficacy of short-time-Fourier neural networks (short STFNets) combined with convolutional layers in learning signals from the time-frequency domain. The authors propose that the underlying physics of a signal is closely related to its frequency characteristics. Hence, incorporating the short-time Fourier transform of the signal as input to a deep neural network can enhance its performance by enabling it to directly learn features from the frequency domain. The transformation of a signal into the frequency domain is known to cause a loss of information about the time domain. The Short-Time Fourier Transform (STFT) method allows for the selection of a sample length for Fourier transformation, resulting in a trade-off between the loss of information in the time and frequency domains. Longer sample lengths provide better frequency resolution but at the cost of reduced time resolution. The STFNet building block overcomes this limitation by applying multiple transformations with varying time

and frequency resolutions. According to Yao et al. [2019], the STFNet outperforms the previous state-of-the-art DeepSense from Yao et al. [2017] and ComplexNet from Trabelsi et al. [2017].

Some previous master theses at NTNU have tried to tackle the problem of recognizing activities based on accelerometer sensor data. Hessen and Tessem [2016] developed a convolutional neural network to train and make predictions on the Trondheim in-lab dataset. Additionally, experiments with different kinds of techniques like dynamic windowing, self-training, and hidden Markov models are conducted. Dynamic windowing where windows are dynamically created based on some change of energy in the signal did not yield any advantage over using standard static segmentation. Self-training did not yield any improvement in performance either. The authors suggest that the reason for this could be the insufficient quantity of unlabeled data in comparison to labeled data. They propose that utilizing self-training on the complete HUNT4 dataset, which was made available after the publication of this paper, could be advantageous. Furthermore, they did not employ the student-teacher self-training technique, which achieved state-of-the-art outcomes for vision in the study by Xie et al. [2020]. Postprocessing the predictions from the CNN with the use of HMM and the Viterbi algorithm is motivated by the fact that some activities are naturally followed by others. In Hessen and Tessem [2016], the probabilities of transition were determined by employing dataset statistics to assess the likelihood of one activity succeeding another, while the emission probabilities were computed using the posterior probabilities supplied by the classifier. Implementation of the Viterbi algorithm resulted in a modest increase in accuracy for both the adults and adolescents datasets, specifically 0.6% and 1.71%, respectively. Notably, the sensitivity of all activities improved, as indicated by an increase in recall, with the most significant enhancement observed in the activity of walking stairs, with a rise of over 3-5 %. As stated in Vågeskar [2017], the datasets used during the experiments are the Trondheim In-Laboratory dataset and do not necessarily represent the performance for free-living conditions. In fact, Vågeskar [2017] found that the overall accuracy decreased by 20 % when the model was tested on the Trondheim free living (TFL) dataset.

In the research conducted by Skauge [2021], an XGBoost classifier was utilized to classify vigorous activities in the domain of HAR. In this context, vigorous activities were defined as short bursts of incidental physical exertion undertaken during routine daily activities, such as running, jumping, and skipping. The XGBoost classifier is an ensemble-based model that relies on hand-crafted features. As part of this study, a total of 95 distinct features were created in both the time and frequency domains. Of these features, 22 were found to be beneficial in enhancing the model's performance.

Chapter 3

Methodology

In this chapter, we first introduce the datasets that will be utilized for our experiments with details on the collection process and its contents. Then we introduce the model architectures, including some reasoning behind various choices made. This includes both the upstream and downstream models in addition to the pipeline used to train and test the models. The software used to build the various models are listed in the appendix.

3.1 Datasets

3.1.1 HUNT4 dataset

HUNT4 is the fourth iteration of the Nord-Trøndelag Health Study (HUNT) [Åsvold et al., 2023] and one of the primary areas of emphasis in the study was examining the activity patterns within the population. For this reason, 31 295 participants volunteered to wear two sensors that record their movements for 1 week while doing their regular work and activities, i.e. free-living environments. Movements were recorded by accelerometers in 3 perpendicular directions. The accelerometers were mounted on the participant's back and thigh resulting in 6 channels of time series data, as shown in figure 3.2. The data are recorded using the small AX3 axivity accelerometers[1].

3.1.2 HARTH dataset

The Human Activity Recognition Trondheim (HARTH) dataset from Logacjov et al. [2021] is a dataset consisting of accelerometer recordings from 22 participants during regular working hours. The dataset was acquired due to the lack of existing free-living HAR datasets that use more than one sensor and have reliable annotations. The setup used by Logacjov et al. [2021] is of high resemblance with the HUNT4 dataset and uses the same two three-axial sensors with the same sensor position and was recorded for 1.5-2 hours. The participants were told to live their everyday life as normal as possible. However, in addition to various sedentary activities, the participants were instructed to do the activities sitting, walking, standing, lying, and running for at least two minutes. The participants were equipped with a GoPro.

Data for the study was gathered in two separate sessions, capturing the daily physical actions of 15 participants, including six women. In the initial session, participants were directed

[1] https://axivity.com/product/ax3

to continue their usual activities for 1.5-2 hours, engaging in actions like sitting, standing, lying, walking, running, and jogging for at least a couple of minutes each. Acceleration data was captured using two sensors at a sampling rate of 100 Hz. The sensors' measurement range extended to approximately 8 g. To align the acceleration and video signals for later use, participants were asked to execute three heel drops at the start of each recording. In total, the data spanned around 30 hours, with the average recording time being around 120 +- 21.6 minutes. Video data was formatted to 25 fps and 640 x 360 pixels, in which every frame was annotated. In addition to the above activities, participants also performed other tasks like ascending and descending stairs, standing with leg movements, sitting and standing while cycling and sitting and standing while in transport.

Recognizing the skewed representation towards light activities and the lack of certain activities in the dataset collected in the first round, the second data collection session focused more on activities such as walking, running, and cycling (both sitting and standing), incorporating flat, uphill, and downhill sections. There were no specific demands about the location or timing of these activities. Participants also engaged in other activities like sitting, lying down, and climbing up and down stairs, which were annotated appropriately. This second session resulted in approximately 7 hours or 417.6 minutes of recorded data, with an average recording duration of around 60 +- 9 minutes per participant. The ANVIL annotation tool was used to annotate the recordings, and the distribution of various activities is illustrated in figure 3.1.

3.1.3 UCI-HAR dataset

The UCI-Human Activity Recognition (UCI-HAR) dataset is a dataset recorded by Anguita et al. [2013]. The data was collected from a cohort of 30 volunteers, all within the age range of 19 to 48 years. These participants performed six distinct activities, namely: walking, walking upstairs, walking downstairs, sitting, standing, and laying.

During the recordings, the participants were equipped with a Samsung Galaxy S II smartphone around their waist. This device is equipped with a built-in 3-axial- gyroscope and accelerometer which record movements using a sampling rate of 50 Hz. All experiments were recorded using an additional video camera for accurate labeling. The dataset has an almost perfect distribution of activities. The data was recorded in-lab and the participants were told what activity to perform at all times. The resulting dataset is a mixture of data from both the accelerometer and gyroscope, all of which underwent a pre-processing step to reduce noise. Furthermore, the acceleration signal was split into body acceleration and gravity components using a Butterworth low-pass filter. This was based on the assumption that the gravitational force exclusively incorporates low-frequency components. Therefore, a cutoff of 0.3 Hz is used for the filter.

3.1.4 Data preparation

The data is segmented using the sliding window segmentation method described in Bulling et al. [2014]. Dynamic energy-based segmentation was also an option. In energy-based segmentation, each new time window is created based on a change in the energy of the signal over a given threshold. However, this did not yield any benefits over the sliding window technique in Hessen and Tessem [2016] as described in section 2.2. The sliding window method is visualized in figure 3.3. In the sliding window method, a window of a predetermined length slides over the recorded signal and creates window segments. Each window will have a label that corresponds to the

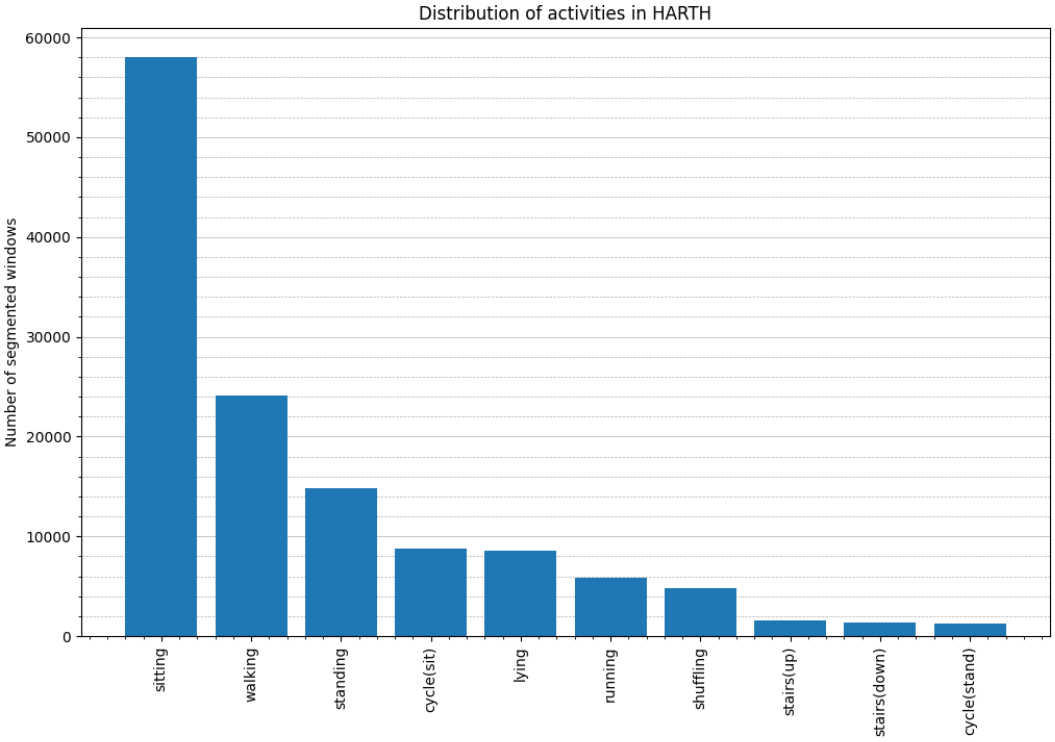


Figure 3.1: Distribution of classes in the harth v1.2 dataset

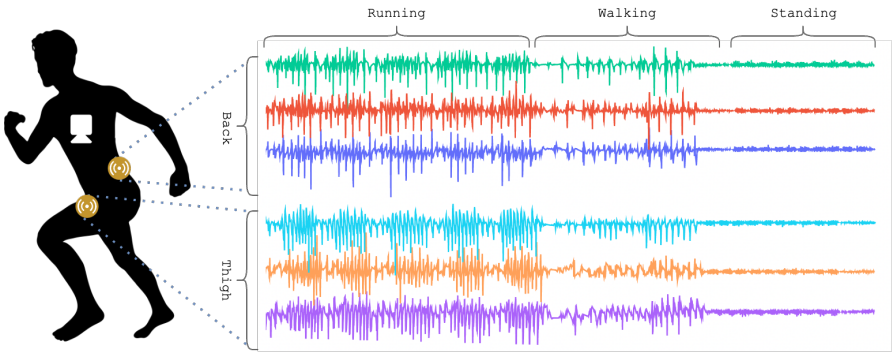


Figure 3.2: Visualization of sensor positions and recordings. The camera was only included when recording the HARTH dataset and not during the HUNT4 study. The three sensor streams from each sensor represent responses in the x, y, and z directions.

performed activity. When one window contains multiple labels, the most frequently occurring activity is chosen. In the example in figure 3.3, a sliding window of size 6 is chosen for simplicity, with 50 % overlap between the segments. To prevent a massive overrepresentation of samples

related to the "lying" activity during the pretraining phase, we exclude large portions of the sensor data from HUNT4 recorded during nighttime hours, namely between 12 a.m. and 6 a.m. We do not convert the windows to spectrograms but rather use the raw data with some additional preprocessing described in section 4.2.2.

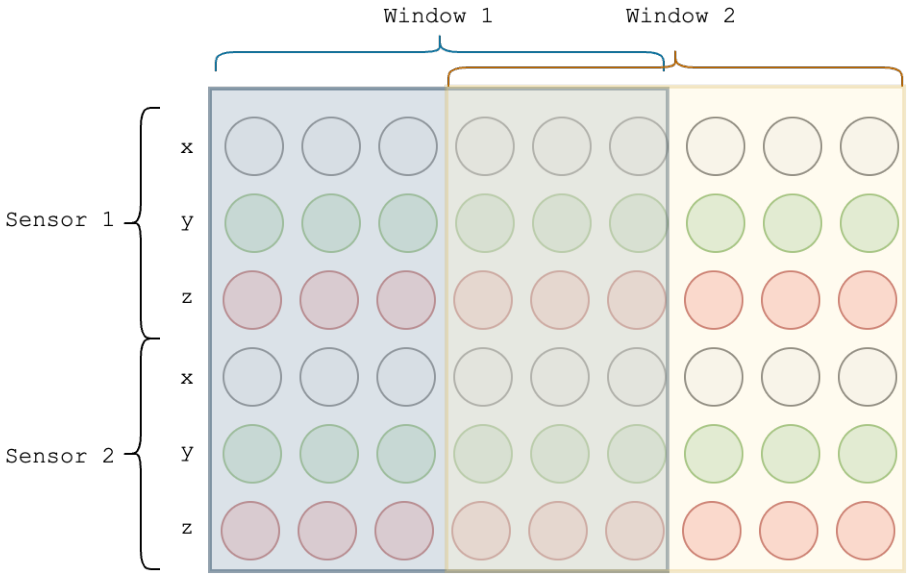


Figure 3.3: Static sliding windowing method.

3.2 Model Architectures

In this thesis, we're looking at the performance of two key models, SimCLR and SimSiam. They are known for being top performers in the area of self-supervised learning for vision and are also composed of convolutional neural networks which have been shown to be highly effective for HAR, as seen from related work. Contrastive learning, represented by SimCLR in this thesis, has proved to be effective for HAR as seen from related work. Non-contrastive methods, on the other hand, represented by SimSiam, are much less explored for HAR. We will also modify the SimCLR model by incorporating a clustering feature, with the goal of identifying false negatives.

3.2.1 High-level architecture

The high-level process of self-supervised pretraining and supervised fine-tuning is visualized in figure 3.4. The large unlabeled HUNT4 dataset is utilized to train the encoder by solving the upstream pretextual task. Subsequently, the encoder is frozen with the idea of working as an automatic feature extractor (following step 3, feature calculation, of the HAR chain from Bulling et al. [2014]) for the downstream task prediction in addition to a trainable fully connected MLP on top. Ideally, the encoder will create clusters from samples originating from the same activity, where a linear MLP (no hidden layer) will separate these clusters into activities.

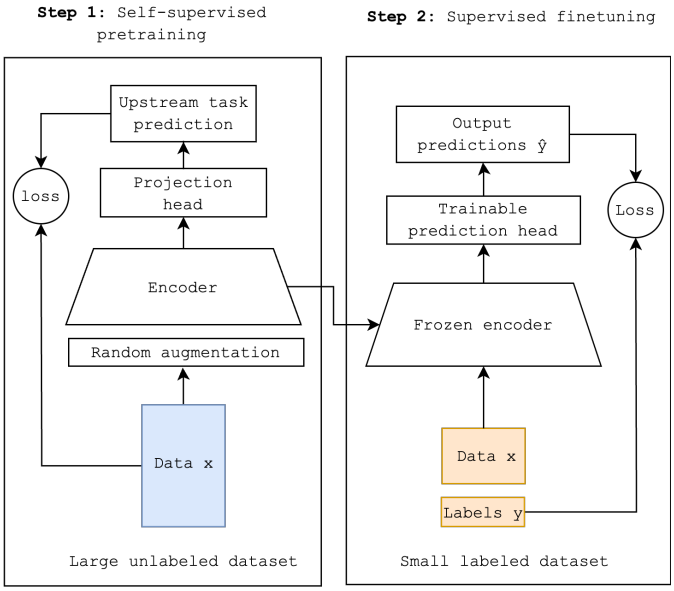


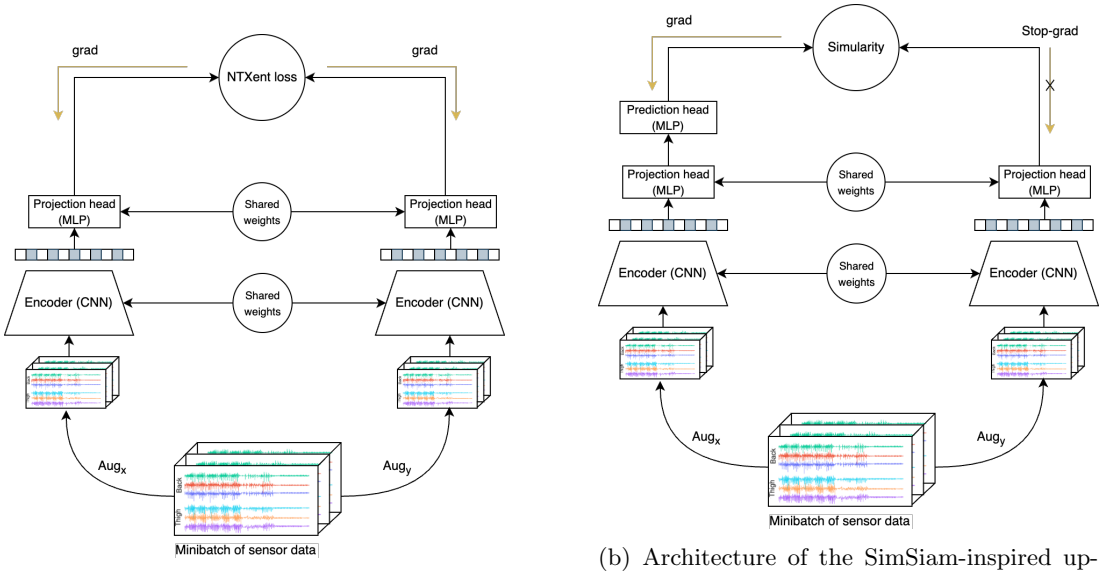
Figure 3.4: High-level architecture.

3.2.2 Pre-training inspired by SimCLR and SimSiam

A more detailed visualization of the architecture of the contrastive and non-contrastive upstream models is shown in figure 3.5 and is inspired by the SimCLR and SimSiam framework from Chen et al. [2020] and Chen and He [2021] respectively. Some inspiration is also taken from Saeed et al. [2021], but without discriminating between the raw sensor signal and its spectrogram. In our work, a mini-batch of sensor data is subject to two different augmentations of the same original sensor data. Note that the sensor data in these mini-batches are already pre-processed and segmented into short time windows. This encoder is originally a convolutional neural network (CNN) [Chen et al., 2020], which specifically for our case consists of three convolutional layers and two fully connected layers as visualized in figure 3.6. Each convolutional layer n consists of C_n filters of size k_n. As the figure shows, each convolutional layer is followed by one ReLu activation function and a dropout. The last block is followed by a global max pool for each channel of the layer to collect the most important features from each channel. This will filter out unnecessary noise, decrease the dimension and also thereby decrease the computational complexity of the fully connected layers that follow. This decrease is also desirable for the clustering application introduced in 4.1.2, as high-dimensional data often can be challenging to cluster. This is especially a problem for algorithms that employ Euclidean distances because each dimension is treated with the same importance when calculating the distance. The global maxPool is inspired by ResNet-18 [He et al., 2016], where a global average pool follows the last conv layer of the network. ResNet-18 is the network used as an encoder for the original SimCLR and SimSiam from Chen et al. [2020] and Chen and He [2021], respectively. However, ResNet is commonly used for datasets such as ImageNet, where the images are typically resized to 224x224 pixels. Applying such a network to 1D sensor signals is excessive and is, for this reason, significantly scaled down. ResNets use residual (skip) connections to mitigate vanishing gradients, a problem often seen in very deep feed-forward neural networks and formalized in Glorot and Bengio [2010]. These connections allow gradients to be backpropagated more directly to facilitate more effective

learning in earlier layers of the network. However, the architecture of the encoder only consists of three convolutional layers and two fully connected layers, so the risk of vanishing gradients is significantly reduced. Therefore, we choose to drop residual connections to avoid unnecessary complexity, still ensuring robust learning with a more efficient model.

The actual implementation of the encoder, including (hyper)parameters, implementation of the convolutional filters, and number of neurons for the fully connected layers are discussed in more detail in section 4.2. Both outputs are flattened into a 1d vector which is input to a projection layer. The projection layer typically takes the form of a multilayered perceptron (MLP). The architecture of the encoders and projection heads are largely the same for both the SimCLR and the SimSiam models. However, there are mainly three differences between the implementation of the SimCLR and SimSiam: 1) The SimSiam model consists of an additional prediction head in one of the branches, which will serve as a countermeasure against the dimensional collapse mentioned in 2.1.4. This prediction head consists of an MLP with a similar architecture as the projection head. 2) The loss of the SimSiam-implementation is a similarity measure between the encoded augmentations. For SimCLR, the NTXent loss from formula 3.4 defined in section 2.1.4 is calculated based on the same similarity measure as used in SimSiam, but also includes similarities between negatives. 3) For SimSiam, one of the branches also employs a stop-grad which prevents certain gradients from flowing backward. As described in 2.1.4, this is also a vital component in SimSiam to counter degenerate solutions.



(a) Architecture of the SimCLR-inspired upstream model

(b) Architecture of the SimSiam-inspired upstream model. Note the extra prediction head, the stop-grad and the loss calculation

Figure 3.5: Architecture of the semi-supervised model

3.2.3 Augmentation functions

The chosen augmentation(s) is an important factor that should be chosen with care for the model to learn valuable features for the downstream task. The augmentations performed should generally represent real variances that might occur from data derived from the same performed

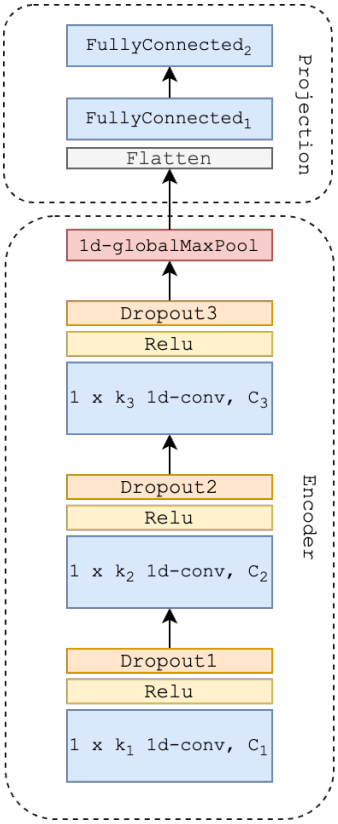


Figure 3.6: Architecture of the encoder (bottom) and the projection head (top) used during pretraining. This encoder is the same for both the SimSiam and the SimCLR.

activities. Therefore, the augmentation should generally be challenging enough for the model to understand the important underlying characteristics for each class, in this case, activity, but not so hard that it completely transforms the input (signal) characteristics (invariances) as highlighted in Tian et al. [2020] for vision. The augmentations can change the signal response by modifying and shifting the signal along the time domain and adjusting and altering the magnitude of the signal. The augmentation functions chosen for the experiments are defined below:

- **Window slice:** Window slicing involves cropping the time series to α % of its original length. In this case, α is set to 0.85. The starting point of the window is randomly selected. The time series is interpolated back to its original length to ensure a fair comparison with other data augmentation methods. This method is the time series equivalent of cropping images.
- **Time warp:** The warping path can be described as a smooth cubic spline curve, which consists of four knots. These knots are given magnitudes that are randomly determined, following a normal distribution with a mean (represented by μ) of 1 and a standard deviation (denoted by σ) of 0.09.
- **Permutation:** Permutation involves rearranging segments within the time series to create a new pattern. This results in a different structured arrangement of the time series data.

- **Jitter:** In jittering, random noise is introduced to the time series data in order to enhance signal variability. The jittering is normally the Gaussian noise $\mathcal{N}(0, \sigma^2)$ where the parameter σ^2 is chosen to be 0.15 in this case. The modified signal x' is defined as

$$x' = x_1 + \epsilon_1, x_2 + \epsilon_2, ..., \ldots, x_T + \epsilon_T; \tag{3.1}$$

- **Magnitude warp:**

  In magnitude warping, the time series data is augmented by warping the magnitude of the response using a smooth curve. The resulting signal x' is defined as

$$x' = \alpha_1 x_1, \alpha_2 x_2, \ldots, \alpha_t x_t \tag{3.2}$$

  where $\alpha_1, \ldots, \alpha_t, \ldots, \alpha_T$ is a sequence created by interpolating a cubic spline $S(u)$.

- **Rotation:** Rotating the time series data involves applying an element-wise random rotation matrix to the signal, denoted as R, where the rotation angle y is drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$. As for jittering, this process introduces randomness and variability to the time series by rotating each element based on the sampled angle.

- **Scaling:** In scaling, a random scalar value is applied to the signal to alter the overall magnitude of the time series. The resulting signal x' after using scaling is defined as

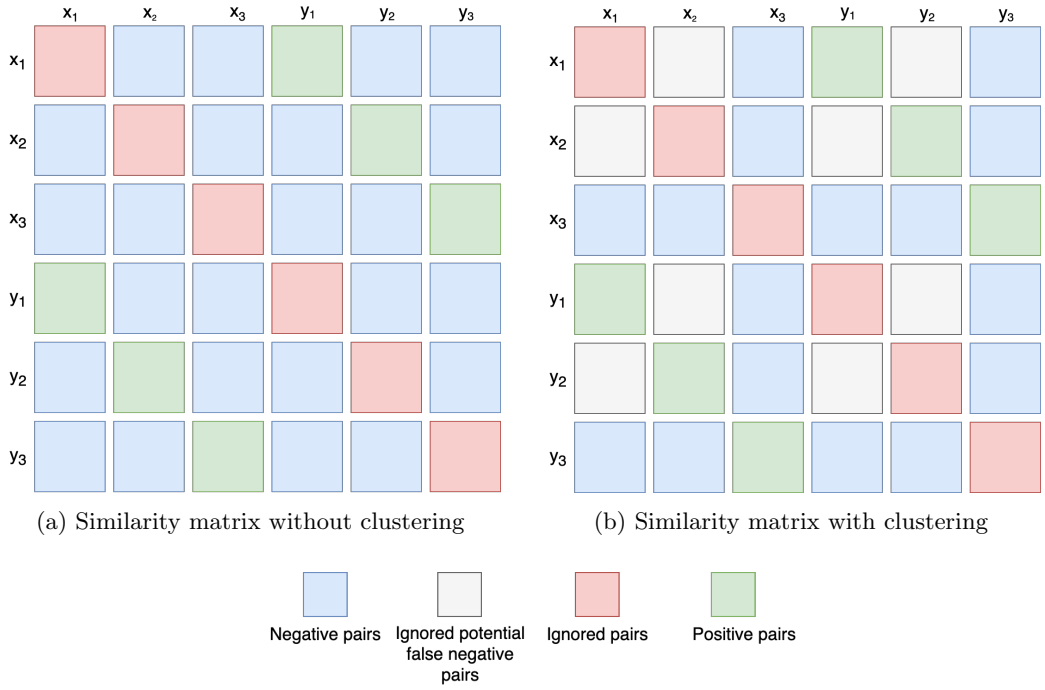
$$x' = \alpha x_1, \alpha x_2, \ldots, \alpha x_t. \tag{3.3}$$

  Scaling alters a time series's overall magnitude or intensity by applying a random scalar value. This scalar value is chosen randomly from a gaussian distribution $\mathcal{N}(0, \sigma^2)$, where sigma is chosen as 0.4 in this particular case.

3.2.4 Unbiased modification of SimCLR

As explained in section 2.1.4, contrastive learning can suffer from a sampling bias where created negative pairs can originate from the same activity. These pairs are called false negatives because they appear as negatives in the similarity matrix but are, in fact, positives. This problem can be amplified when the class distribution is highly skewed, as with the HARTH dataset seen from the distribution of classes in figure 3.1. While pretraining is not done on the HARTH dataset, assuming a similar or worse distribution on the unlabeled HUNT4 dataset is not unreasonable due to the nature of free-living environments. When the datasets are skewed, each training batch will also be skewed towards certain activities, increasing the risk of creating negative pairs of time series samples originating from the same activity, which in this case will be sitting for the most part. Consequently, these pairs get pulled apart in the latent space, which is the opposite of what we want. Thus, it is desirable to modify the proposed architecture shown in figure 3.5a to prevent this from happening. In the absence of labels, we want to make a qualified guess on what samples should be chosen as negatives.

The way we address this is to cluster the latent vectors from the mini-batch. By doing this, we can create negative pairs only from vectors that originate from different clusters where, ideally, each cluster represents one performed activity. This is visualized in figure 3.8 where the latent space consists of 9 different samples originating from 4 different activities. The idea is that SimCLR manages to form clusters of samples originating from the same activity in the latent space so that the chosen clustering algorithm will have an easier time filtering out these activities



(a) Similarity matrix without clustering

(b) Similarity matrix with clustering

Figure 3.7: Similarity matrix for the SimCLR. In SimSiam, only the green squares (positives) are utilized for the loss.

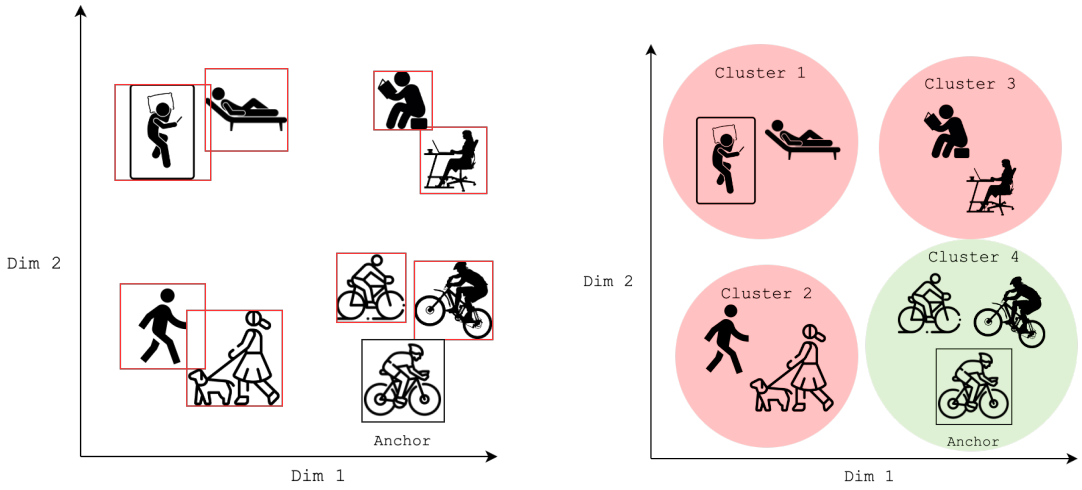
from the similarity matrix. Figures 3.8a and 3.7b show the similarity matrix with and without clustering, respectively. In 3.7b, the false negative pairs are marked as grey. For simplicity, this example shows a batch with three samples augmented in two different ways, $[x_1, x_2, x_3]$ and $[y_1, y_2, y_3]$, and is the same as the one formalized in section 2.1.4. In this example, (x_1, x_2) originate from the same cluster and are therefore ignored as negative pairs. Since x_2 originate from the same sample as y_2, the pair (x_1 y_2) is also ignored. The same logic goes for the pairs (x_2, y_1) and (y_1, y_2), resulting in a total of 4 new ignored negative pairs. Also, since the similarity matrix is symmetric over the red diagonal, the same 4 ignored pairs can be found under the red diagonal. By modifying the loss function in 3.4 to include the clustering, the loss is now defined as

$$\ell(i,j) = -\log\left(\frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i, \text{C}(z_k)\neq \text{C}(z_i)]} \exp(\text{sim}(z_i, z_k)/\tau)}\right) \tag{3.4}$$

Where C(z) returns the cluster index of the latent vector z.

3.2.5 Downstream supervised fine-tuning and testing

Figure 3.9 shows how the downstream supervised training process is performed. Raw data is extracted from 21 subjects, pre-processed, and used for supervised model finetuning. The supervised model utilized is the same as one shown in step 2 in figure 3.4. After each iteration of



(a) Negative selection without clustering. All samples that are different from the anchor are chosen as negatives.

(b) Negative selection with clustering. All samples that are in other clusters than the anchor are chosen as negative.

Figure 3.8: Negative selection with and without clustering from 9 samples of 4 different activities.

leave-one-subject-out cross-validation (LOSO), the predictions for each test participant are stored in a list. When all 22 subjects have been tested, the metrics based on this list are calculated.

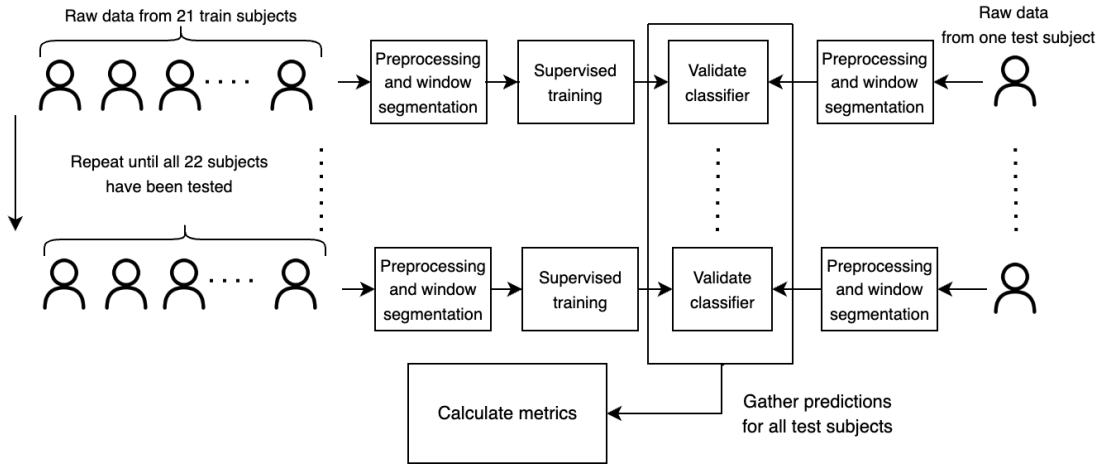


Figure 3.9: Supervised training and validation process using LOSO

Chapter 4

Experiments and Results

Research Question 2 and its associated sub-questions explore the potential application of existing contrastive- and non-contrastive self-supervised learning (SSL) methods to free-living data like HUNT4 and HARTH for human activity recognition. This chapter aims to investigate this issue through a series of experiments. This involves a detailed description of the experiments to be conducted and the experimental setup - which encompasses data preprocessing, augmentation methods, and model hyperparameters. Finally, we will present the results of these experiments.

4.1 Experimental Plan

The experimental plan follows research question 2 and its subquestions from section 1.1. We will evaluate the contrastive model *SimCLR LF*, the non-contrastive model SimSiam, the modification of the SimCLR, and lastly the robustness of the various models in the transfer learning setting and low-data regime.

4.1.1 Experiment 1: Self-supervised contrastive learning for HAR

In the first experiment, the self-supervised model is trained without using any techniques to correct for the sampling bias that might occur with contrastive learning. The contrastive model used is the SimCLR shown in 2.3a. To analyze to what extent pre-training is beneficial for the downstream task, we propose 7 different setups in which one of them are the CNN from Hessen and Tessem [2016], three are different setups of the SimCLR *with* pretraining and three are different setups of the SimCLR *without* pretraining. Due to the confusion this might induce, we will give them names that will be used from here on and throughout the rest of the thesis. Below are the definitions of the models, preceding some reasoning on what insights these setups might give. For some additional clarity: LF = Linear Frozen, HF = Hidden Frozen, HU = Hidden Unfrozen.

1. **SimCLR LF**: Contrastive model pretrained on the HUNT4 data with *frozen* weights and a *linear* prediction head.

2. **SimCLR HF**: Contrastive model pretrained on the HUNT4 data with *frozen* weights and one *hidden* layer before the output layer.

3. **SimCLR HU**: Contrastive model pretrained on the HUNT4 data, one hidden *hidden* layer before the output layer and *unfrozen* weights for the *last* layer of the encoder.

4. **Baseline CNN**: CNN proposed by Hessen and Tessem [2016].
5. **Random Init LF**: Same architecture as (2), but with a *randomized* (no pretraining) and *frozen* encoder.
6. **Random Init LU**: Same architecture as (2), but with a *randomized* (no pretraining) and *completely unfrozen* encoder.
7. **Random Init HU**: Same architecture as (3), but with a *randomized* (no pretraining) and *completely unfrozen* encoder.

The model developed by Hessen and Tessem [2016] will serve as a baseline for comparison. Their model performed well on the TIL dataset, but it's interesting to see whether this model still performs well on a free-living dataset like HARTH. The architecture and most of the hyperparameters will stay the same for a fair comparison, but small changes will be made to the learning rate. In (2), the weights of the pre-trained encoder are frozen, and a linear trainable prediction head with ten neurons corresponding to the number of classes is attached. This model will insight into the quality of the encoder's feature representation, using only a linear layer with minimal trainable parameters for the downstream task. The primary objective of the linear layer is to distinguish and classify the clusters formed by the encoder into their respective activity categories. Because minimal help is given by the linear layer, the results from this setup are the most important and will therefore be utilized in the other experiments as well. In (3), an extra hidden layer is placed between the encoder and output prediction. This will determine the effectiveness of incorporating a trainable nonlinear MLP in improving the model's performance. In (4), the last layer of the encoder is unfrozen (every other layer stays frozen). This setup aims to investigate the impact of updating the last layer's parameters on the model's performance and will assess to what extent the encoder learns sub-optimal representations.

In the setups (5-7), no pretraining is done and the weights and biases of the encoder are randomly initialized. In (5), the encoder is frozen and only the linear prediction head is trained. This model is expected to perform poorly but will be used to assess the extent to which (2) learns any features compared to randomly initializing the encoder's weights. In (6), all the weights of the encoder are unfrozen. This model will highlight the differences in the learned representation between *contrastive self-supervised* pretraining and *purely supervised* training in terms of F1 scores of the different activities. The last model (7) is mainly there to see how well a supervised CNN largely based on the architecture of the encoder of SimCLR can perform in free-living HAR with optimized hyperparameters.

4.1.2 Experiment 2: Non-contrastive and unbiased contrastive self-supervised learning for HAR

In this experiment, different techniques to deal with the sampling bias is going to be implemented into the contrastive SSL model. The following setups will be implemented:

1. **SimCLR LF (without clustering)**: *Contrastive* model inspired by the SimCLR architecture creating negative pairs of samples randomly (same as model 2 in experiment 1) for comparison.
2. **SimCLRCluster LF (with clustering)**: *Contrastive* model inspired by the SimCLR architecture using clustering to filter out false negative pairs of samples during pretraining.
3. **SimSiam**: *Non-contrastive* model inspired by the SimSiam architecture.

The encoded vector produced by the above setups will also be analyzed using TSNe plots, which will show how well the models manage to create clusters of data with the same activity label.

4.1.3 Experiment 3: Contrastive and non-contrastive learning within a low-data regime and transfer learning setting

In the third experiment, it is desired to understand how the model performs in both different semi-supervised and transfer-learning settings. In the first case, this means limiting the amount of supervised training data that will be available for training and seeing how this affects the supervised performance of the model. To simulate different semi-supervised scenarios, we vary the percentage of labeled data available for training, ranging from only 0.05 % to 30% of the total training dataset. For each scenario, the labeled data will be randomly sampled from the original training set, ensuring a consistent distribution of classes across all conditions that resemble the full-size dataset. The contrastive model will be compared to a purely supervised model with no pretraining, with the aim of understanding how pre-training affects the model's learning efficiency when working with a limited amount of labeled data.

In addition, the model's robustness to transfer learning will be explored by pretraining the model on different datasets. This will give us some insight into how transfer learning affects the performance of the models on the labeled HARTH dataset. In this setup, only half of the subjects in the HARTH dataset will be available for finetuning, and the rest will be used for testing. The reason for this is mainly due to LOSO being too time-consuming for the number of models to be tested. Additionally, half of the subjects are chosen for testing due to the large differences in the time used to perform various activities between the participants. Hence, the somewhat unconventional division of train/test data is selected in this part of the experiment to ensure a more trustable test score.

4.2 Experimental Setup

4.2.1 Models and parameters

The baseline CNN used for the experiments has the exact same architecture as the one used by Hessen and Tessem [2016]. It consists of two convolutional (conv) layers and ends with two fully connected layers. The first 1d-conv layer consists of 40 kernels with height = 1 and length = 30. The second conv layer consists of 80 kernels with height = 1 and length = 30. The first fully connected layer consists of 1500 nodes and the output layer has ten nodes. The resulting number of trainable parameters is around 5.2e6. The ReLu activation function is used after both conv layers and a dropout with a probability of 50 % after the first fully connected layer. Softmax is used after the last layer to transform the network outputs into a probability distribution for each class. The cross-entropy loss function is used to calculate the loss and the Adam optimizer is used for backward propagation to adjust the network weights.

The chosen parameters for the contrastive (SimCLR) and non-contrastive (SimSiam) upstream models are shown in table 4.1 and 4.2, respectively. The parameters not mentioned for the SimSiam, e.g. encoder parameters, remain the same as those used in SimCLR. Note that the batch size used is 512. Although this is normally considered as a large batch size for a neural network, Chen et al. [2020] found that the SimCLR benefits from extremely large batch sizes of up to 4096 with a saturation of performance going above this, and argued that this was due to the

importance of having enough negative samples per positives. However, we found that batch sizes of above 512 for the contrastive models did not yield any improved performance. Also, Chen et al. [2020] found that a nonlinear projection head was important for the learned representations, and was also employed in our model with 128 and 50 neurons in the hidden- and output layer respectively. For SimSiam however, both layers of the projection head consist of 128 neurons due to the following prediction head of SimSiam. See figure 3.5b. This prediction head consists of 128 and 50 neurons in its hidden and output layer, respectively. The loss functions used in SimCLR and SimSiam are adam and SGD respectively, which are the loss functions used in the original papers.

Parameter	**Value**
Input size	6 x 100
Filters in conv layer 1 (conv1)	32
Filters in conv layer 2 (conv2)	64
Filters in conv layer 3 (conv3)	96
Filter size (w x h)	24 x 1
Filter size (w x h)	16 x 1
Filter size (w x h)	8 x 1
N. nodes in the first layer of the projection head	128
N. nodes in the second layer of the projection head	50
Dropout after conv1, conv2 and conv3	10 %
Activation function after conv1, conv2 and conv3	ReLu
Optimizer	Adam
Batch size	512
Starting learning rate	0.001
Lr scheduler	Cosine Decay
Augmentation methods	aug_x = Scaling, aug_y = Window slice
Loss function	NTXent loss (InfoNCE), temperature = 0.1
Total number of trainable parameters (encoder + projection head)	86 720 + 18 802 = **105 522**

Table 4.1: Hyperparameters for the contrastive (SimCLR) upstream model

The trainable MLP attached to the encoder for finetuning consists of a dropout layer with a probability of 10 % and a linear fully connected layer of 10 neurons corresponding to the 10 classes. For models 2 and 3 in 4.1.1 a hidden layer of 1024 neurons, dropout of 10 %, and ReLu before the output layer is also inserted into the downstream model. SGD is used as the optimizer for finetuning with cosine annealing as a learning rate scheduler and a starting learning rate of 0.01 ending at 0. These parameters are used for all downstream models for a fair comparison.

Parameter	Value
N. nodes in FC1 of projection head	128
N. nodes in FC2 of projection head	128
N. nodes in FC1 of prediction head	128
N. nodes in FC2 of prediction head	50
Optimizer	SGD
Batch size	256
Starting learning rate	0.01
Lr scheduler	Cosine annealing, min lr = 0.0
Loss function	Cosine similarity
Total number of trainable parameters (encoder + projection head + prediction head)	86 720 + 28 928 + 12 978 = **128 626**

Table 4.2: Hyperparameters for the non-contrastive (SimSiam) upstream model that are either different from or do not exist in the contrastive SimCLR model. The parameters of the encoder are the same as for the SimCLR in table 4.1 and are therefore omitted here.

Parameter	Value
upstream encoders	SimCLR and SimSiam
N. nodes in the hidden layer of prediction head	1024
N. nodes in the output layer (corresponding to num classes)	10
Batch size	128
Epochs	20
Loss	CrossEntropy
Starting learning rate	0.01
Lr scheduler	Cosine annealing, min lr = 0.0
Total number of trainable parameters (*excluding* the hidden layer of the prediction head)	**970**
Total number of trainable parameters (*including* hidden layer of the prediction head)	99 328 + 10 250 = **109 578**

Table 4.3: Hyperparameters for the downstream model

4.2.2 Preprocessing of data

Both the HARTH v1.2 and the HUNT4 datasets are normalized by calculating the mean and variance of both datasets. A static sliding window of 1 second is used with an overlap of 50 % between each adjacent time segment. 1 second windows are enough to represent the simple and repetitive activities we are dealing with, without feeding the model with too much information at the same time. Also, this window size has been successful in previous works [Logacjov et al., 2021; Hessen and Tessem, 2016; Skauge, 2021]. Banos et al. [2014] state that a window size of 1-2s yields the best tradeoff between speed and accuracy. A sliding window of 1s and sampling frequency of 100Hz yields a time series window of 100 samples. The labels for each window are

re-sampled resulting in one activity label for each window. In rare situations where one window consists of multiple activities, the most frequently occurring activity is chosen as the label for that window.

It is found that oversampling the minority classes is important for these classes to not be entirely ignored during training. The dataset consists of 44.94% sitting, and a naive classifier that only predicts sitting will therefore achieve 44.94% total accuracy. However, an assumption made in this thesis is that all classes are of the same importance and thus, oversampling is considered a straightforward and effective approach to prevent the classifier from disregarding minority classes.

4.2.3 Hyperparameter tuning, training- and validation method

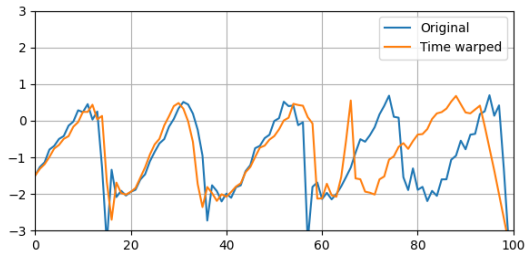
The model should not be trained and tested on the same subjects. To address this requirement, the Leave-One-Subject-Out (LOSO) approach is used. LOSO maximizes the available training data by leaving out one subject from the training set for validation until all subjects are validated. This approach ensures that the model's generalizability is assessed across various subjects and reduces the potential for subject-specific biases or overfitting. During hyperparameter optimization, the most important parameters including augmentation methods, batch size, learning rate, and size of the convolutional and fully connected layers (i.e. number of neurons and size of kernels) are prioritized through a grid-search strategy. This strategy goes through every combination of parameters in the grid and finds the optimal combination. Due to the number of parameters to be optimized and the time-consuming process of LOSO, the hyperparameter-optimization is done by a regular 80/20 split of the data. Different augmentation types have in previous works shown to yield highly different results. Because of this, augmentation methods are tuned first by employing standard values for the other important parameters that are found in the literature.

The main evaluation metric used for the models is the macro f1 score for each activity, along with an average (macro) f1-score which combines the unweighted average score for all activities. The weighted F1 score is an alternative that takes into account class imbalances. However, as argued by PlÖtz [2021], this method could potentially skew the results towards favoring the majority classes. In our experiments, we collect the predictions for all subjects in a list and calculate metrics based on this list. An alternative is to calculate metrics for each subject and take the average. This method weights each subject and predictions for their performed activities the same and there is less bias towards subjects with more recorded data. Nonetheless, it is clear from the HARTH dataset that there are subjects with minimal or even no recorded samples for specific activities. Consequently, we believe that assigning equal weight to the scores for each activity of these subjects, as compared to those who have a substantial number of recorded samples for the same activities, is an unsuitable evaluation strategy that we choose to omit here. We also examine the precision and recall derived from normalized confusion matrices, which will provide additional understanding of the areas where the model underperforms.

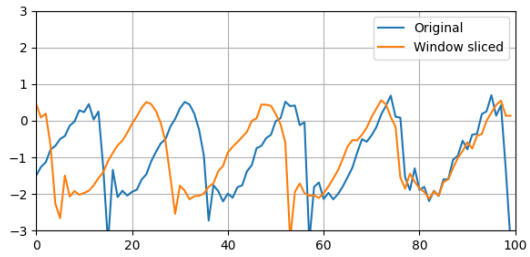
When it comes to the required duration of pretraining, our findings suggest that extending the pretraining time beyond 45 hours didn't significantly enhance the model's performance. This observation is shown in Figure A.1 in the appendix. Consequently, a pretraining duration of 45 hours has been selected for all models that are pre-trained on the HUNT4 dataset.

4.2.4 Data augmentations for the contrastive model

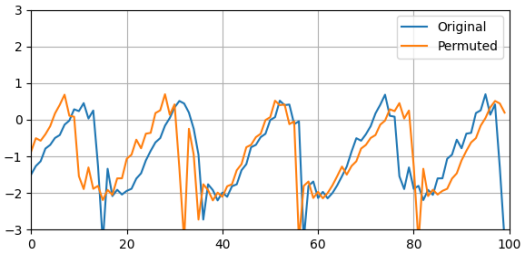
Experiments with different kinds of augmentation methods are conducted that vary in difficulty and the combination yielding the best f1-score is used for final testing. The task of performing one separate augmentation for each side of the contrastive network is experimented with. Additionally, only performing one augmentation on one side and leaving the other side untouched is also performed. Due to the large number of existing augmentation methods to choose from, a strategy of combining (1) augmentation methods that only alter the signal along the time axis (x) with (2) augmentation methods that only alter the signal in the response axis (y) is performed. This is made clear in figure 4.1 and 4.2 below which shows plots of the augmentation functions defined in 3.2.3. The plots are split into time-domain augmentations and magnitude-domain augmentations. The original time-series window is chosen randomly from one of the participants in the HARTH dataset and represents the activity *running*. Note that only the alteration of one time-series channel is visualized for simplicity.



(a) Time warp



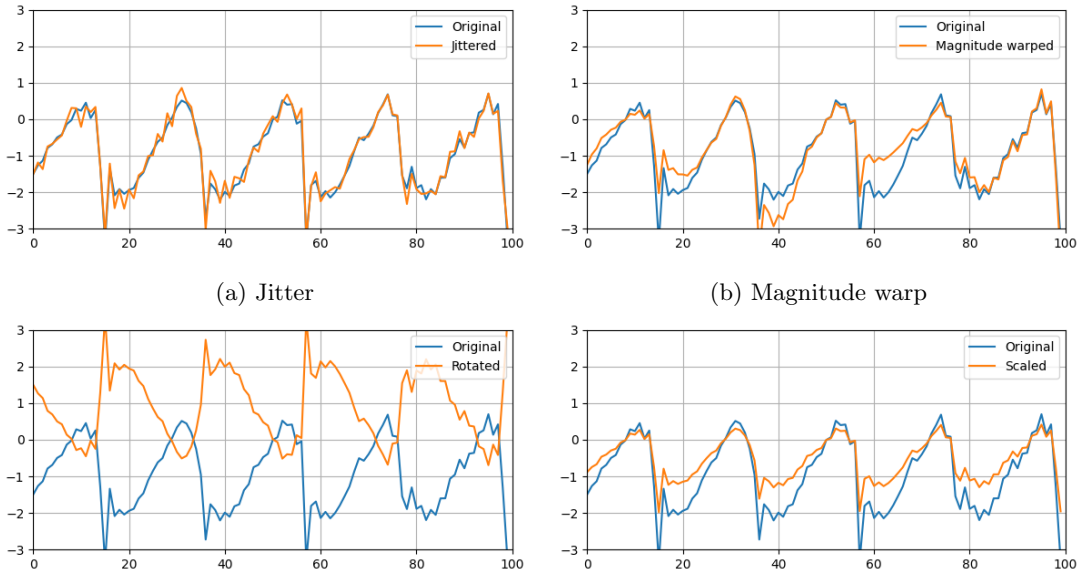
(b) Window slice



(c) Permutation

Figure 4.1: Augmentations in the time domain for one randomly sampled time-series window of the activity *running*

The downstream performances after combining different kinds of augmentations are shown in table 4.4. As seen from table the table, scaling generally performs well when combined with other augmentation functions. Combining permutation with scaling yields the best F1 score and is therefore chosen as the main augmentation during the optimization of the other parameters and the rest of the experiments. There is a gap in performance from some of the augmentations, with a gap of 17 % between the worst-performing and best-performing augmentations.



(a) Jitter (b) Magnitude warp

(c) Rotation (d) Scaling

Figure 4.2: Augmentations in the magnitude domain for one time-series window of the activity *running*

4.3 Experimental Results

The results of the experiments are presented below and are grouped corresponding to the experimental plan. The F1 score is presented for all models. Additionally, we demonstrate the models' performance in recognizing various activities in the first two experiments where we provide normalized confusion matrices, using both precision and recall. This is to get an insight into which areas the model fails to correctly classify certain activities. t-SNE plots are presented to provide insight into how effective the different models are in grouping together or repelling various instances. For the last experiment involving little labeled data and transfer learning, only the average F1 score is presented. Be aware that the provided F1 scores are derived from the aggregated predictions for all subjects, rather than the average of the individual F1 scores per subject as explained previously. For the second and third experiments, we only present the results of the linear and frozen (LF) versions of the models. The aim is to investigate the effi-

	Jitter	**Magnitude warp**	**Rotation**	**Scaling**	**None**
Window slice	51.40	59.64	45.67	60.20	57.63
Time warp	54.11	56.72	45.23	60.35	56.85
Permutation	58.69	54.42	50.82	**62.57**	56.33
None	49.50	53.93	49.84	52.09	-

Table 4.4: F1 score, in percent, of downstream classification after combining different augmentations during pre-training. Augmentations in the time domain are shown on the top rows while augmentations in the magnitude domain are shown in the leftmost column. *None* means that no augmentation is performed for one of the branches.

ciency of the pre-training process in developing meaningful features, with only marginal support from the trainable prediction head in separating the encoded samples into various activities.

4.3.1 Self-supervised contrastive learning for HAR

Table 4.5 shows the performance on various activities using different setups of the contrastive model (setup 1-3) and table 4.6 shows the performance of the purely supervised setups (4-7). As seen from the scores marked in bold, the purely supervised model achieves the best F1 score for all activities with an average F1 score of 81.10 for the *Rand. Init HU* in addition to having the smallest standard deviation of f1 scores. The Rand. Init LU beats the *SimCLR LF* with 20.99 percentage points. The pre-trained *SimCLR LF* improves over the *Rand. Init LF* with 27.38 percentage points. However, the *SimCLR LF* has a big difference in F1 scores for various activities, with a standard deviation of 27.48 %, the worst F1 score of 13.27 % in stairs (desc), and the best F1 score of 96.68 % in sitting. Adding an extra hidden layer improves the average F1 score of the model by 2.01 %, and unfreezing the last layer improves the average f1 score even further by as much as 6.64 %. Most of these improvements come from initially bad-performing activities like cycling (standing), stairs (descending), stairs (ascending), and walking with an improvement of 16.11 %, 12.5 %, 19.9 %, and 16.76 % respectively, going from *SimCLR LF* to *SimCLR HU*.

	SimCLR LF	**SimCLR HF**	**SimCLR HU**
Sitting	96.68	96.86	98.15
Walking	56.27	68.38	73.03
Standing	79.76	78.70	80.70
Cycling (sit)	76.27	75.64	80.55
Lying	84.57	87.81	92.99
Running	88.28	87.05	90.04
Shuffling	34.13	36.04	38.50
Stairs (Asc)	30.88	40.09	50.78
Stairs (Desc)	13.27	20.27	25.77
Cycling (stand)	39.18	28.60	55.29
Average	59.93 ± 27.5	61.9 ± 26.5	68.6 ± 23.4

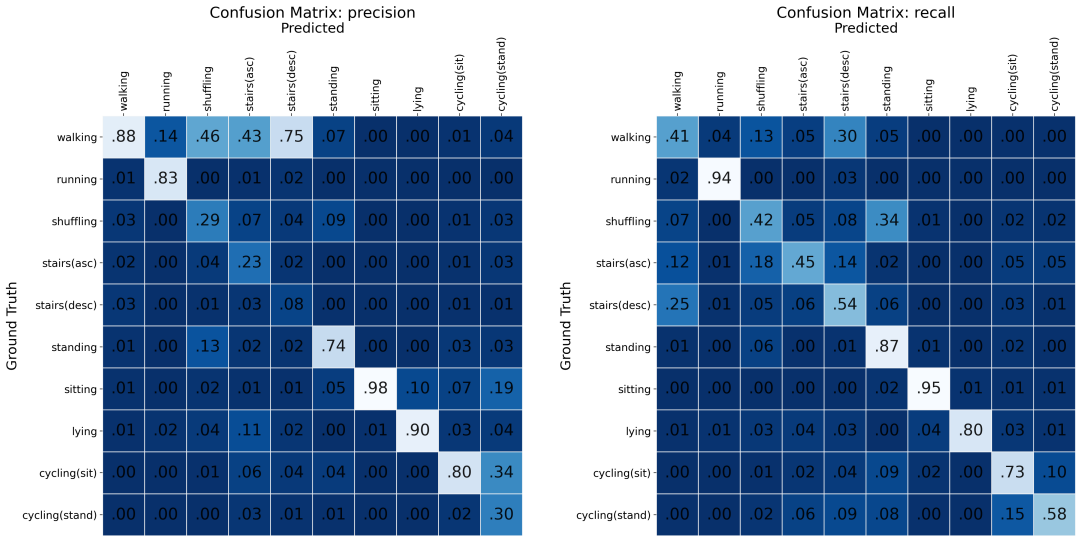
Table 4.5: F1 score, in percent, of downstream classification for the self-supervised models corresponding to configuration 1-3 in 4.1.2. The average F1 scores also show the standard deviation of the F1 score for all the activities. The rows are ordered with decreasing frequency of the activities in the HARTH dataset.

The confusion matrices for the *SimCLR LF* are depicted in figure 4.3. By analyzing the diagonal of the matrix for SimCLR, we observe that it yields a low precision rate of 8%, 23%, and 29% for activities such as descending stairs, ascending stairs, and shuffling, respectively. Furthermore, for these specific activities, the majority of the predictions (75%, 43%, and 46% respectively) actually have the ground truth label walking. The easiest activity to classify is sitting, with a precision of 98 %. The model has a better recall than precision for the aforementioned activities descending stairs, ascending stairs, and shuffling, with 54 % 45 %, and 42 % in recall respectively.

The confusion matrices of the purely supervised model are shown in figure 4.4. The worst precision is achieved for the activities shuffling, stairs(desc), and cycling (stand) with scores of 39 % 55 %, and 54 % respectively. The classifier achieves a precision of 99 % on the activities

	Baseline CNN	Rand. Init LF	Rand. init HU	Rand. Init LU
Sitting	97.91	91.27	**98.89**	**98.89**
Walking	79.72	27.62	87.50	**87.76**
Standing	84.05	54.71	84.49	**84.57**
Cycling (sit)	66.19	2.83	88.87	**90.20**
Lying	91.32	55.55	98.87	**99.23**
Running	89.68	73.19	**97.50**	97.18
Shuffling	45.59	1.02	49.95	**50.22**
Stairs (Asc)	45.33	2.27	**77.29**	73.74
Stairs (Desc)	56.94	7.62	**66.28**	66.09
Cycling (stand)	35.26	9.45	61.30	**61.34**
Average	69.20 ± 21.22	32.55 ± 31.77	**81.10 ± 16.16**	80.92 ± 16.33

Table 4.6: F1 score of downstream classification for the purely supervised models corresponding to configuration 4-7 from 4.1.1. The first column is the results from the CNN of Hessen and Tessem [2016]. Be aware that Rand. Init HU and LU have all layers unfrozen, while SimCLR HU only has the last layer unfrozen.

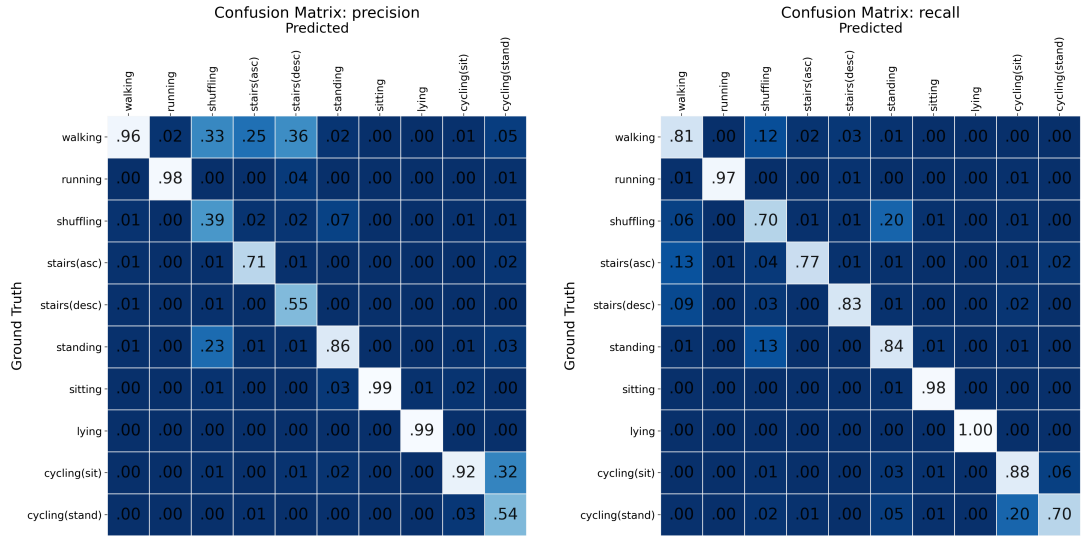


(a) *SimCLR LF* normalized along the columns (precision) (b) *SimCLR LF* normalized along the rows (recall)

Figure 4.3: Confusion matrices of the *SimCLR LF*

sitting and lying and a recall of 98 and > 99 % on the same activities. The recall is also better than the precision for the shuffling, stairs(desc), and cycling (stand) with a score of 70 %, 77 %, and 70 % respectively. 36 %, 25 % and 33 % of the activities predicted as stairs(desc), stairs(asc) and shuffling respectively, are actually walking. 23 % of the activities predicted as shuffling are actually standing and 32 % of the activities predicted as cycle (sit) are cycle (stand).

In Figure 4.5, we observe the t-SNE plots representing the encoded vectors obtained from a pre-trained SimCLR model without any fine-tuning. Notably, the activity "running" demonstrates a highly clustered pattern. Similarly, the activities "lying," "sitting," and "standing" also



(a) *Rand Init LU* normalized along the columns (precision) (b) *Rand Init LU* normalized along the rows (recall)

Figure 4.4: Confusion matrices of the *Rand Init LU*

show noticeable clustering. Interestingly, the "cycling (sit)" activity appears to be divided into two distinct and well-separated clusters.

4.3.2 Non-contrastive and unbiased contrastive SSL for HAR

In table 4.7 are the results for the experiments with the non-contrastive *SimSiam LF* and the contrastive *SimCLRCluster LF*. Additionally, the *Rand Init LF* and *SimCLR LF* (w/o clustering) from the previous experiments are shown in the two leftmost columns for comparison. Using clustering for the SimCLR seems to have very small effects on the performance. If anything, clustering for false negative detection worsens the performance by 0.43 percentage points. When we add clustering, the F1 scores for certain activities remain relatively unchanged. However, for activities like 'Cycling (sit)' and 'Running,' we see a significant drop of 7.2 and 9.02 percentage points respectively. Conversely, the F1 scores for 'Lying', 'Shuffling', and 'Cycling (stand)' increase by 9.55, 3.29, and 2.98 percentage points, respectively. The *SimSiam LF* perform the worst out of all the implemented models, with an average F1 score of 51.25 %. The model only achieves the highest score in two activities, namely sitting and stairs (descending) with a score of 97.84 % and 19.84 %, respectively.

The confusion matrices for *SimSiam LF* and *SimCLRCluster LF* are shown in figure 4.6 and 4.7. The precision is, just like for the simCLR, worst for stairs (descending), stairs (ascending), and shuffling. However, this phenomenon is even more extreme for *SimSiam LF*, with 76 %, 82 %, and 65 % of these predictions having the true label walking. The worst precision is seen in the activity stairs (ascending) where only 8 % of the predicted samples are actually stairs (ascending). The recall is generally better, with the worst score being 42 % for walking and stairs (ascending) and the best score being 94 % for sitting. The most extreme misclassification, however, is that 89 % of the samples that have the true label shuffling are classified as standing

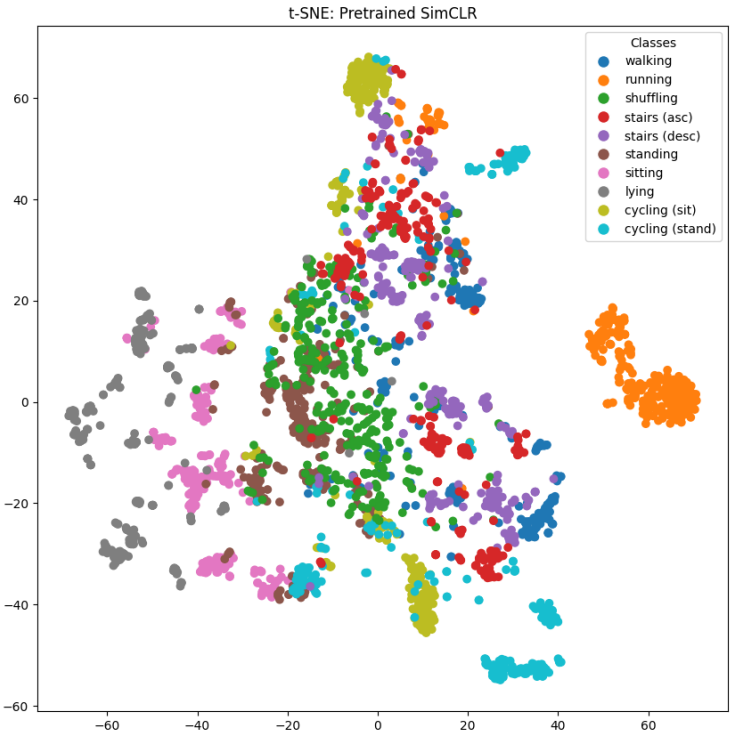


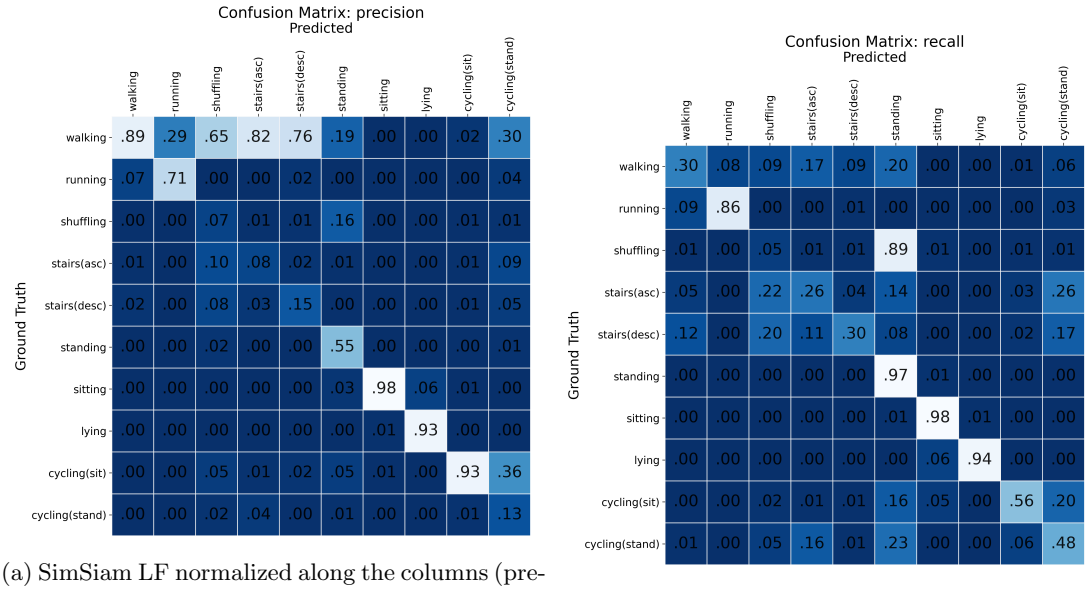
Figure 4.5: t-SNE plot for a pre-trained SimCLR without any finetuning

	Random init	**SimCLR LF w/o clustering**	**SimCLR LF w/ clustering**	**SimSiam LF**
Sitting	91.27	96.68	96.44	**97.84**
Walking	27.62	56.27	**56.59**	44.83
Standing	54.71	**79.76**	78.27	70.35
Cycling (sit)	2.83	**76.27**	69.07	69.99
Lying	55.55	84.57	**94.12**	93.53
Running	73.19	**88.28**	79.26	77.63
Shuffling	1.02	34.13	**37.42**	5.88
Stairs (Asc)	2.27	**30.88**	28.63	12.50
Stairs (Desc)	7.62	**13.27**	13.25	**19.84**
Cycling (stand)	9.45	39.18	**42.16**	20.16
Average	32.55 ± 30.29	**59.93 ± 26.20**	59.52 ± 25.74	51.25 ± 31.51

Table 4.7: Average F1-scores comparing the randomly initialized and contrastive model from experiment 1, with a modification of the Contrastive model and the *SimSiam LF*.

by the *SimSiam LF*.

The confusion matrices for the *SimCLRCluster LF* have a similar structure as the matrices for *SimCLR LF*. However, one notable difference is in precision, where 19 % and 11 % of the samples predicted as stairs (ascending) and stairs (descending) respectively, have true label cycling (sit).



(a) SimSiam LF normalized along the columns (precision)

(b) SimSiam LF normalized along the rows (recall)

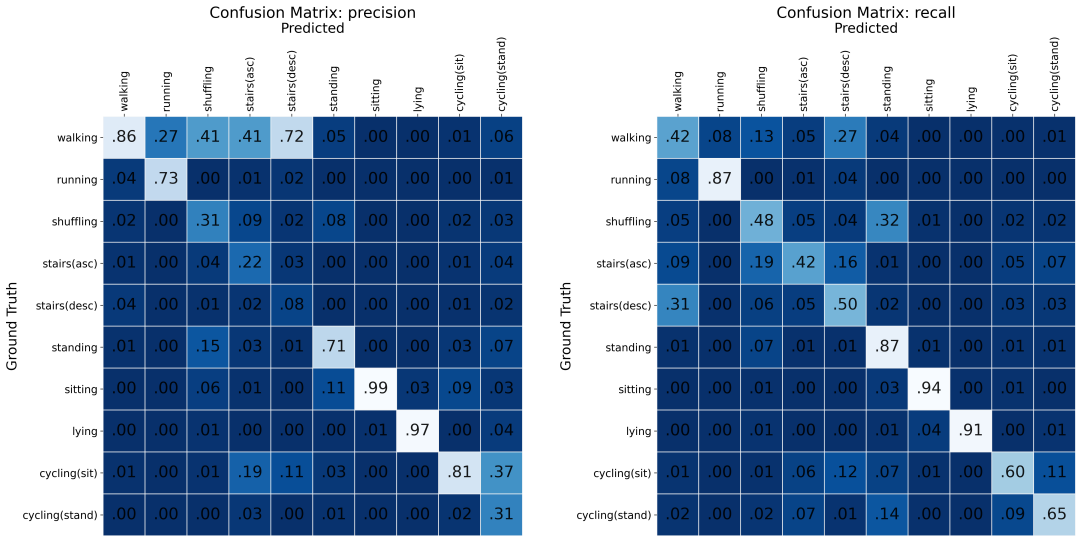
Figure 4.6: Confusion matrices of the *SimSiam LF*

Figure 4.8 showcases the t-SNE plots for the *SimSiam LF* and the *SimCLRCluster LF*. Looking at the plot, *SimSiam LF* creates more dense clusters compared to *SimCLRCluster LF* for activities like sitting, standing, and shuffling. In contrast, the samples in *SimCLRCluster LF* display a more scattered and less organized pattern. Both models effectively cluster samples labeled as "running," although there are some outliers present in *SimCLRCluster LF*. Furthermore, both models show the tendency to create two distinct clusters for the "cycling, sit" activity, with a significant distance separating them.

4.3.3 Contrastive learning within a low-data regime and transfer learning setting

The F1 scores obtained from pretraining the models on different kinds of datasets are shown in table 4.8. As seen from the results, SimCLR benefits mostly when pre-trained on the HUNT4 (an increase of 2.07 percentage points from HARTH %) data, and SimCLRCluster benefits most in pretraining on the HARTH dataset. The amount of HUNT4 and HARTH data used in pretraining are the exact same, for a fair comparison. UCI-HAR however, is a smaller dataset. Ideally, the number of samples is the same for all datasets. However, some consideration is taken by increasing the number of epochs so that the number of training iterations is similar. Pretraining on the UCI HAR dataset yields the worst performance, with 52.26 %, 43.17 %, and 26.01 % for the SimCLR, *SimSiam LF*, and SimCLRCluster, respectively.

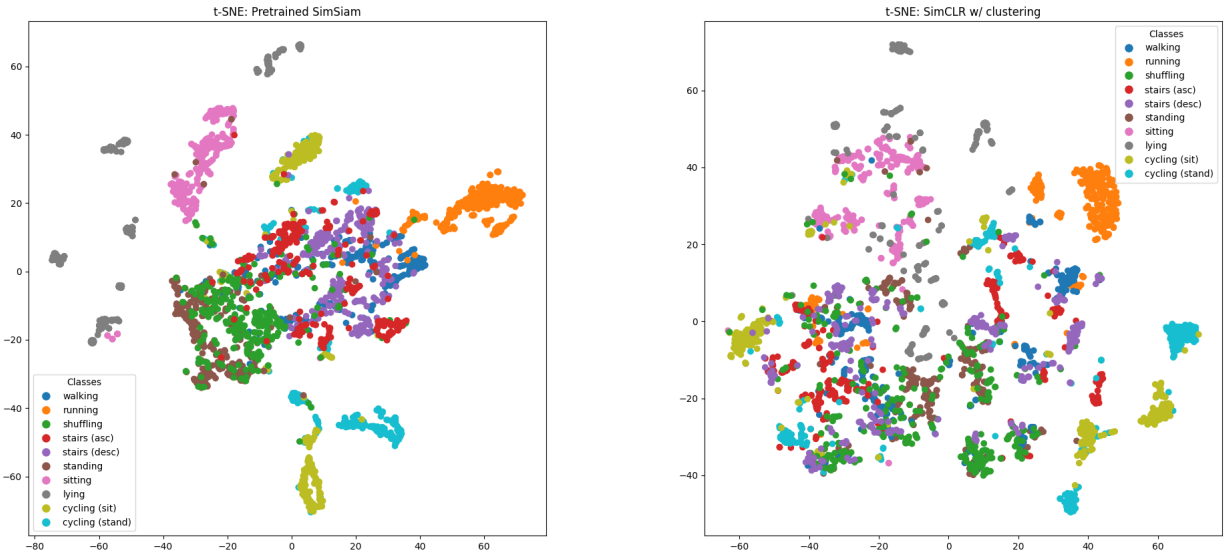
In figure 4.9 we see the F1 score of various models when pre-trained on the HUNT4 data



(a) *SimCLR LF* with clustering normalized along the columns (precision)

(b) *SimCLR LF* with clustering normalized along the rows (recall)

Figure 4.7: Confusion matrices of the SimCLR w/clustering



(a) Pretrained *SimSiam LF* without finetuning

(b) pretrained SimCLR w/ clustering and without finetuning

Figure 4.8: t-SNE plots

	HARTH → HARTH	HUNT4 → HARTH	UCI HAR → HARTH
SimCLR	55.69	57.76	52.26
SimSiam	47.56	47.30	43.17
SimCLRCluster	55.74	50.97	26.01

Table 4.8: F1-score, in percent, when pretraining on different kinds of datasets, freezing the encoder, and then fine-tuning and testing the models on the labeled HARTH dataset

and finetuned on an increasing amount of labeled data. The chosen percentages of the original HARTH dataset were chosen as 0.05 %, 1 %, 10 %, and 30 %, which were sampled randomly. Looking at the plots we see that the pre-trained SimCLR only outperforms the Rand. Init LF with 3.41 % when less than 0.05 % of labeled data is available. At just 1% of labeled data available, Rand. Init LU outperforms the pre-trained *SimCLR LF*, SimCLRCluster LF, and *SimSiam LF* by 10.85, 17.52, and 28.38 percentage points, respectively. The disparity becomes even more significant as the amount of labeled data increases.

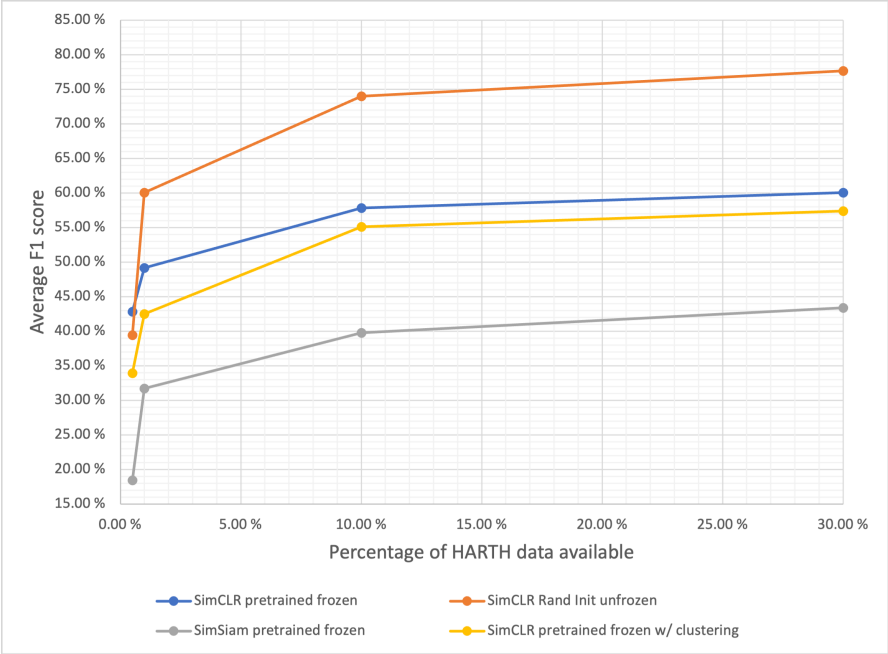


Figure 4.9: Plot of the F1 score after training the models using different amounts of HARTH data

Chapter 5

Discussion

The findings from section 4.3.1 reveal noticeable deviations in performance both across different setups and for various activities. Certain activities are straightforward to categorize, even for the most ineffective models, whereas some prove significantly more challenging. The performance disparities between the contrastive and non-contrastive methods become apparent despite their similar architectures. The modified SimCLR's performance was underwhelming, and we aim to provide some explanations for this outcome. Ultimately, we will offer explanations for the outcomes observed in the concluding experiments concerning transfer learning and the acquisition of knowledge from a limited quantity of labeled data. In essence, we will provide some reasoning behind the key findings while acknowledging various limitations associated with the executed experiments. The discussion consists of one section for each experiment. Each section will be served with a short paragraph containing the main points discussed.

5.1 Self-supervised contrastive learning for HAR

The model initialized with random weights and a frozen encoder was expected to perform badly and was mainly created for comparison. However, the significant improvement of roughly 27 percentage points in the F1 score, when these weights undergo pretraining, certainly indicates that the model can learn useful features for the downstream task. Most notable is the increase in cycling (sit), going from below 3 % to above 76 %. There is a general increase of above 25 % F1 score for most activities except those where the randomly initialized model already somehow manages to get a decent score, like sitting, confirming that the model learns valuable features that benefit all activities, and not just a small subset. The contrastive model struggles with creating valuable features for stairs (desc). Firstly, this activity is the most challenging to classify even for the most effective fully supervised model. It suggests this activity remains difficult even when labels are available and the dataset has been oversampled. Furthermore, looking at the class distribution in 3.1, this particular activity has the lowest frequency within the HARTH dataset, suggesting a similar distribution in the HUNT4 dataset. Since oversampling is not possible when labels are absent, only a restricted amount of samples representing this activity can be presented to the model during the pre-training phase, compared to other activities. This is a problem in general when pre-training on a free-living dataset versus in-lab. Also, pretraining on more free-living data to increase the number of samples originating from minority activities does not increase the performance of the contrastive model as seen from Figure A.1. Adding a hidden layer to the SimCLR during supervised training does increase the average F1 score by 2 %. However, unfreezing the last layer of the SimCLR show a notable increase in performance of

almost 7 %, indicating that unfreezing pre-trained weights can be beneficial during fine-tuning. The increase is most notable for less frequent activities like stairs-walking and cycling (stand).

The SimCLR model with entirely unfrozen and randomly initialized weights delivers superior performance for most activities, except for shuffling and cycling (stand). It was anticipated that this model would outperform a SimCLR model with pre-trained and frozen weights, as the amount of trainable parameters in the downstream *SimCLR LF* model (not including the frozen parameters) is 970 versus 87 690 (encoder + projection head) for the *Rand Init LU*. This was also reported in the original SimCLR paper Chen et al. [2020] where the supervised baseline (Resnet-50) outperform the SimCLR using ResNet-50 as an encoder with frozen weights for finetuning with around 7 %. However, the difference is much more substantial in our case, where a 21 % improvement from the frozen *SimCLR LF* to the randomly initialized SimCLR LU implies that the encoder of the SimCLR does not learn optimal features during the self-supervised pretraining stage. The research presented in Saeed et al. [2019], which has a similar setup to ours, demonstrated that a purely supervised model also often performs better than a self-supervised frozen model (SCN) of the same design for most datasets. The performance gap typically falls under 5%, with the largest difference observed at 12 % between the supervised and self-supervised models. However, it's worth noting that the datasets used in Saeed et al. [2019] for Human Activity Recognition (HAR) were collected in a lab setting, and there are some variances in what activities that are labeled. Additionally, they report a score of over 60% for most datasets, even when the encoder weights were randomly initialized and frozen, which can suggest that their data are easier to separate using just a linear classifier on top. This contrasts with our findings, where we achieved an average F1-score of just 32.55 with randomly initialized weights. Saeed et al. [2021] also share similar problems of separating stairs ascending and descending from the activity walking during the pretraining phase. The state-of-the-art CPC from Haresamudram et al. [2021] also yields similar results regarding the performance of a pre-trained and frozen encoder vs a randomly initialized and unfrozen encoder trained end-to-end on benchmark HAR datasets. Harish et al. [2020] however, find that using their masked reconstruction method for pretraining actually surpasses the F1-score of the supervised version on 2 out of 4 benchmark HAR datasets recorded in-lab. This is also found in Jain et al. [2022], where their pre-trained model with a frozen encoder actually outperforms the unfrozen and purely supervised model.

The pre-trained SimCLR model with unfrozen weights for the final layer (SimCLR HU) exhibits comparable performance to the convolutional neural network proposed by Hessen and Tessem [2016], despite having fewer parameters. Moreover, the results reported by Hessen and Tessem [2016] exhibit a notable decline of around 20 % when compared to the results obtained by retraining and evaluating a model built on their architecture in a real-world environment like HARTH. Generally, activities with the worst F1 score are the least frequent activities in the HARTH dataset. Looking at the bottom 4 rows of Table 4.5, these activities include shuffling, stairs (ascending), stairs (descending), and cycling (stand). It can also be argued that these activities involve the most complex movements resulting in a rather chaotic sensor response, with examples shown in figure C.1 in the appendix.

The confusion matrices for *SimCLR LF* depicted in figure 4.3 explicitly highlight the areas where the model encounters difficulties. Figure 4.3a shows the confusion matrix normalized along the columns with the precision along the diagonal, proving that most of the samples predicted as stairs(asc) are actually walking. The same goes for stairs(asc) and shuffling. When observing the activity plots in Figure C.1 found in the appendix, it becomes clear that certain activities share similar signal traits with the 'walking' activity. Moreover, oversampling in the downstream

dataset may lead to a classifier that becomes overly sensitive to rare activities during fine-tuning. It's noteworthy that a significant fraction of activities labeled as cycling(stand) have cycling(sit) as the ground truth. This can be attributed to the fact that both cycling(stand) and cycling(sit) activities incorporate comparable pedal and side movements. Furthermore, cycling(stand) occurs infrequently and is subject to high oversampling. The second confusion matrix in Figure 4.3a demonstrates similar challenges. However, it is obvious that the recall for most activities is higher. Especially infrequent activities from the HARTH dataset achieve a significantly better recall than precision. The aforementioned tendency of classifying walking as other similar activities is reflected in the reduced precision for the activity walking.

Activities such as walking, sitting, and running obtain both reasonable good recall and precision. These are activities with more distinct definitions and activity profiles. Comparing the confusion matrices of the *SimCLR LF* with the *Rand Init LU* there is a large upgrade in both precision and recall for all activities except in recall for 'standing'. The ambiguity that the *Rand Init LU* model experiences between walking and similar activities still persists, but at a significantly reduced level. Especially shuffling, where 33 % of its classifications is walking and 23 % is standing. This misclassification is to be expected, considering that the activity certainly sounds like a combination of both, given the vague definition *"movement not as part of walking bout. Without being able to see the feet, if the movement of the upper body and surroundings indicate non-directional feet movement, shuffling can be inferred"* [Logacjov et al., 2021]. The similarity between the activities "shuffling", "walking" and "standing" can also be seen by comparing the plots of the shuffling and walking segment in Figure C.1. The signal response for these two is almost identical, although there might be some additional small fluctuations in the signal response of shuffling. Combining "shuffling" into "standing" is not unreasonable, given this definition and an analysis of the signal responses for these activities, either through post- or preprocessing.

Figure 4.5 demonstrates the clustering of samples belonging to the same activities and the separation from other activities. Certain activities are more easily clustered together than others. For example, running and standing are easily clustered together while stairs (asc and desc) and shuffling are much more scattered around. An interesting observation involves cycling (sit), where two distinct clusters are positioned at a considerable distance from each other. A plausible explanation is that cycling (sit) actually comprises the sub-activities cycling (sit, inactive) and cycling (sit, active) as previously mentioned, in which one activity involves pedal movements and the other does not. This was done due to the extremely low frequency of either of these classes. In retrospect, the model could potentially have improved its performance if these activities were combined after the model was trained and the predictions were made, a step that would occur during the post-processing stage.

It's important to mention that activities that are more similar to running usually appear further to the right on the graph. On the other hand, activities that involve no movement, like lying down, are typically placed on the left side of the plot. Activities like standing and walking are typically found around the center of the plot. This is a logical outcome considering the foundational principle of SimCLR, which is based on a similarity measure. In SimCLR activities that exhibit high similarities tend to attract one another within the latent space, while activities that are dissimilar repel each other, creating a contrasting effect.

To summarize, pretraining the SimCLR manages to create some valuable features among all activities for the downstream performance compared to just initializing the weights randomly and freezing them. Considering that only 970 out of 87 690 parameters of the downstream model are

adjustable, the performance is impressive. Adding an extra hidden layer and unfreezing the last layer is beneficial for the performance of the model and doing this achieves a highly comparable performance to the CNN in Hessen and Tessem [2016] despite still having much fewer parameters. The model manages to cluster certain activities well as seen from the TSNe plots which are reflected in the table for the F1 scores. Interestingly, the model clusters samples according to the similarity of their activity profiles. Unfortunately, the model comes short when compared to a purely supervised model with all parameters adjustable, and additional pretraining is not beneficial. This is especially the case for minority activities and activities that are of similar nature. In retrospect, some activities could have been combined during postprocessing.

5.2 Non-contrastive and unbiased contrastive SSL for HAR

In the second experiment, it is clear that both the non-contrastive SimSiam and the SimCLR-Cluster perform worse compared to the contrastive SimCLR model. Particularly, the SimSiam achieves an average F1 score of 51.25 %, which is approximately 9 % lower than the contrastive model without clustering. This outcome is somewhat surprising since the SimSiam does not rely on negatives and thus avoids the challenge of sampling false negatives in a highly skewed dataset. Notably, there is a substantial difference in F1 scores, especially for the activities cycling (stand), stairs (asc), and shuffling. Additionally, the t-SNE plot in Figure 4.8a demonstrates that the SimSiam achieves better clustering for certain activities than SimCLR, including those just mentioned. The bigger spread in samples in general for the SimCLR is to be expected given that SimCLR runs the risk of contrasting false negatives as mentioned in 2.1.4, resulting in a phenomenon that Wang et al. [2021] term as over-clustering, implying that the model is unable to effectively extract features from an overly abundant number of negative sample pairs. This leads to the model mistakenly dividing samples from the same actual class into small separate clusters with size 1 in the most extreme cases. However, analyzing the t-SNE plot of the SimSiam-encoded samples, it seems likely that the model could be experiencing the inverse effect, namely under-clustering. Under-clustering often arises due to a lack of negative samples, especially in cases where it's inherent in the algorithm design such as BYOL Grill et al., 2020 and SimSiam [Chen and He, 2021] which are purely based on similarities between positives. This can cause different object categories to overlap. Under-clustering effectively reduces the model's learning efficiency, as it reduces the model's ability to efficiently identify the dissimilarities between samples from different activities. In general, the importance of including negatives for self-supervised HAR is also supported by Rahimi et al. [2021].

Combining SimCLR with clustering for false negative detection is not beneficial for training, at least not with the clustering algorithms used in this thesis, K-means and BIRCH. The intention was to mitigate the contrasting of false negatives and subsequently reduce the distance between samples within a cluster representing a specific activity. The t-SNE plot in Figure 4.8b show a more chaotic structure compared to the t-SNE plots shown in Figures 4.5 and 4.8a although the model still manages to cluster activities like running correctly. Although BIRCH and k-means are widely used, they have like other clustering algorithms some limitations. Firstly, the output of the encoder is vectors with a length of 96 when flattened. This introduces what is commonly termed "the curse of dimensionality", in which each cluster is equally spread out over "less equally important" dimensions. Both algorithms use the Euclidian distance measure to create clusters, where distance in all 96 dimensions is treated as equally important. And since sensor data consist of a lot of noise and can be hard to encode, this will seriously affect the performance of the clustering. An additional limitation is that the initial batches processed by

the model are clustered based on vector encodings generated by a model that has not really been trained yet. Consequently, the clustering algorithm might identify clusters inaccurately and filter out useful true negatives. Lastly, Robinson et al. [2020] argues that hard negatives, i.e. negatives that lie close to the anchor, are the most informative examples for the model. Creating n clusters corresponding to n activities increases the risk of filtering out these hard negatives because they are encoded close to the anchor.

To summarize, the SimSiam seems to cluster activities well and performs much better than a model with random and frozen weights. However, the model's performance falls short compared to the SimCLR, particularly in the tasks where SimCLR already has difficulties in accurate classification. Its shortcomings can be viewed as an amplified reflection of SimCLR's limitations. This underscores the significance of incorporating negatives (contrastive learning) for self-supervised Human Activity Recognition (HAR) in a free-living context. Applying either BIRCH or K-means clustering for false negative detection failed to boost the downstream performance of the SimCLR. Looking back, this could be both due to poor handling of large dimensions, or because the clustering algorithm still tries to group latent vectors coming from an initially untrained model, and exclude these vectors from the InfoNCE loss even if these vectors might be true negatives.

5.3 Transfer learning and effectiveness in a low-data regime

For the third and last experiment, one of the goals was to simulate training environments where little labeled data is available. As seen from Figure 4.9, the *SimCLR LF* surpasses the purely supervised *Rand Init LU* approach only in scenarios where a minimal amount of labeled data is available (less than 1%). 1 % of the data amounts to 919 samples in which the least occurring activity, cycling(stand) is only observed 9 times. The anticipation was that the *SimCLR LF* would demonstrate improved performance in a low data regime. However, it was unexpected that the *Rand Init LU* model would outperform the pre-trained *SimCLR LF* already at 1 % of available labeled data. The average F1 score already begins to stabilize at 10 % of labeled data for all models with a modest improvement of 2-3 % in F1 score going from 10 % labeled data to 30 %. This suggests that only a subset of the HARTH dataset is enough to learn optimal features, even when training an unfrozen model without any pretraining.

The other goal for the last experiment was to simulate an environment where a pre-trained model is transferred to another environment (transfer learning). This was done by pretraining the model on one dataset, and doing downstream supervised training on another. This was done on three datasets, namely HUNT4, HARTH, and UCI-HAR where the last dataset simulate the biggest change in environment due to the difference in sensor types and positions. Minor variations in performance are observed when using the HARTH and HUNT4 datasets for pretraining. On one side, the HUNT4 dataset has a strong resemblance with the HARTH dataset, hence it's logical that pretraining on either dataset would lead to comparable outcomes. However, using the HARTH dataset for pretraining involves the same data for subsequent supervised fine-tuning. This finding highlights that pretraining and fine-tuning on identical data can still be beneficial. This could be attributed to the difference in how the model learns (similarity of samples vs labels) and the fact that data undergo augmentation, thereby creating new pseudo-data samples as a by-product.

The worst downstream performance is obtained when pretraining using the UCI-HAR dataset, which was expected. However, several factors might account for this outcome: Firstly, the UCI-

HAR dataset consists of fewer samples compared to what is derived from HUNT4 and HARTH for pretraining. Consequently, even though the count of training iterations remains constant by increasing the number of epochs, the reduced data volume results in less robustness to variations in unseen data. Secondly, the UCI-HAR dataset was recorded in-lab and does not catch the diverse variations associated with activities carried out in free-living environments. Finally, certain activities present in the HARTH dataset are absent from the UCI-HAR dataset, which might complicate the task of the pre-trained encoder in generating meaningful features for these specific samples during the downstream task. Additionally, the UCI-HAR dataset was recorded using an accelerometer and gyroscope whereas both HUNT4 and HARTH consist of recordings from two accelerometers. Yet, the SimCLR and SimSiam models are able to extract some relevant features that can be shared between the two datasets. However, the SimCLRCluster model does not perform well and fails to create useful features when it's pre-trained on the UCI-HAR dataset.

To summarize, both the SimCLR and SimSiam are able to extract some valuable features, even if the model is pre-trained on a different type of dataset (UCI-HAR) containing other types of sensors and sensor positions with some decrease in performance. SimCLRCluster on the other hand does no better than a model with random and frozen weights. The SimCLR, with its pre-trained and frozen weights, is the only one that surpasses the performance of the purely supervised model with unfrozen weights when less than 0.05 percent of labeled data is used. This implies that the supervised model does not require a substantial amount of HARTH data to enhance its performance rapidly.

Chapter 6

Conclusion and Future Work

The primary objective of this chapter is to provide direct answers to the main research questions and associated sub-questions. Subsequently, we will offer recommendations for possible enhancements in the future and suggest potential steps that can be taken to advance the field of self-supervised Human Activity Recognition (HAR) using free-living data.

6.1 Conclusion

We begin concluding by answering the first research question:

Research question 1 *What are the latest advances and key characteristics of state-of-the-art supervised- and self-supervised HAR systems?*

In section 2.2 we presented work related to both self-supervised- and supervised HAR. Hammerla et al. [2016] found that supervised approaches such as CNNs are more suited than LSTMs for longer, repetitive activities such as the ones we are dealing with in this thesis. The combination of both done in Roggen et al. [2010b] can achieve state-of-the-art results, which is why their model frequently has been used as one of the supervised baselines in other recent studies for HAR. Multiple self-supervised methods that have shown great success in other domains like vision, are also successful in HAR. This includes CPC [Haresamudram et al., 2021], multi-task learning [Saeed et al., 2021] and masked reconstruction [Harish et al., 2020]. Our decision to implement a variant of SimCLR for human activity recognition was inspired by the state-of-the-art results obtained by Chen et al. [2020], as well as the efficacy of using CNNs for our specific task. This decision was made despite the original development of SimCLR targeting vision applications rather than time-series data. Also, the finding that sampling bias can harm the performance of contrastive methods motivated us to both 1) try to fix the problem through clustering and 2) completely avoid it through a non-contrastive method. Among non-contrastive methods, SimSiam was found to yield state-of-the-art results within vision while having a similar architecture as the SimCLR based on siamese networks, making it a suitable method for comparison as well.

For the second research question, we begin by answering its associated sub-questions, before we generalize by answering the main research question:

Subquestion 2.1 *What is the comparative performance of the selected self-supervised methods, and how well do they perform against purely supervised models?*

SimCLR, which utilizes contrastive learning, generally yields the best performance across most activities, while SimSiam, which employs non-contrastive learning, demonstrates poor performance in comparison. False negative detection through clustering did not enhance the performance of the SimCLR using either BIRCH or k-means clustering. The fully supervised versions perform much better than all of the self-supervised methods by a great margin. Distinguishing between stairs ascending, stairs descending and walking is the biggest challenge for all models. For all supervised and self-supervised setups, the minority activities are the hardest to classify even when oversampling is performed.

Subquestion 2.2 *How robust are the selected self-supervised methods in scenarios with limited labeled data, and to what extent do they demonstrate successful transfer learning between different types of HAR contexts?*

When dealing with limited label availability, the top-performing model, SimCLR, only surpasses the supervised model when less than 0.05 % of labels are available. With less than 0.05 % of labels, SimSiam, which generally underperforms, manages to secure less than a 20 % F1 score, further illustrating its struggles in limited data scenarios. Regarding the transfer learning capabilities, both the SimCLR and SimSiam are still able to extract some meaningful features on the different dataset UCI HAR. SimCLRCluster on the other hand does not perform better than a randomly initialized and frozen encoder (Rand init LF).

Subquestion 2.3 *How well do the experimental results align with the findings reported in the literature study?*

In most studies, training an encoder end-to-end without any pretraining performs better than a trained and frozen encoder, although on a much smaller scale than in our case [Chen et al., 2020; Saeed et al., 2019, 2021; Chen and He, 2021; Haresamudram et al., 2021]. However, some studies [Harish et al., 2020; Jain et al., 2022] even report that using a frozen and pre-trained encoder performs even better than the purely supervised baseline. This does not align with our findings, where the self-supervised model performs poorly compared to the supervised model of the same architecture (corresponding to Rand Init LF in the experimental plan). Despite being a somewhat unexpected outcome as discussed in 5.2, the discovery that SimCLR considerably outperforms the SimSiam in performance aligns with the research by Rahimi et al. [2021], which credits the inclusion of negatives as the reason for the improved performance. Similar to our findings, other research in self-supervised HAR and HAR in general, find the separation of stairs ascending, stairs descending, and walking to be the hardest activities to distinguish between [Saeed et al., 2021; Logacjov et al., 2021].

Research question 2 *Can existing contrastive and non-contrastive self-supervised methods like SimCLR and SimSiam be exploited or further developed for human activity recognition on free-living data like HUNT4 and HARTH?*

Both the SimCLR and SimSiam can be utilized for free-living time-series data like HUNT4 and HARTH. However, the results indicate that none of these models manages to learn optimal representations when comparing them to the purely supervised models. This is especially true for the non-contrastive method indicating that including negatives in pretraining can be important. Also, the performance of the SimCLR did not improve when modified by using clustering for false negative detection.

6.2 Contributions

In this study, we focused on two prominent self-supervised learning techniques for vision, namely SimCLR, and SimSiam, and adjusted them for human activity recognition on free-living time-series data from two accelerometers like HUNT4 and HARTH which have, to our knowledge, not previously been done before. These specific methods were selected based on their demonstrated success, the fact that they represent two different learning paradigms - contrastive and non-contrastive learning, and the fact that both approaches utilize similar siamese architectures, thereby making them suitable for comparison. For our specific problem, we optimized the model parameters, identified the optimal combination of time series augmentation methods for time series-based human activity recognition, and determined the necessary amount of pretraining for our model. Our findings revealed that SimCLR outperforms SimSiam significantly, a result we attribute to the importance of including negatives in the loss function. Furthermore, we investigated a novel technique of clustering the latent output vectors from SimCLR with the aim of reducing false negatives but observed that this approach did not enhance SimCLR's performance.

We discovered that self-supervised methods were successful in identifying useful features for free-living human activity recognition. However, supervised methods still displayed superior performance compared to self-supervised ones, more so than indicated in previous studies that are not based on free-living HAR. We observed that both SimCLR and SimSiam were capable of extracting meaningful features when pre-trained on a distinct human activity recognition dataset, UCI-HAR, which employs different sensors and sensor placements. We also assessed the performance of various methods in situations with limited labeled data availability. Our results indicated that, for the HARTH dataset, even a very small fraction of labeled data was sufficient for the supervised methods to outperform both contrastive and non-contrastive self-supervised methods.

6.3 Future Work

This section is split into three parts. The first part discusses potential improvements that can be made for applied methods. The second part discusses improvements that can be made regarding input data and its preprocessing. The suggestions in these parts are either practical solutions by focusing specifically on limitations pointed out in the previous chapter, or just further extensions that can be beneficial for the performance of the models. In the last part, we come up with three additional suggestions for other interesting applications that can be explored within supervised- and self-supervised HAR from sensor data streams.

6.3.1 Model improvements

Firstly, there exist numerous other self-supervised methods that are worth exploring other than the two chosen for this thesis. Some of these can potentially prove to be more fitting to free-living self-supervised HAR given the underwhelming results of SimCLR and SimSiam compared to the purely supervised models. Suggestions that are worth exploring are the previous state-of-the-art methods BYOL [Grill et al., 2020], SwAV [Caron et al., 2020], MoCo He et al. [2020], and CPC [Robinson et al., 2020]. The relevance of considering these methods lies in their varied adaptations of the Siamese architectures and different strategies for addressing collapsed solutions compared to SimCLR and SimSiam, thereby potentially providing new insights and solutions for free-living HAR. MoCo computes the loss by evaluating the similarity between both positive

and negative pairs, similar to the SimCLR method. Conversely, BYOL and SWAV calculate the loss considering only the similarity between positive pairs, as with SimSiam. By exploring these techniques, we expect to broaden the understanding and application of SSL techniques in the field of free-living HAR, with the potential of enhancing the performance of self-supervised free-living HAR.

In the pursuit of improved model performance, this thesis employed clustering to filter out false negatives, though this approach did not yield enhanced results. We, therefore, propose two alternative techniques for handling false negatives that can be explored for further research. One promising method is a teacher-student-inspired technique. This approach begins with training a "teacher" model in a supervised way. The motivation for using this method is that the "teacher" can assist a "student" model by applying pseudo-labels to batches from the unlabeled dataset, thus strategically selecting negatives with different pseudo-labels. The second suggestion is the use of a debiased contrastive objective, as developed in Chuang et al. [2020]. This method can correct the sampling bias of negative samples, even in the absence of access to labels. This makes it a potentially valuable tool, especially considering its implementation involves just a few lines of code. While the authors of the referenced study assume a class distribution close to uniform, they demonstrated that the debiased objective still improved the baseline even when dealing with unbalanced datasets. The motivation for applying these methods is that they may help improve the model's ability to handle false negatives more effectively, leading to more accurate and robust pretraining.

While existing studies indicate that convolutional neural networks (CNNs) typically demonstrate robust performance for Human Activity Recognition (HAR), one limitation is that each new sample is treated independently, irrespective of previously executed activities when directly applied to windowed data. However, the probability of executing certain activities isn't independent of prior actions. For instance, it is highly unlikely to transition directly from a lying position to cycling without transitioning through stages such as standing and walking. Implementing a simple Hidden Markov Model (HMM) can account for this dependence. This approach effectively combines the simplicity and efficiency of HMMs in considering previous states with CNN's ability to predict activities based on spatial information. The HMM consists of three components: the initial probability of each class, transition probabilities between classes, and emission probabilities for each class. In this case, the output predictions (probabilities) of the CNNs will reflect the emission probabilities, while the transition probabilities will be based on statistics from the dataset, i.e. how often one activity is followed by another. In other words, the HMM will be performed in a post-processing step, after the initial predictions are made. Although not necessarily a method to improve the self-supervised learning step, the justification for introducing this method lies in its potential to adjust the predictions of samples that can be perceived as outliers, by incorporating statistics related to transitions between different activities. These outliers could, for example, be samples that are labeled as lying but may contain some movement due to a shift in position.

6.3.2 Improvements in data and preprocessing

There are other experiments and potential improvements that can be made regarding the data and its preprocessing. First, there is a potential to investigate a wider variety of augmentation functions for sensor data beyond the subset explored in this thesis. The importance of such exploration arises from the broad range and diversity of these functions. Exploring various augmentation methods should not be underestimated looking at the performance differences for the

various augmentations used in this thesis. We have three suggestions for further investigation that can be explored in this area. The first suggestion is to explore pattern mixing augmentation functions like SPAWNER [Kamycki et al., 2019] or weighted DBA [Forestier et al., 2017] in which both methods are based on averaging multiple time-series signals. The second suggestion is to apply multiple augmentation functions in a series for each branch of the Siamese network. This will increase the differences and randomness between the two augmented batches in the network, force the network to look at the invariances for the augmented positive pairs and discard noise occurring from the augmentations. The last suggestion is that a random augmentation function can be selected from a pool of augmentation functions for each sampled batch. The idea is to ensure randomness and variations in signal augmentations, which in turn, is expected to enhance the model's robustness through a more realistic reflection of variances found in real-world acceleration signals.

The primary focus of the original SimCLR and SimSiam papers is on image recognition, focusing on optimizing the performance of a dataset like ImageNet. This thesis, however, focuses on augmenting the raw data and utilizing one-dimensional convolutional layers. A problem is that this does not explicitly present the frequencies within the response to the model, which is a defining characteristic of certain activities. A suggestion for improvement involves converting the raw data into spectrograms prior to its input into the model. This change would reshape the input to resemble image data, aligning more closely with the techniques utilized in the previously mentioned self-supervised learning papers. By doing this, the data now contains a more explicit time-frequency representation that might be beneficial. This is also a technique that has been successful in domains like speech recognition, like the SpecAugment method proposed by Park et al. [2019].

In this thesis, we focused mainly on the HUNT4 and HARTH dataset for free-living applications, as well as included UCI-HAR for transfer learning. However, incorporating other existing HAR datasets like WISDM, MotionSense, MobiAct, and so on, have multiple applications and potential benefits. First of all, these datasets can be used to further analyze the transfer-learning capabilities of the proposed models, as these datasets consist of a wide range of sensor setups, activities, and subjects. A second suggestion is to pre-train the proposed models on a dataset that is composed of various other HAR datasets. One of the points made in the discussion was that the underwhelming performance of the contrastive and non-contrastive might originate from the skewness that the HUNT4 dataset has towards sedentary activities. After all, a large portion of the population has jobs that involve sitting in a chair for most of the day at work. By pre-training the model on a dataset consisting of multiple other datasets from various studies, we can achieve the benefits of data volume for pre-training but potentially maintain a more uniform distribution of various activities that the model can learn from. Additionally, the model might be more robust to minor variations in sensor positions and types of subjects. However, if the goal is specifically to achieve good performance on the HARTH dataset, the chosen datasets to be combined for pre-training should have sensor setups that resemble the HARTH dataset and should preferably use two accelerometers. Some examples include the Opportunity dataset [Roggen et al., 2010a] the PAMAP2 dataset [Reiss and Stricker, 2012] and WISDM [Weiss, 2019] which are all given a short description in the appendix among various other HAR datasets.

During supervised training, we utilized a simple random oversampling technique to ensure that each batch contains a fairly uniform distribution of classes. While this oversampling technique significantly enhanced the performance of minority classes compared to scenarios without any oversampling, it's important to note that such a straightforward oversampling approach

could potentially increase the risk of overfitting. This is especially true at higher oversampling rates, as pointed out in Branco et al. [2016]. A suggestion is to utilize some of the existing created augmentation functions from 4.2 and 4.1 like *jittering, timewarp* and *scaling* to increase the variations of the data and create additional synthetic samples.

6.3.3 Further extensions and applications

One further extension can be in the field of sports analytics. Mounting accelerometers on athletes can enable the analysis of their activity profiles during events, particularly in sports involving free-roaming movements such as football and handball. In this case, the desired activities can be more complex than analyzed in this thesis and include activities such as *jumping, walking backward, jogging, falling* and *sprinting*. Another application is within the medical treatment of patients with Parkinson's disease. Patients with Parkinson's suffer from freezing of gaits (FOG), which is one of the least understood, yet disabling symptoms of Parkinson's. FOG is an episodic inability to step, often triggered by starting to walk or turning. Its presence can lead to balance issues, increased falls, and a decreased quality of life [Rahimpour et al., 2021]. By building a reliable system that detects FOG, the system can provide a rhythmic audio signal that stimulates the patient to resume the performed activity. There is a limited amount of research done in this exact area, but one existing study is done by [Bachlin et al., 2009] which includes a data collection process of people suffering from FOG. The dataset is publicly available in the UC Irvine machine learning repository [1].

Federated learning is a machine learning approach where a model is trained across multiple devices or servers holding local data samples, without exchanging these samples. This allows for training on a large amount of decentralized data, improving the model's accuracy and quality, while also preserving data privacy since raw data does not need to leave its original device. This makes it highly interesting for HAR, due to the large amount of private activity data that emerges from devices like smartwatches and smartphones. Also, because only model updates need to be transmitted rather than large amounts of raw data, federated learning can be more network-efficient than traditional cloud-based machine learning approaches. Implementing federated learning for HAR presents certain challenges. One problem is that data collected from devices like smartwatches are likely to be non-independent and identically distributed (non-IID). This means that the data from each device are expected to have unique distributions, influenced by individual habits, interests, and locations. Additionally, the quantity of data available on each device is likely to vary, which needs to be taken into account during the update of a global model. The presence of non-IID data in federated learning can be a challenge as gradient updates from different devices may guide the model in different directions. This can make it difficult to achieve a consistent and effective learning process.

[1] https://archive.ics.uci.edu/dataset/245/daphnet+freezing+of+gait

Bibliography

Anguita, D., Ghio, A., Oneto, L., Parra Perez, X., and Reyes Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, pages 437–442.

Åsvold, B. O., Langhammer, A., Rehn, T. A., Kjelvik, G., Grøntvedt, T. V., Sørgjerd, E. P., Fenstad, J. S., Heggland, J., Holmen, O., Stuifbergen, M. C., et al. (2023). Cohort profile update: the hunt study, norway. *International journal of epidemiology*, 52(1):e80–e91.

Bachlin, M., Plotnik, M., Roggen, D., Maidan, I., Hausdorff, J. M., Giladi, N., and Troster, G. (2009). Wearable assistant for parkinsonâs disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446.

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al. (2023). A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*.

Banos, O., Galvez, J.-M., Damas, M., Pomares, H., and Rojas, I. (2014). Window size impact in human activity recognition. *Sensors*, 14(4):6474–6499.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Blunck, Henrik, B. S. P. T. K. M. and Dey, A. (2015). Heterogeneity Activity Recognition. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5689X.

Branco, P., Torgo, L., and Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM computing surveys (CSUR)*, 49(2):1–50.

Bulling, A., Blanke, U., and Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):1–33.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Doersch, C. and Zisserman, A. (2017). Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Ericsson, L., Gouk, H., and Hospedales, T. M. (2021). How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423.

Forestier, G., Petitjean, F., Dau, H. A., Webb, G. I., and Keogh, E. (2017). Generating synthetic time series to augment sparse datasets. In *2017 IEEE international conference on data mining (ICDM)*, pages 865–870. IEEE.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

Guan, D., Yuan, W., Lee, Y.-K., Gavrilov, A., and Lee, S. (2007). Activity recognition based on semi-supervised learning. *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*.

Guan, Y. and Plötz, T. (2017). Ensembles of deep lstm learners for activity recognition using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):1–28.

Hammerla, N. Y., Halloran, S., and Plötz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*.

Haresamudram, H., Essa, I., and Plötz, T. (2021). Contrastive predictive coding for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–26.

Haresamudram, H., Essa, I., and Plötz, T. (2022). Assessing the state of self-supervised human activity recognition using wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–47.

Harish, Beedu, A., Agrawal, V., Grady, P. L., Essa, I., Hoffman, J., and Plötz, T. (2020). Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 international symposium on wearable computers*, pages 45–49.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hessen, H.-O. and Tessem, A. J. (2016). Human activity recognition with two body-worn accelerometer sensors. Master's thesis, NTNU.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Jain, Y., Tang, C. I., Min, C., Kawsar, F., and Mathur, A. (2022). Collossl. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1â28.

Jiang, W. and Yin, Z. (2015). Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1307–1310.

Kamycki, K., Kapuscinski, T., and Oszust, M. (2019). Data augmentation with suboptimal warping for time-series classification. *Sensors*, 20(1):98.

Khaertdinov, B., Ghaleb, E., and Asteriadis, S. (2021). Contrastive self-supervised learning for sensor-based human activity recognition. In *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8. IEEE.

Kofod-petersen, A. (2015). How to do a structured literature review in computer science - researchgate.

LeCun, Y. (2021). Self-supervised learning: The dark matter of intelligence.

Liu, D. and Abdelzaher, T. (2021). Semi-supervised contrastive learning for human activity recognition. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 45–53. IEEE.

Logacjov, A., Bach, K., Kongsvold, A., Bårdstu, H. B., and Mork, P. J. (2021). Harth: A human activity recognition dataset for machine learning. *Sensors*, 21(23):7853.

Mahmud, S., Tonmoy, M., Bhaumik, K. K., Rahman, A. M., Amin, M. A., Shoyaib, M., Khan, M. A. H., and Ali, A. A. (2020). Human activity recognition from wearable sensor data using self-attention. *arXiv preprint arXiv:2003.09018*.

Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. (2018a). Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, pages 1–6.

Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. (2018b). Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, pages 1–6.

Micucci, D., Mobilio, M., and Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101.

Mitchell, T. M. et al. (2007). *Machine learning*, volume 1. McGraw-hill New York.

Murad, A. and Pyun, J.-Y. (2017). Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11):2556.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Ordóñez, F. J. and Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

PlÖtz, T. (2021). Applying machine learning for sensor data analysis in interactive systems: Common pitfalls of pragmatic use and ways to avoid them. *ACM Computing Surveys (CSUR)*, 54(6):1–25.

Prince, S. A., Adamo, K. B., Hamel, M. E., Hardt, J., Gorber, S. C., and Tremblay, M. (2008). A comparison of direct versus self-report measures for assessing physical activity in adults: a systematic review. *International journal of behavioral nutrition and physical activity*, 5(1):1–24.

Rahimi, S., Rainbow, M. J., and Etemad, A. (2021). Self-supervised human activity recognition by learning to predict cross-dimensional motion. In *2021 International Symposium on Wearable Computers*, pages 23–27.

Rahimi, S. R., Rainbow, M., and Etemad, A. (2022). Self-supervised human activity recognition with localized time-frequency contrastive representation learning. *arXiv preprint arXiv:2209.00990*.

Rahimpour, S., Gaztanaga, W., Yadav, A. P., Chang, S. J., Krucoff, M. O., Cajigas, I., Turner, D. A., and Wang, D. D. (2021). Freezing of gait in parkinsonâs disease: invasive and noninvasive neuromodulation. *Neuromodulation: Technology at the Neural Interface*, 24(5):829–842.

Reiss, A. and Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109.

Robinson, J., Chuang, C.-Y., Sra, S., and Jegelka, S. (2020). Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*.

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Forster, K., Troster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., Creatura, M., and MillÃ n, J. d. R. (2010a). Collecting complex

activity datasets in highly rich networked sensor environments. In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, pages 233–240.

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al. (2010b). Collecting complex activity datasets in highly rich networked sensor environments. In *2010 Seventh international conference on networked sensing systems (INSS)*, pages 233–240. IEEE.

Ronao, C. A. and Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244.

Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

Saeed, A., Ozcelebi, T., and Lukkien, J. (2019). Multi-task self-supervised learning for human activity detection. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1â30.

Saeed, A., Salim, F. D., Ozcelebi, T., and Lukkien, J. (2021). Federated self-supervised learning of multisensor representations for embedded intelligence. *IEEE Internet of Things Journal*, 8(2):1030â1040.

Skauge, S. N. (2021). Vigorous activity detection in human activity recognition. Master's thesis, NTNU.

Tang, C. I., Perez-Pozuelo, I., Spathis, D., Brage, S., Wareham, N., and Mascolo, C. (2021). Selfhar: Improving human activity recognition through self-training with unlabeled data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1):1â30.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839.

Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., and Pal, C. J. (2017). Deep complex networks.

Vågeskar, E. (2017). Activity recognition for stroke patients. Master's thesis, NTNU.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vavoulas, G., Chatzaki, C., Malliotakis, T., Pediaditis, M., and Tsiknakis, M. (2016). The mobiact dataset: Recognition of activities of daily living using smartphones. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, volume 2, pages 143–151. SciTePress.

Wang, G., Wang, K., Wang, G., Torr, P. H., and Lin, L. (2021). Solving inefficiency of self-supervised representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9505–9515.

Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. *Pattern recognition letters*, 119:3–11.

Warburton, D. E., Nicol, C. W., and Bredin, S. S. (2006). Health benefits of physical activity: the evidence. *Cmaj*, 174(6):801–809.

Weiss, G. M. (2019). Wisdm smartphone and smartwatch activity and biometrics dataset. *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*, 7:133190–133202.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698.

Yao, S., Hu, S., Zhao, Y., Zhang, A., and Abdelzaher, T. (2017). Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th international conference on world wide web*, pages 351–360.

Yao, S., Piao, A., Jiang, W., Zhao, Y., Shao, H., Liu, S., Liu, D., Li, J., Wang, T., Hu, S., et al. (2019). Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks. In *The World Wide Web Conference*, pages 2192–2202.

Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., and Tröster, G. (2008). Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *European Conference on Wireless Sensor Networks*, pages 17–33. Springer.

Zeng, M., Gao, H., Yu, T., Mengshoel, O. J., Langseth, H., Lane, I., and Liu, X. (2018). Understanding and improving recurrent networks for human activity recognition by continuous attention. In *Proceedings of the 2018 ACM international symposium on wearable computers*, pages 56–63.

Zhang, M. and Sawchuk, A. A. (2012). Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 1036–1043.

Appendix A

Additional Results

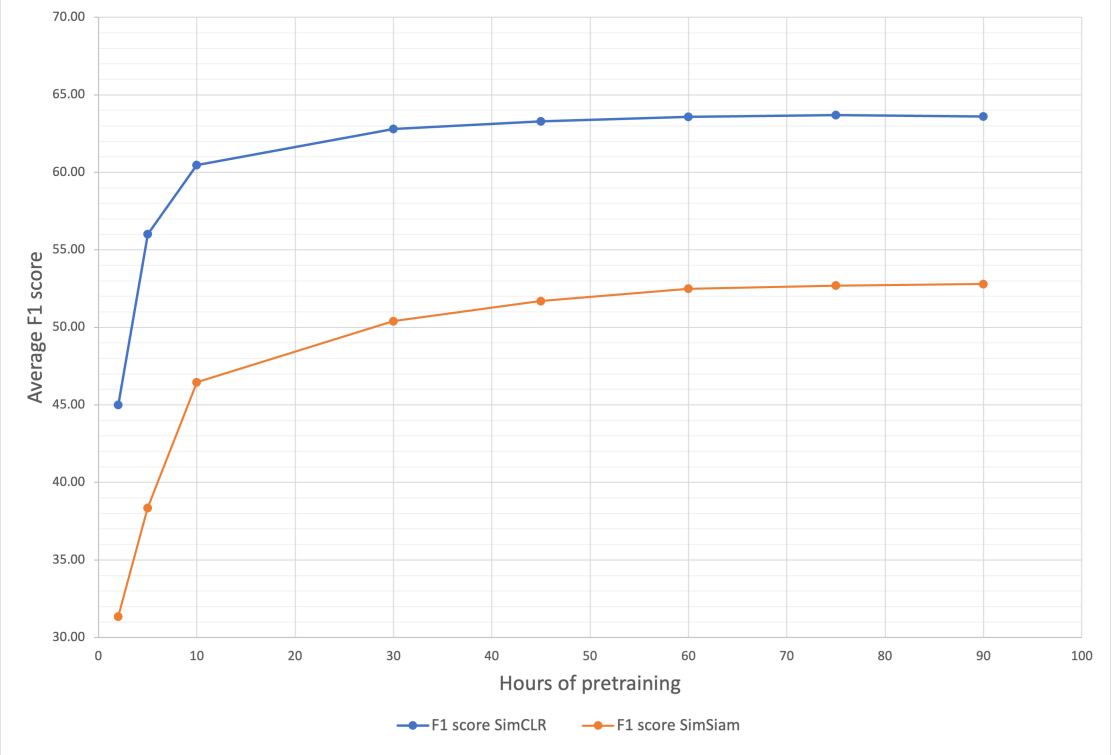


Figure A.1: Downstream performance of SimCLR and SimSiam on the HARTH dataset using various hours of pretraining on the HUNT4 dataset

Appendix B

Related Work Supplements

Dataset	Description
Opportunity [Roggen et al., 2010a]	Sensor information was collected from participants performing various tasks within a mock studio apartment setting. Participants were given a general script to follow, yet were given the freedom to execute the activities based on their interpretation whenever they wanted. Each participant was part of six recording sessions, one of which was specifically tailored to produce a significant quantity of activity instances.
UCI-HAR [Anguita et al., 2013]	This data set contains information gathered from 30 voluntary participants, each engaged in six activities with a smartphone attached to their waist. The collection process took place within a controlled laboratory environment by using the smartphone's accelerometer and gyroscope. The data was captured at a frequency of 50 Hz. Video recordings of the experiments were utilized to manually assign labels to the data at each frameshift.
HHAR [Blunck and Dey, 2015]	The dataset utilized in this study contains sensor data gathered from smartphones and smartwatches. This data was used to investigate the influence of sensor dissimilarities on human activity recognition algorithms. The collected data represent measurements from nine users executing six predetermined activities, while wearing a smartwatch and carrying a smartphone. The sequence in which these activities were carried out was not defined.
MobiAct [Vavoulas et al., 2016]	This dataset, accessible to the public, contains data collected from a smartphone used by participants engaged in a variety of activities, including doing a variety of falls. Containing four distinct fall types, twelve various daily routines, and in excess of 3200 trials from 66 individuals, the chosen activities were identified considering factors such as the resemblance to fall events, short or quick movements, and normal everyday tasks.
HAPT	In this study, information was gathered from a cohort of 30 individuals who performed a variety of tasks while having a smartphone attached to their waist. The participants utilized the inbuilt accelerometer and gyroscope of their smartphone, readings were captured at a frequency of 50 Hz, and all the tasks were video recorded for manual labeling purposes. The actions carried out by participants included both stationary stances and active movements, along with transitioning from one stationary position to another.
MotionSense [Malekzadeh et al., 2018b]	This dataset comprises time-series data captured in a controlled laboratory setting, using accelerometer and gyroscope sensors. Collected from 24 participants performing six distinct activities under identical conditions and environments, the aim of the dataset is to explore if it is possible to predict personal characteristics, including gender and personality traits, from the sensor-acquired time-series data.
UniMib [Micucci et al., 2017]	This dataset consists of recordings from 30 participants, ranging from 18 to 60 years of age, who performed a variety of regular activities, including falls. The data is separated into 17 detailed classes, which are reduced into two broad categories. The first category consists of data from 9 daily activities, and the second category comprises data from 8 types of falls. The dataset is arranged in a way that makes it easy to filter and select of samples based on diverse parameters like activity category, age, and gender.
PAMAP [Reiss and Stricker, 2012]	Sensor data were gathered from nine individuals engaging in 18 distinct physical tasks. The recordings were enabled through wearable technology like inertial measurement units and a device tracking heart rate. The participants took part in a diverse set of physical activities including walking, bike riding, and participation in athletic games.
WISDM [Weiss, 2019]	Data were collected from 51 individuals who were instructed to perform 18 specific activities, each lasting for a duration of three minutes. The participants were equipped with a smartwatch in their primary hand and a smartphone tucked into their pockets.
USC-HAD [Zhang and Sawchuk, 2012]	This dataset contains consist of sensor recordings from 14 individuals engaged in 12 regular day-to-day activities. It's tailored to include a broad range of subjects with a variety of physical attributes, including weight and height, as well as demographic factors such as gender and age. Data relating to activities was recorded utilizing a high-grade inertial sensor, as opposed to less expensive and less accurate alternatives.

Table B.1: Benchmark datasets used by the literature

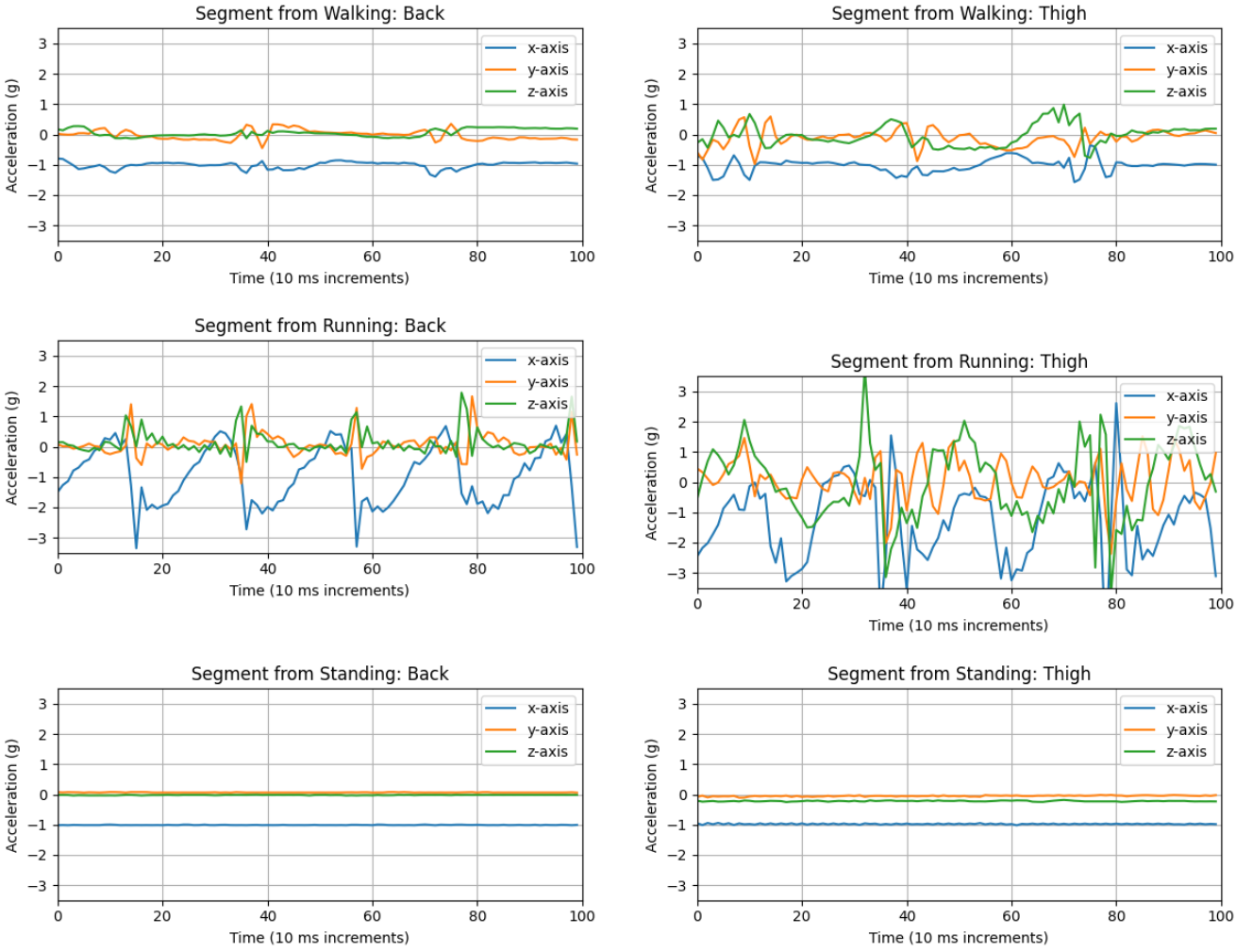
Model names	HHAR	MotionS	MobiA	UCIHAR	USCHAD	HAPT	PAMAP	Opport.	UniMb	WISDM
SCN [Saeed et al., 2021]	82 (w)	91 (w)								
Rahimi et al. [2022]	82.6 (w)	93.4 (w)				91.1 (w)				
ColloSL [Jain et al., 2022]							78.4 (m)	70.0 (m)		
CPC [Haresamudram et al., 2021]		88.22 (m)	76.24 (m)	90.24 (m)	65.15 (m)					
CSSHAR [Khaertdinov et al., 2021]			81.13 (m)	91.14 (m)	57.76 (m)					
MultiTask-SSL [Saeed et al., 2019]	78.62 (w)	90.05 (w)		89.46 (w)					84.25 (w)	86.86 (w)
SelfHAR [Tang et al., 2021]	78.46 (w)	96.31 (w)	93.55 (w)	91.35 (w)					87.31 (w)	90.81 (w)
SemiC-HAR [Liu and Abdelzaher, 2021]	85.10 (w)	93.93 (w)	94.79 (w)	92.64 (w)						90.06 (w)
Harish et al. [2020]		88.02 (m)	76.81 (m)	81.89 (m)	49.31 (m)					
Rahimi et al. [2021]		91.8 (m)		90.8 (m)		79.2 (m)				
		93.4 (w)		90.6 (w)		90.0 (w)				
DRNN [Murad and Pyun, 2017]				96.0 (w)	97.0 (w)			92.0 (w)		
Ensemble LSTM [Guan and Plötz, 2017]							85.4 (m)	72.6 (m)		
DeepConvLSTM [Ordóñez and Roggen, 2016]								70.4 (m) 91.7 (w)		

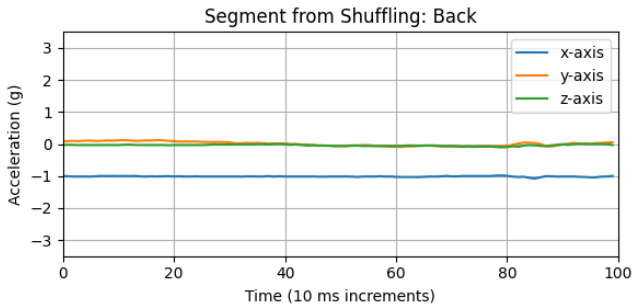
Table B.2: Overview of the F1-scores reported by various authors on baseline HAR datasets. Most of these datasets are data collected in-lab. (m) = macro F1-score, (w) = weighted F1-score.

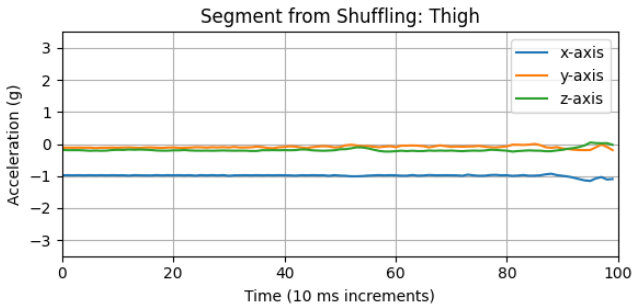
Appendix C

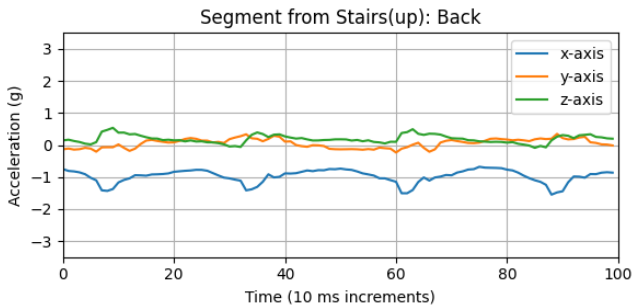
Window Segments

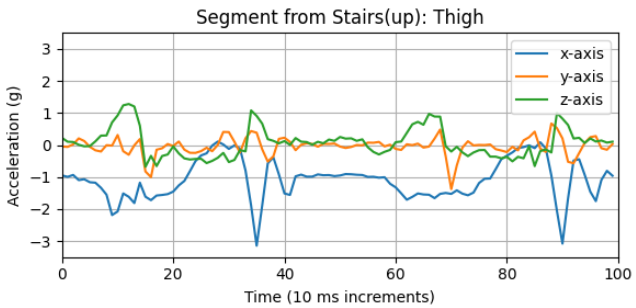
Figure C.1: Segments from all 10 activities performed by one participant.



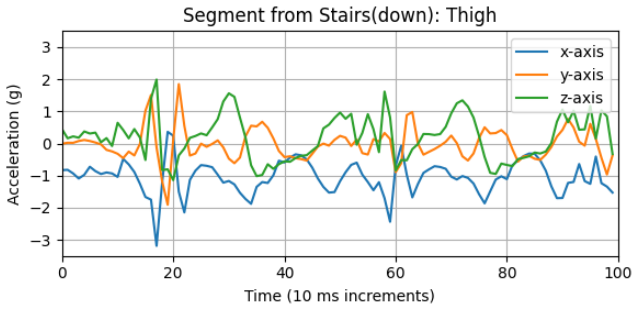

Segment from Shuffling: Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

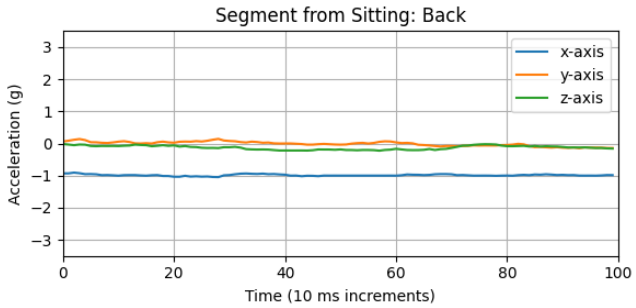

Segment from Shuffling: Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

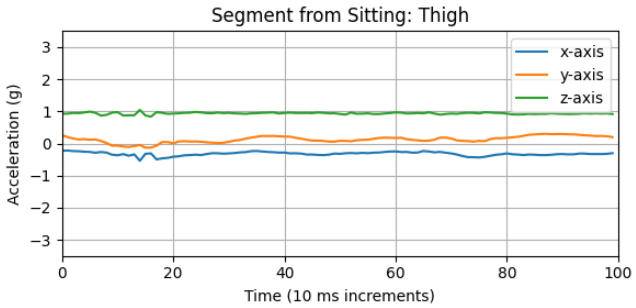

Segment from Stairs(up): Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

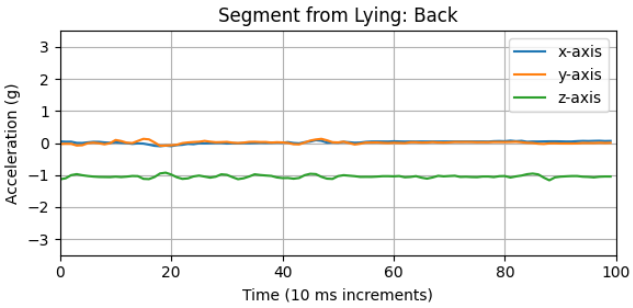

Segment from Stairs(up): Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

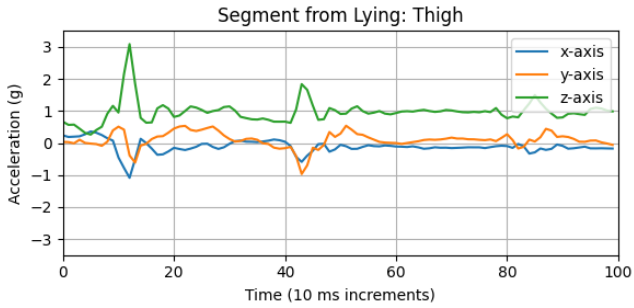

Segment from Stairs(down): Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

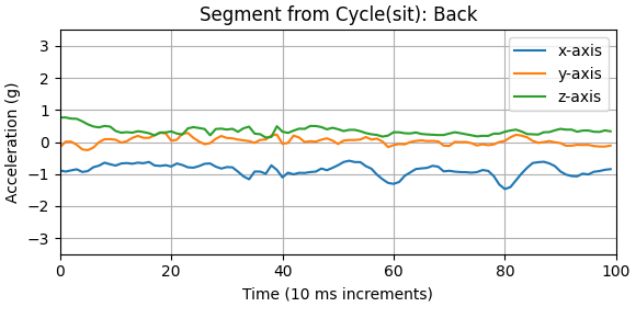

Segment from Stairs(down): Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

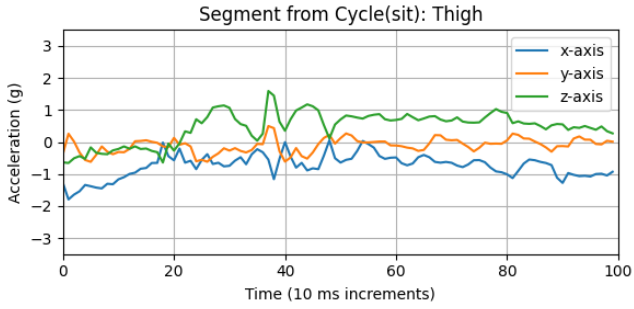

Segment from Sitting: Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

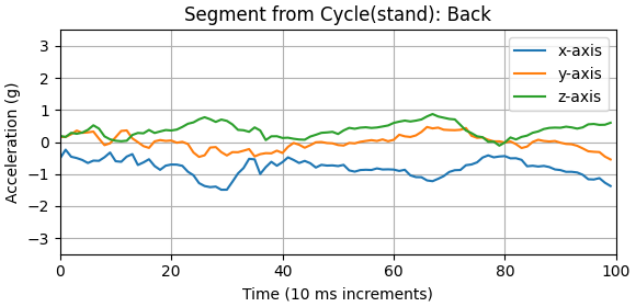

Segment from Sitting: Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

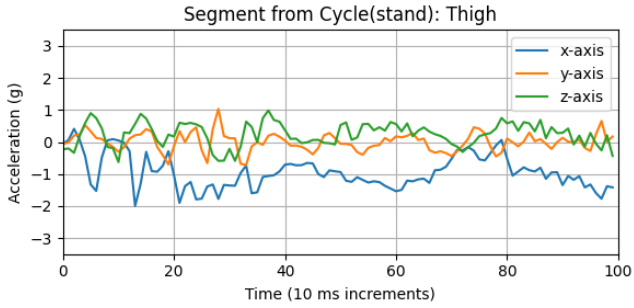

Segment from Lying: Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)


Segment from Lying: Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)


Segment from Cycle(sit): Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)


Segment from Cycle(sit): Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)


Segment from Cycle(stand): Back
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)


Segment from Cycle(stand): Thigh
x-axis
y-axis
z-axis
Acceleration (g)
Time (10 ms increments)

Appendix D

Pseudocode

Algorithm 1 SimCLR pseudocode

procedure SIMCLR
  Initialize base encoder f, projection head g
  procedure SIM(z_i, z_j) ▷ Similarity function, often cosine similarity
    Return $z_i^T z_j$ / ($\|z_i\| \cdot \|z_j\|$)
  end procedure
  for each batch x in loader **do**
    $x_1, x_2 = aug_1(x)$, $aug_2(x)$ ▷ Random augmentation
    $z_1, z_2 = g(f(x_1))$, $g(f(x_2))$ ▷ Projections
    for all i in $[1, 2N]$ and j in $[1, 2N]$ **do**
      $s_{i,j} = \text{sim}(z_i, z_j)$ ▷ Pairwise similarity
    end for
    for all i, j in $[1, 2N]$ **do**
      $\ell(i, j) = -\log\left(\frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(s_{i,k}/\tau)}\right)$ ▷ Compute '(i, j)
    end for
    $L = \frac{1}{2N} \sum_{k=1}^{N} (\ell(2k-1, 2k) + \ell(2k, 2k-1))$
    Backpropagate $L.backward()$
    update f, g ▷ Adam update
  end for
  return f ▷ Return the trained encoder network
end procedure

Algorithm 2 SimSiam pseudocode

procedure SimSiam
  Initialize base encoder f, projection head g, and prediction MLP h
  procedure D(p, z) ▷ Negative Cosine Similarity
    z.detach() ▷ Stop gradient
    Normalize p and z along $dim = 1$
    Return $-(p \cdot z)$.sum(dim=1).mean()
  end procedure
  for each batch x in loader **do**
    $x_1, x_2 = aug_1(x)$, $aug_2(x)$ ▷ Random augmentation
    $z_1, z_2 = f(x_1)$, $f(x_2)$ ▷ Projections
    $p_1, p_2 = h(z_1)$, $h(z_2)$ ▷ Predictions
    Compute the loss $L = D(p_1, z_2)/2 + D(p_2, z_1)/2$
    Backpropagate $L.backward()$
    update f, g, and h ▷ SGD update
  end for
  return f ▷ Return the trained encoder network
end procedure

Appendix E

Software

List of software used and its versions.

Python 3.8.10 Programming language used.

PyTorch 2.0.0 [Paszke et al., 2019]. for implementation of various models.

Weights and Biases 0.13.10 [Biewald, 2020]. Used to keep track of all the experiments by logging hyperparameters and output metrics from runs. Great visualization tools for losses and other various metrics.

Numpy 1.23.4 [Harris et al., 2020]. Tool used for handling arrays and matrices.

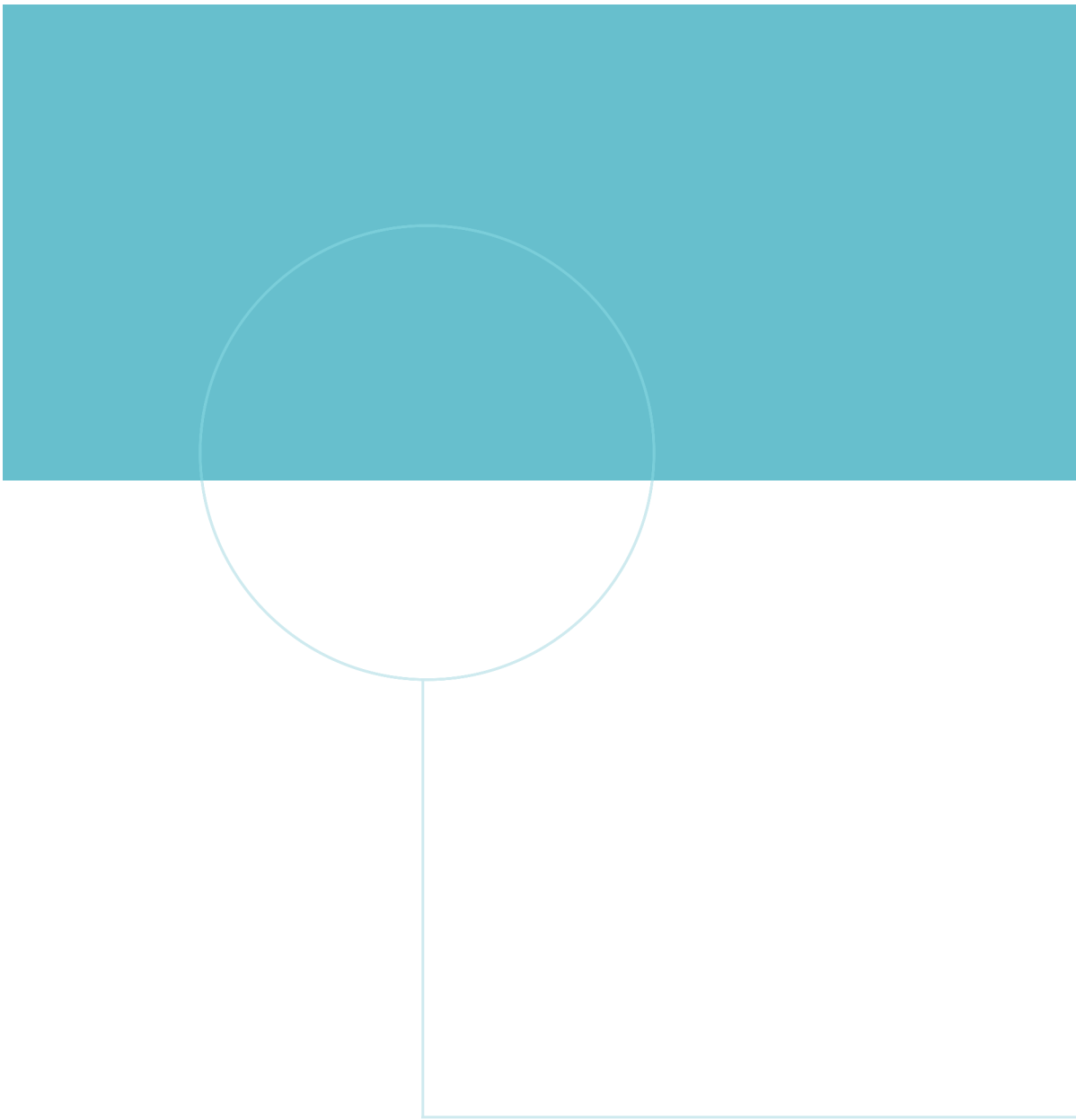
Pandas 1.5.3 [Wes McKinney, 2010].

Matplotlib 3.7.1 [Hunter, 2007].

Scikit-learn 1.2.2 [Pedregosa et al., 2011].

JetBrains client (pycharm gateway) 2022.2.5. Lightweight IDE for remote development. Used to connect to the NTNU leia server and get access to the GPU and the required datasets.

Figma and draw.io Used to create figures to illustrate the various models and setups in the methodology.



NTNU

Norwegian University of
Science and Technology