UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Improving the Corona Warn App with human activity recognition

Verbesserung der Corona Warn App mit Hilfe von human activity recognition

Bachelorarbeit

verfasst am
Institut für IT-Sicherheit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Julian Grimm

ausgegeben und betreut von
Prof. Dr. Esfandiar Mohammadi, Dr.-Ing. Frédéric Li, Dr.-Ing. Adeel Nisar

Lübeck, den 18. November 2022

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

_____

Julian Grimm

Zusammenfassung

Seit Beginn der Covid19-Pandemie gibt es Initiativen zur Verbesserung der Systeme zur Ermittlung von Kontaktpersonen, um das Virus besser einzudämmen. Der Kontakt mit einer infizierten Person ist jedoch nicht das einzige Kriterium, das für die Ansteckung mit dem Virus relevant ist. Die Erkennung der von den Personen ausgeübten Aktivitäten könnte hilfreich sein, um ihren Standort und den Kontext, in dem sie interagieren, einzuschätzen. In dieser Arbeit habe ich ein neuronales Netz trainiert, das die Aktivitäten des Nutzers mittels Daten verschiedener Arten von Smartphones klassifizieren kann. Da Smartphones heutzutage allgegenwärtig sind und über eine große Anzahl von Sensoren verfügen, sind sie ein optimales Instrument, um Daten zur Erkennung von Aktivitäten auf relativ unaufdringliche Weise zu sammeln. Die vorhandenen Datensätze für die Smartphone-basierte Aktivitätserkennung sind jedoch klein, was für eine maschinelle Lernaufgabe suboptimal ist. Hinzu kommt, dass verschiedene Smartphones recht unterschiedliche Sensordaten liefern, selbst wenn sie die gleichen Sensoren verwenden. Aufgrund dieser Probleme wird die Idee des transfer and contrastive learnings in Betracht gezogen, um Informationen von einer source domain auf einen target domain in sinnvoller Weise zu übertragen und so die Performanz in der Zielaufgabe zu verbessern. In den Experimenten zeigt sich, dass die Verwendung verschiedener Datensätze aus der Literatur zum pre-training des Modells mit einem contrastive loss zu besseren Leistungen führt als ein einfacher transfer learning Ansatz.

Abstract

Since the beginning of the Covid19 pandemic, there have been initiatives to improve contact tracing systems to better contain the virus. However, contact with an infected person is not the only criterion that is relevant for contracting the virus. Recognizing the activities performed by the individuals could be helpful to estimate their location and the context in which they are interacting. In this thesis, I trained a neural network that can classify the user's activities using data coming from various smartphone types. Since smartphones are ubiquitous nowadays and have a large number of sensors, they are an optimal tool to collect data to recognize activities in a fairly non-intrusive way. But existing data sets for smartphone-based activity recognition are small, which is suboptimal for a machine learning task. In addition different smartphones return quite different sensory data, even if they use the same sensors. Due to these problems the idea of using transfer and contrastive learning is considered and a machine learning model is created using a contrastive supervised loss. The idea of transfer learning is to train a model on a source data set and afterwards use it on a different target data set. In the experiments, it is shown that using different data sets from the literature to pre-train the model with a contrastive loss leads to better performances than with a simple transfer learning approach.

Acknowledgements

I would like to express my sincere thanks to Dr.-Ing. Frédéric Li who, despite his workload, always found time to answer my questions and support me.

I would also like to thank Prof. Dr. Esfandiar Mohammadi and Dr.-Ing. Adeel Nisar, who constantly provided me with interesting suggestions.

Last but not least, I would like to thank my fellow students who supported me with either advice or words of encouragement.

Contents

1 Introduction **1**

1.1 Context and Motivation 1
1.2 Contributions of this Thesis 2
1.3 Related Work 2
1.4 Structure of this Thesis 3

2 Fundamentals **4**

2.1 Activity Recognition Chain 4
2.2 Deep Neural Networks 4
2.3 Convolution Neural Networks 5
2.4 Transfer Learning 6
2.5 Contrastive Learning 7

3 Approach / Methodology **8**

3.1 Methodology 8
3.2 Experimental Overview 8

4 Experiments and Results **10**

4.1 System design 10
4.2 Data Sets 10
4.3 The Baseline Experiment 12
4.4 Basic Transfer Learning 16
4.5 Contrastive Learning 19
4.6 Contrastive Learning with the UCI dataset 23
4.7 Contrastive Learning with the UCI dataset and WISDM dataset 24

5 Discussion **26**

6 Conclusion and Future Work **28**

1

Introduction

1.1 Context and Motivation

Since the beginning of the Covid19 pandemic, various tools have been developed to better contain the pandemic. One of these tools contact tracking apps, which where developed by various countries to combat the pandemic. The app, developed in Germany, uses Bluetooth Low Energy to check if other devices that also have the corresponding app installed are close enough for their owners to be infected.
However, the distance to other people is not the only criterion relevant for infection. For example, two drivers in different cars could be standing next to each other at a traffic light. They would therefore be close to each other, but would be in different spaces. This could lead to the app issuing an unnecessary warning, even though there is no risk whatsoever in this specific case. To determine the risk of infection more reliably, it would be advantageous for such an app to recognize the current action that a person was performing at the time of contact. This would make it easier to assess whether people can really infect each other or are simply spatially close to each other.
Fortunately, smartphones have numerous sensors that provide valuable data. Almost all current devices have a microphone, an accelerometer and a gyroscope, and many even have additional sensors such as a linear accelerometer, a magnetometer or a gravity sensor. These can be used to collect data in a relatively unobtrusive way that can then be processed using machine learning techniques to draw conclusions about other properties of the wearer of the device. For example, Johannes Liebenow showed in his case study (Liebenow, 2021) that it is possible to identify the surroundings of the smartphone based on acoustic signals in the vicinity of a device. However, audio data by itself is not sufficient to classify the context of the person. For example, a covered microphone can lead to incorrect classification. Since many people carry their smartphone in their pocket, this problem occurs more frequently. Fortunately, smartphones have other sensor modalities that can be used, for example, to enable better recognition of the environment or to directly identify activities that increase or reduce the risk of infection.
In this thesis, the most commonly available in sensors, i.e. 3D accelerometers and gyroscopes, are used to draw conclusions about the current activity of the user. I used supervised machine learning techniques to train a model to recognize various activities relevant

for context recognition (Bending, Lying, Sitting, Squatting, Standing, Walking) given input accelerometer and gyroscope data. More specifically, I trained a convolutional-based neural network to learn relevant features for activity recognition on the publicly available Cognitive Village (CogAge) dataset (Li et al., 2020; Nisar et al., 2020).
However, in order to provide the most comprehensive protection possible, the app must be able to work on a very large number of devices, regardless of their differences in sensors, while still providing good classification. The difficulty here is that even data from the same sensors on different smartphones can sometimes differ greatly, and that machine learning models trained on one device might not perform well anymore when transposed to another (Stisen et al., 2015). In different experiments I showed that a basic transfer of a neural network between source datasets publicly available online and the CogAge dataset as target, does not achieve good results. In this thesis, I also pre-trained a model based on supervised contrastive learning and investigated whether this model achieves better classification results. It can be seen that the performance of the supervised contrastive learning approach is significantly better than the basic transfer approach.

1.2 Contributions of this Thesis

In this thesis, I conducted experiments using the basic transfer learning approach consisting in training a model on a first publicly available smartphone-based dataset, and then fine-tuning it on another target one. These show that the accuracy of the target task drops significantly in this context.
Another experiment using the supervised contrastive learning approach with two publicly available smartphone-based data sets leads to the result that this approach works better than the basic transfer learning approach.
Furthermore, a third experiment was conducted using three publicly available smartphone-based activity recognition datasets. Here, two of the datasets were used as the source for pretraining, and the third served as the target. This experiment showed that adding more data had minimal effect on further accuracy compared to the previous experiment.

1.3 Related Work

Jialin Pan et al. gave a comprehensive overview of different transfer learning techniques and explained differences between three primary approaches (inductive transfer learning, transductive transfer learning, unsupervised transfer learning) (Pan and Yang, 2009). Matthew D. Zeiler et al. provided a deeper look into how convolutional neural networks work and analyzed why they perform so well (Zeiler and Fergus, 2013). Ting Chen et al. presented SimCLR, a contrastive learning framework for visual representations. Among other things, they showed that the type of data augmentation plays an important role in contrastive learning and that contrastive learning benefits more than supervised learning from larger batch sizes and a higher number of training steps (Chen et al., 2020).

Khosla et al. show how they can use label information to create a supervised contrastive learning approach. They present their supervised contrastive loss and show that it performs better than cross entropy loss. (Khosla et al., 2020). Allan Stisen et al. investigated how the heterogeneity of smartphones affects the aspect of human activity recognition. They showed that differences in the hardware and operating systems of different smartphones affect the recognition process. (Stisen et al., 2015). Frédéric Li et al. developed an evaluation framework to compare feature learning methods for sensor based human actitivity recognition (Li et al., 2018). Johannes Liebenow showed in his case study an acoustic scene classification process which could extend the utility of the current Corona Warn App (Liebenow, 2021). Furthermore, he presented in his master thesis a privacy-preserving way to process these data (Liebenow, 2022). In his bachelor thesis, Timothy Imort developed methods to efficiently integrate the acoustic scene classification process into the Corona Warn app (Imort, 2022).

1.4 Structure of this Thesis

In chapter two, the fundamental concepts relevant to the thesis are discussed. These include the activity recognition chain, feature extraction, convolutional neural networks, transfer and contrastive learning.
In chapter three I explain the methods and materials used in the frame of this thesis. I briefly talk about the challenges and then conclude with a brief overview of the experiments I conducted.
In chapter four, I briefly describe the software and tools I used in the experiments. I discuss the different data sets, explain what makes them different and what modifications are necessary to use them in the experiments. Afterwards I explain the experiments I conducted in detail, more specifically regarding their implementation, what problems occurred in each case, and how they were solved. The results of each experiment are also presented in this chapter.
Subsequently, these results are discussed in chapter five.
Finally, in chapter six, conclusions follow, as well as possibilities of future directions to expand on this work.

2

Fundamentals

2.1 Activity Recognition Chain

To obtain a properly trained machine learning model, it is common to follow a standardized procedure referred to as the Activity Recognition Chain (Bulling, Blanke, and Schiele, 2014) as shown in *figure 2.1*. It includes the following steps:

1. **Acquisition of sensorical data:** First, the raw data of the sensors selected for the activity recognition process are recorded at a specified frequency.
2. **Pre-processing:** In the second step, the collected data must be converted from the raw form into a form suitable for analysis. For example, errors in the records or problems in the sampling are corrected.
3. **Data Segmentation:** In this step, the pre-processed data is divided into segments containing the desired information. This is often made more difficult by the fact that movements are executed constantly and in a flowing change.
4. **Feature Extraction:** In this step the segments are reduced to features describing them. It is important to find robust features which are as close as possible between subjects for the same activity, but which are as far away as possible from the features of other activities.
5. **Classification:** In the last step, a classifier is developed with the help of the extracted features. This should distinguish between the different activities as reliably as possible.

With the exception of data acquisition, I worked on each of these steps as part of my thesis.

2.2 Deep Neural Networks

Deep Neural Networks (DNNs) are a class of machine learning models that have become very popular over the past years due to their ability to learn strong features automatically (Chen et al., 2020). They consist of an input and an output layer of artificial neurons that are simple computational units applying a non-linearity on a weighted sum of in-

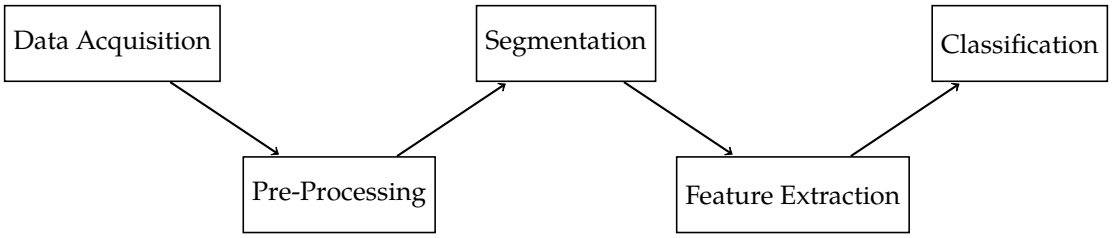

Figure 2.1: Activity Recognition Chain: First, the data must be collected using the sensors, then it is converted into a form suitable for analysis, followed by segmentation into relevant sections, then the relevant features are extracted and then the features are used for classification.

puts. DNNs also have any number of hidden layers between the input and output. An example DNN is shown in *figure 2.2*.

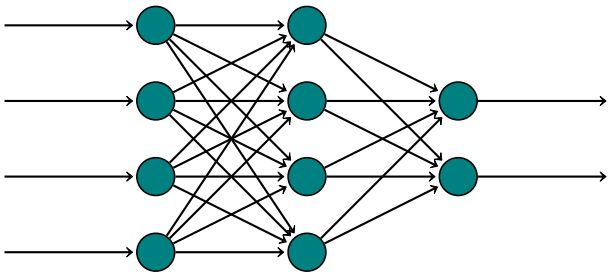

Figure 2.2: Example DNN: The inputs are connected to the input layer which is connected to the hidden layer. Then follows the output layer and the outputs

2.3 Convolution Neural Networks

A Convolution Neural Network (CNN) is a subclass of DNN. They have gained importance due to the fact that they achieve very good performances for various classification tasks involving either images or other types of data, even with significantly fewer parameters than DNNs. A CNN is a model based on convolutional layers, which are usually combined with pooling layers. In convolutional layers, kernels are used as filters to make the input data smaller and thus easier to process. A kernel is a matrix with the same dimensions as the input data, but much smaller in size. *Table 2.3* shows examples of kernels. The process of convolution involves moving kernels over the data matrix and performing an addition of the products of the corresponding cells of the data with the cells of the kernel. *Table 2.4* illustrates the process.
The convolution process is usually followed by a pooling process in pooling layers. Pooling again reduces the complexity of the matrix created by convolution. There are several recognized pooling strategies. Among the best known are max pooling or average pooling. In both cases, a fixed size matrix is again moved over the matrix created by convolution. In the case of max pooling the maximum value of the covered area is chosen, in

1	0	1
0	1	0
1	0	1

0	1	0
1	0	1
0	1	0

-1	0	0
0	-1	1
0	0	1

Table 2.3: Three examples of possible 3x3 kernels

2	**1**	0
0	**0**	1
2	0	1

1	**0**
0	**1**

2	2
0	1

Table 2.4: Example of a Convolution: The marked part of the input data is combined with the 2x2 kernel to calculate the marked value. The calculation here is $2 = 2 * 1 + 1 * 0 + 0 * 0 + 0 * 1$

the case of average pooling the average is taken. *Table 2.5* illustrates the process of a max pooling process.
For a classification, it is common to finish the CNN with one or more fully connected layers.

2.4 Transfer Learning

Transfer learning defines a dataset and a specific task (source domain) and tries to transfer the knowledge we have learned in this domain to another dataset and task combination (target domain). The goal of transfer learning is to improve the results in the target domain. A common reason for this is, for example, to address a target dataset that would be too small to learn proper features only using it. There are many different types of transfer learning. For example, it became very popular for DNNs, because it consists in a simple transfer of parameters (weights and biases) from one model to another, and it has showed to be very effective for various image processing tasks.
In time series a transfer is often much more difficult because of the many different formats in time series.
The transfer learning is illustrated in *figure 2.6*

1	7	2	2
0	4	2	8
1	4	1	1
3	0	9	1

7	8
4	9

Table 2.5: Example of a max pooling with size 2x2: The colored areas in the result matrix correspond to the maximum of the colored areas in the data matrix

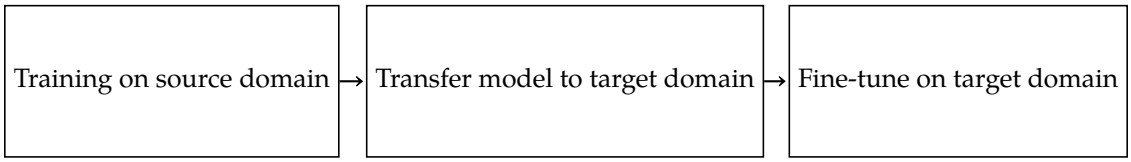

Figure 2.6: Process of transfer learning: The model is trained on the source domain. Afterwards the extracted features are transferred to the target domain

2.5 Contrastive Learning

Contrastive learning is usually an approach that extracts features from data by creating a task and its associated labels, and training a model to solve it (self-supervised learning). It learns the labels for a given task independently by minimizing the distance between features of similar examples and maximizing the distance between features of different examples. For this, positive pairs are formed from the source dataset using data augmentation techniques. For example, common augmentation techniques for positive pair generation of images include rotating or flipping the images. Two augmented examples created from the same original example form a positive pair. To minimize or maximize the distance, a contrastive loss function is used for training (Chen et al., 2020)
A disadvantage of this approach is that two examples characterizing the same information will lead to two pairs of augmented examples that are considered negative. To avoid this problem, supervised contrastive learning can be used. In contrast to self-supervised contrastive learning as used in Chen et al. (Chen et al., 2020), the supervised contrastive learning of Khosla et al. (Khosla et al., 2020) uses labeled data to classify all equally labeled examples as positive.
In both supervised and self-supervised contrastive learning, the model is trained with a contrastive loss on the source domain and then fine-tuned on the target domain.

3

Approach / Methodology

3.1 Methodology

In my thesis, I worked on the recognition of the activities bending, lying, sitting, squatting, standing and walking using machine learning. I investigated how these activities can be recognized on smartphones of different types, even if only little data is available for the classification with the respective devices. For this I used sensor data from the smartphone types Nexus 5, Nexus 5X, Galaxy S6, Galaxy S2 and LG-G5. These had at least data of the sensors 3D accelerometer and gyroscope included. The data from other devices was used as source dataset, with a source task being either the classification of source activities for transfer learning or the contrastive pre-training for contrastive learning. To investigate this problem in more detail, I conducted five different experiments, which are described in more detail in the next subsection.
Challenges encountered in the scope of work include the following:

1. **Heterogeneity of smartphones:** Even if the sensors of the smartphones are the same, they do not necessarily lead to the same data (Stisen et al., 2015). The data were usually different not only in structure, but also in their sampling rate. My approach to the solution was to restructure and interpolate them.
2. **Data Augmentation on Time Series:** The chosen approach here was window warping, as described by Le Guennec et al. (Le Guennec, Malinowski, and Tavenard, 2016).

3.2 Experimental Overview

The Baseline Experiment

In this experiment, only the target domain is considered. The target task consists of the classification of the activities associated to the target dataset. This configuration corresponds to the most commonly used in machine learning problems.

Basic Transfer Learning

In the basic transfer learning experiment, I trained a model on a source domain using the labels associated to the source dataset. It was then fine-tuned for the target domain and used to classify the validation data of the target domain.

Contrastive Learning with same source and target datasets

This experiment - like for the baseline one - had the purpose of determining the performance of classification using only the target domain. Part of the model (the encoder) was trained with a supervised contrastive loss function and then embedded into a model with multiple branches. The complete model was retrained on the target.

Contrastive Learning with one source dataset

In this experiment, the encoder was trained on the source domain with a supervised contrastive loss using the labels of the source dataset and then embedded in a model with multiple branches as before. This model was then fine-tuned on the target domain afterwards.

Contrastive Learning with two source datasets

In this last experiment, two different datasets coming from different devices were mixed and used as the source domain for training the encoder. The class labels on both datasets were combined and used for the training of the supervised contrastive loss. The transferred model was then finally fine-tuned on the target domain.

4

Experiments and Results

4.1 System design

All experiments were performed in *Jupyter Notebook* (Jupyter, 2014) or *Google Colaboratory* (Inc., n.d.) using *Tensorflow Framework* (Abadi et al., 2015) in version *2.8.2*.
The following libraries were used:
tensorflow addons (SIG-addons, n.d.), *NumPy* (Harris et al., 2020), *matplotlib* (Hunter, 2007) and *sklearn* (Pedregosa et al., 2011)
As a basis for the code of contrastive supervised learning, I used an example by *Khalid Salama* (Salama, 2020).

4.2 Data Sets

Cognitive Village data set

The Cognitive Village data set (CogAge) used here is actually a combination of two Cognitive Village data sets.

The first set (CogAge1) (Nisar et al., 2020) was recorded using three devices. A NEXUS 5X smartphone, which was carried by the subject in the left front pocket. It recorded data from five sensors (3D accelerometer, gyroscope, linear accelerometer, gravimeter and magnetometer). All sensors except the magnetometer recorded at a frequency of 200Hz. The magnetometer used a frequency of 50Hz. In addition to the smartphone, a Microsoft Band 2 was used that recorded at a frequency of 67Hz and was attached to the subject's left arm. The third device used was a JINS MEME glasses with a frequency of 20Hz. The activities were performed by four subjects 20 times each for 5 seconds each. A total of 61 activities (6 state activities, 55 behavioral activities) were recorded in this way.
Since in the context of this thesis the finished model is to be used in a wide field, and the use of a Microsoft Band 2 and JINS MEME glasses is rare, the data of these two devices were not used for the experiments carried out here. Due to the fact that some of the activities can only be classified well with the help of these devices, the set of activities was

also reduced to 6 activities (Bending, Lying, Sitting, Squatting, Standing, Walking).

The second set (CogAge2) (Li et al., 2020) was also recorded with three devices. An LG-G5 smartphone that was worn in the front left pocket and that has seven different sensors (3D accelerometer, gyroscope, linear accelerometer, gravimeter, magnetometer, barometer, orientation sensor). In addition, the subjects wore a Huawei Watch on their wrist and JINS MEME glasses. The samping frequencies here are identical to the CogAge1 set except for the magnetometer. This was recorded with a frequency of 100Hz. The 61 activities were performed here by eight subjects 20 times each for 4 seconds. Like for with the first data set, only the data collected from the smartphone was used in the experiments in this thesis and the activities were reduced to the six mentioned above.

Due to the strong similarity of the CogAge1 and CogAge2 sets, they were combined into one large set (CogAge). For this, it was necessary to shorten the 5 seconds from the CogAge1 set to 4 seconds by just taking the first four seconds of a signal. Furthermore, the two additional sensors of the LG-G5 smartphone were not used. In addition, the frequency of the magnetometer from the CogAge2 set had to be adjusted to match the 50Hz of the CogAge1 data set. This was done with downsampling the magnetometer data from the CogAge2 dataset by a factor two. After these modifications, the data sets could be mixed without any problems.
In all of the following experiments, the CogAge dataset was used as the target and was divided into 50% for training and 50% for testing.

Human Activity Recognition Using Smartphones Dataset

The Human Activity Recognition Using Smartphones Dataset (UCI) (Anguita et al., 2013) data was collected using the 3D accelerometer and gyroscope of a Samsung Galaxy S II smartphone attached to the waist. Six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) were performed by 30 different subjects and recorded at a frequency of 50Hz.
In all transfer and contrastive learning experiments, the UCI dataset was used as the source. The data was divided into 70% for training and 30% for testing purpose.

The Wireless Sensor Data Mining Dataset

The wireless sensor data mining dataset (WISDM) (Weiss, Yoneda, and Hayajneh, 2019) was recorded using two devices: a smartphone and a smartwatch. For the data acquisition with the smartphone, three different models were used: a Nexus 5, Nexus 5X or Galaxy S6. Additionally, an LG G smartwatch was worn on the wrist. Data for the 3D accelerometer and gyroscope were collected with all devices. 51 subjects performed the 18 activities for three minutes each and the devices recorded the sensory data at a sampling rate of 20Hz.

subject	label	timestamp	x	y	z
1600	A	252207666810782	-0.36476135	8.793503	1.0550842
1600	A	252207717164786	-0.8797302	9.768784	1.0169983
1600	A	252207767518790	2.0014954	11.10907	2.619156
1600	A	252207817872794	0.45062256	12.651642	0.18455505
1600	A	252207868226798	-2.1643524	13.928436	-4.4224854

Table 4.1: Exampe rows from the WISDM dataset: Each file contains in each row the subject ID, the label, the timestamp and all three dimensions of the sensor data

The WISDM data are provided in 51 files. One file per subject. An example of the content of one files is shown in *table 4.1* The WISDM dataset was only used in the supervised contrastive learning experiment as additional source dataset. As for the UCI dataset the data was divided into 70% for training and 30% for testing purpose.

4.3 The Baseline Experiment

Experimental Setup and design choices

The purpose of this experiment was to find out what accuracy is achieved with an ordinary CNN only on the CogAge data. It was performed in two variants. Once with only two sensor modalities, the accerelometer and the gyroscope, and a second time with all five sensor modalities present in the CogAge data set.

Experimental implementation

Data pre-processing:
For this experiment the CogAge dataset is used, which contains of the two CogAge1 and CogAge2 sets. How the mixing of the two data sets (CogAge1 and CogAge2) is performed has already been explained in more detail in the dataset section 4.2.

Accelerometer and Gyroscope:
In this variant, the model is only trained for the accelerometer and gyroscope sensors. The CNN for classification consists of four convolutional layers with corresponding pooling layers. The dense and dropout layers follow.

```
Model: "har-classifier"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
All (InputLayer)             [(None, 800, 6, 1)]       0

All_conv1 (Conv2D)           (None, 700, 6, 8)         816
```

```
All_max1 (MaxPooling2D)      (None, 350, 6, 8)         0

All_conv2 (Conv2D)           (None, 350, 6, 8)         3272

All_max2 (MaxPooling2D)      (None, 175, 6, 8)         0

All_conv3 (Conv2D)           (None, 175, 6, 8)         3272

All_max3 (MaxPooling2D)      (None, 87, 6, 8)          0

All_conv4 (Conv2D)           (None, 87, 6, 8)          3272

All_max4 (AveragePooling2D)  (None, 43, 6, 8)          0

All_out (Flatten)            (None, 2064)              0

State_dense (Dense)          (None, 64)                132160

State_dropout (Dropout)      (None, 64)                0

state (Dense)                (None, 6)                 390

=================================================================
Total params: 143,182
Trainable params: 143,182
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.2 illustrates the har-classifier for two sensor modalities. For the CNN with all five sensors a multibranch CNN architecture is used with one branch processing the accelerometer and gyroscope data, another the linear accelerometer and gravimeter, and a third one the magnetometer.

```
Model: "har-classifier"
__________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================
All (InputLayer)             [(None, 800, 6, 1)] 0       []

Add (InputLayer)             [(None, 800, 6, 1)] 0       []

Magne (InputLayer)           [(None, 200, 3, 1)] 0       []

All_conv1 (Conv2D)           (None, 700, 6, 8)   816     ['All[0][0]']

Add_conv1 (Conv2D)           (None, 700, 6, 8)   816     ['Add[0][0]']

Magne_conv1 (Conv2D)         (None, 200, 3, 8)   416     ['Magne[0][0]']
```

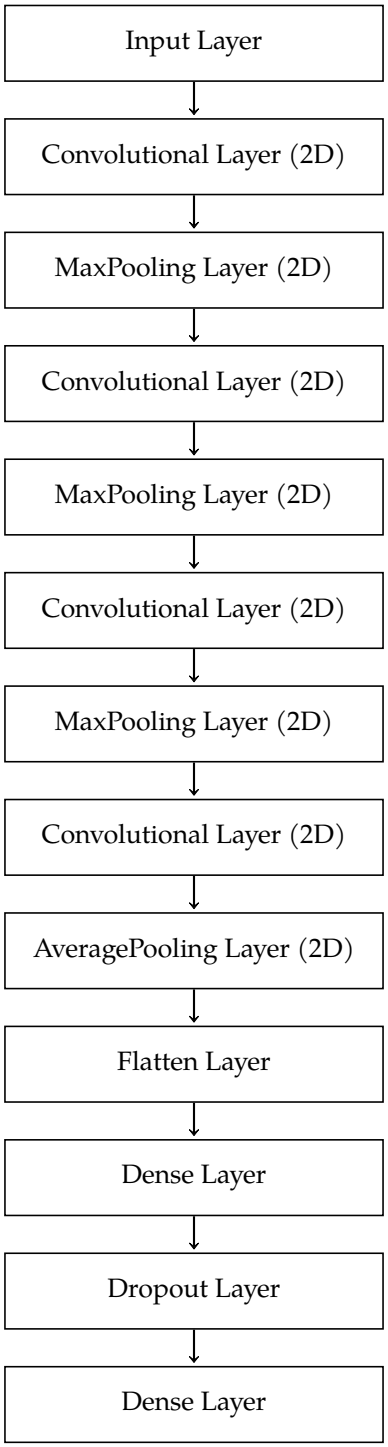

Figure 4.2: har-classifier for two sensor modalities: The accerelometer and gyroscope inputs lead to three repeating combinations of convolutional layers and max-pooling layers. This is followed by a convolutional layer and average-pooling combination, which are connected with a flatten layer followed by a fully-connected layer and a dropout layer. Finally, there is a fully-connected layer for classification.

```
All_max1 (MaxPooling2D)        (None, 350, 6, 8)   0       ['All_conv1[0][0]']

Add_max1 (MaxPooling2D)        (None, 350, 6, 8)   0       ['Add_conv1[0][0]']

Magne_max1 (MaxPooling2D)      (None, 100, 3, 8)   0       ['Magne_conv1[0][0]']

All_conv2 (Conv2D)             (None, 350, 6, 8)   3272    ['All_max1[0][0]']

Add_conv2 (Conv2D)             (None, 350, 6, 8)   3272    ['Add_max1[0][0]']

Magne_conv2 (Conv2D)           (None, 100, 3, 8)   1672    ['Magne_max1[0][0]']

All_max2 (MaxPooling2D)        (None, 175, 6, 8)   0       ['All_conv2[0][0]']

Add_max2 (MaxPooling2D)        (None, 175, 6, 8)   0       ['Add_conv2[0][0]']

Magne_max2 (MaxPooling2D)      (None, 50, 3, 8)    0       ['Magne_conv2[0][0]']

All_conv3 (Conv2D)             (None, 175, 6, 8)   3272    ['All_max2[0][0]']

Add_conv3 (Conv2D)             (None, 175, 6, 8)   3272    ['Add_max2[0][0]']

Magne_conv3 (Conv2D)           (None, 50, 3, 8)    1672    ['Magne_max2[0][0]']

All_max3 (MaxPooling2D)        (None, 87, 6, 8)    0       ['All_conv3[0][0]']

Add_max3 (MaxPooling2D)        (None, 87, 6, 8)    0       ['Add_conv3[0][0]']

Magne_max3 (MaxPooling2D)      (None, 25, 3, 8)    0       ['Magne_conv3[0][0]']

All_conv4 (Conv2D)             (None, 87, 6, 8)    3272    ['All_max3[0][0]']

Add_conv4 (Conv2D)             (None, 87, 6, 8)    3272    ['Add_max3[0][0]']

Magne_conv4 (Conv2D)           (None, 25, 3, 8)    1672    ['Magne_max3[0][0]']

All_max4 (AveragePooling2D)    (None, 43, 6, 8)    0       ['All_conv4[0][0]']

Add_max4 (AveragePooling2D)    (None, 43, 6, 8)    0       ['Add_conv4[0][0]']

Magne_max4 (AveragePooling2D) (None, 12, 3, 8)     0       ['Magne_conv4[0][0]']

All_out (Flatten)              (None, 2064)        0       ['All_max4[0][0]']

Add_out (Flatten)              (None, 2064)        0       ['Add_max4[0][0]']

Magne_out (Flatten)            (None, 288)         0       ['Magne_max4[0][0]']

CombFeat (Concatenate)         (None, 4416)        0       ['All_out[0][0]',
                                                            'Add_out[0][0]',
```

```
                                                    'Magne_out[0][0]']

State_dense (Dense)          (None, 64)          282688   ['CombFeat[0][0]']

State_dropout (Dropout)      (None, 64)          0        ['State_dense[0][0]']

state (Dense)                (None, 6)           390      ['State_dropout[0][0]']

==================================================================================
Total params: 309,774
Trainable params: 309,774
Non-trainable params: 0
__________________________________________________________________________________
```

Figure 4.3 illustrates the har-classifier for five sensor channels. Both networks are trained for 30 epochs each with a batch size of 64 and the Adam optimizer. Sparse categorical crossentropy was used as loss.

Results

The CNN with only one branch achieves 76.86% accuracy, while the network that uses data from all five sensors achieves 80.20% accuracy in classification.

4.4 Basic Transfer Learning

Experimental Setup and design choices

In this experiment, a CNN is firstly trained with the data from a source dataset and then fine tuned with the training data from the target dataset. The UCI set was used as the source dataset and the CogAge set as the target. The goal of the experiment was to evaluate whether the accuracy of a network trained only on the target data can be further increased by pretraining.

Experimental implementation

Data pre-processing:
In order to use the UCI dataset as a source, it had to be adapted to the target. First, the data had to be reshaped so that their arrangement was similar to that of the CogAge set. The sampling rate of the UCI set with 50Hz is much lower than the sampling rate of the CogAge set with 200Hz and had to be adjusted. Because of the large difference in frequencies, the data from the larger set was not downsampled in this case. Instead, the 50Hz was interpolated to a frequency of 200Hz by linear interpolation.

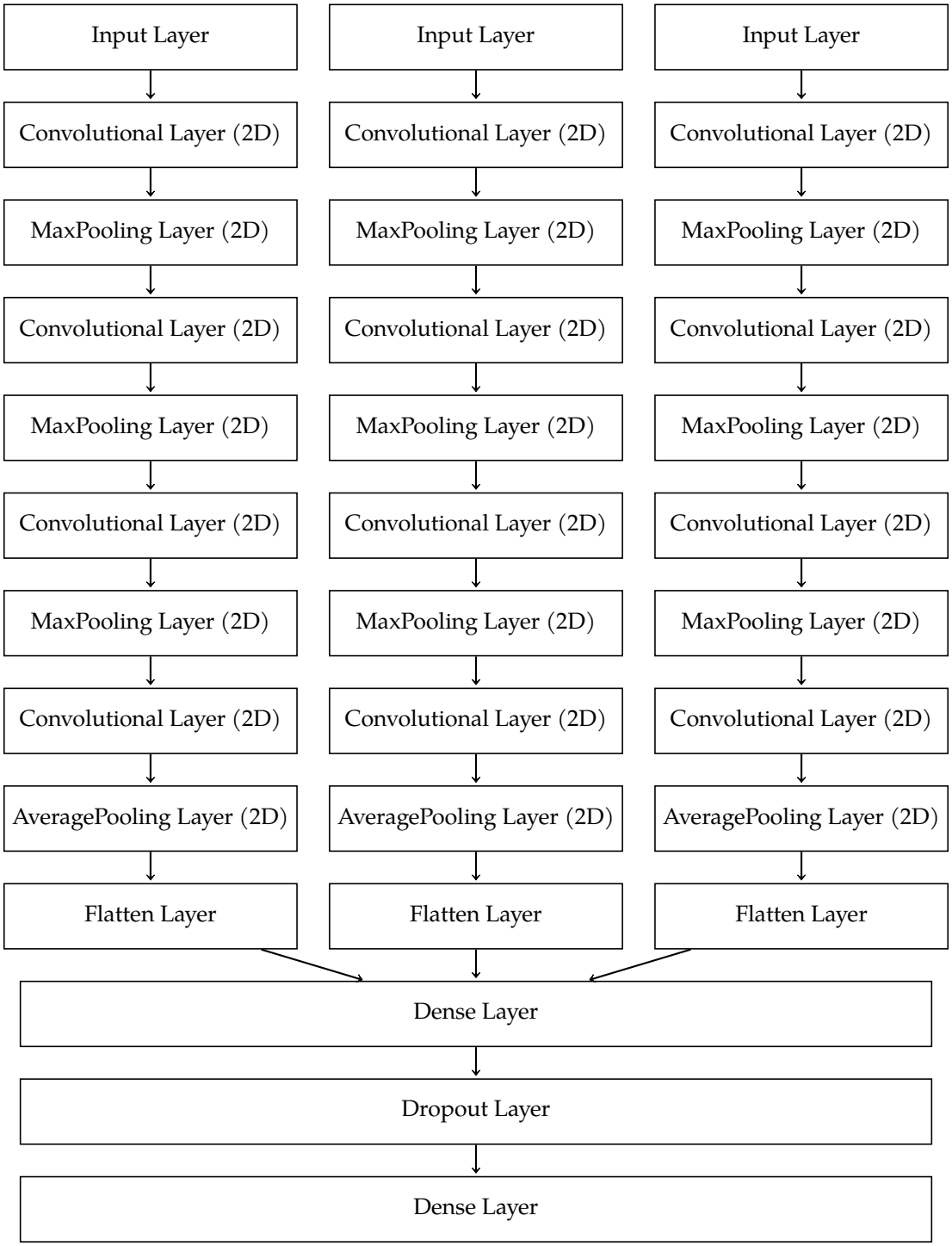

Figure 4.3: har-classifier for five sensor modalities: The model consists of three equally structured branches. One for the accelerometer and gyroscope, one for linear accelerometer and gravimeter, one for the magnetometer. Then all three branches are combined in a fully-connected layer. The dropout layer and the fully-connected layer for the classification follow.

Pretraining the model:
The UCI data set, in contrast to the CogAge data set, only contains data for the accelerometer and the gyroscope. For this reason, this experiment is performed only on these.
The model is the same as the one used for the baseline experiment with two sensor modalities, and consists of four convolutional layers together with their corresponding pooling layers. It is completed with dense and dropout layers:

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
All (InputLayer)             [(None, 800, 6, 1)]       0
_________________________________________________________________
All_conv1 (Conv2D)           (None, 700, 6, 8)         816
_________________________________________________________________
All_max1 (MaxPooling2D)      (None, 350, 6, 8)         0
_________________________________________________________________
All_conv2 (Conv2D)           (None, 350, 6, 8)         3272
_________________________________________________________________
All_max2 (MaxPooling2D)      (None, 175, 6, 8)         0
_________________________________________________________________
All_conv3 (Conv2D)           (None, 175, 6, 8)         3272
_________________________________________________________________
All_max3 (MaxPooling2D)      (None, 87, 6, 8)          0
_________________________________________________________________
All_conv4 (Conv2D)           (None, 87, 6, 8)          3272
_________________________________________________________________
All_max4 (AveragePooling2D)  (None, 43, 6, 8)          0
_________________________________________________________________
All_out (Flatten)            (None, 2064)              0
_________________________________________________________________
State_dense (Dense)          (None, 64)                132160
_________________________________________________________________
State_dropout (Dropout)      (None, 64)                0
_________________________________________________________________
state (Dense)                (None, 6)                 390
=================================================================
Total params: 143,182
Trainable params: 143,182
Non-trainable params: 0
_________________________________________________________________
```

The model was then trained on the source data for 150 epochs using the Adam optimizer and a batch size of 64. The categorical cross entropy loss was used as the loss.
Afterwards the model obtained an accuracy of 89.21% for the source task. Then the trained model was exported for fine-tuning.

Fine-tuning the model:

Afterwards the model was fine-tuned for the target task with the help of the CogAge data set. The model already trained on the source data was imported. It was then compiled and optimized in the same way on the target data as in the pre-training with the source data.

Results

In the validation, the trained model achieved an accuracy of 64.63% on the test data of the targeted dataset. Compared to the results of the baseline experiment, in which an accuracy of 76.86% was obtained for the model with two sensor modalities, it can be seen that the accuracy drops by about 12%.
It can be seen that pre-training on a completely different dataset results in the classification on the original dataset degrading significantly. For this reason, this strategy seems to be unsuitable for implementation.

4.5 Contrastive Learning

In this experiment, it was determined what classification accuracy a supervised contrastive learning model achieves when trained only on the target data (the CogAge set). The results could thus be compared later with the results trained with the addition of further data sets. This experiment is refered as *Contrastive Learning I* in the following sections.

Experimental implementation

Generating positive pairs:
In advance of the actual training, variations of the labeled data had to be created to be divided into related groups as described in the paper (Khosla et al., 2020). To do this on time-series segments, I used the window warping method described in the paper (Le Guennec, Malinowski, and Tavenard, 2016). Here two new modified segments are generated from each segment. A random window of size 80 (10% of the length of the segment) is selected from the original segment and the contents of the window are stretched once to 160 data points and shortened to 40 points for the second output segment. To make sure the two new segments have the same length as the original one, the shortened segment is filled with the last data points of the original segment, and the stretched segment is cut off at the end.

Pretraining the model:
First, the model of the encoder was created. For the encoder, the same architecture as the one used in the baseline experiments (with two sensor modalities) was used, minus the final classification layers. Since the encoder will later be fed only with the data from two sensors, the encoder only processes the data from the accelerelometer and the gyroscope in this experiment as well.

The model of the encoder is provided as follows:

```
Model: "hac-encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
All (InputLayer)             [(None, 800, 6, 1)]       0

All_conv1 (Conv2D)           (None, 700, 6, 8)         816

All_max1 (MaxPooling2D)      (None, 350, 6, 8)         0

All_conv2 (Conv2D)           (None, 350, 6, 8)         3272

All_max2 (MaxPooling2D)      (None, 175, 6, 8)         0

All_conv3 (Conv2D)           (None, 175, 6, 8)         3272

All_max3 (MaxPooling2D)      (None, 87, 6, 8)          0

All_conv4 (Conv2D)           (None, 87, 6, 8)          3272

All_max4 (AveragePooling2D)  (None, 43, 6, 8)          0

All_out (Flatten)            (None, 2064)              0

=================================================================
Total params: 10,632
Trainable params: 10,632
Non-trainable params: 0
```

A projection head was added to the encoder for the training phase as described in the associated paper (Khosla et al., 2020), so that it can be trained independently. This is illustrated in **figure 4.4**.

```
Model: "har-encoder_with_projection-head"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
All (InputLayer)             [(None, 800, 6, 1)]       0

hac-encoder (Functional)     (None, 2064)              10632

state (Dense)                (None, 6)                 12390

=================================================================
Total params: 23,022
Trainable params: 23,022
Non-trainable params: 0
_________________________________________________________________
```

The training of the encoder was performed using the CogAge data as source in 30 epochs and a batch size of 64 with a supervised contrastive loss and the Adam optimizier. Subsequently, the resulting weights were stored.

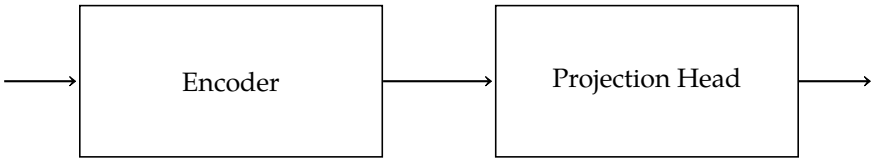

Figure 4.4: Encoder with projection head: To train the encoder independently it must be connected with a projection head.

Fine-tuning the model:
A multibranch CNN architecture similar as the one in the baseline with 5 sensor modalities is used on the target domain. Since the source dataset only contain data from the 3D accelerometer and gyroscope, a transfer is only performed for the branch processing this type of data. The two branches processing the three sensor modalities not present in the source dataset (linear accelerometer, gravimeter and magnetometer) are not transferred.

```
Model: "additional-layers"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Add (InputLayer)             [(None, 800, 6, 1)]       0

Add_conv1 (Conv2D)           (None, 700, 6, 8)         816

Add_max1 (MaxPooling2D)      (None, 350, 6, 8)         0

Add_conv2 (Conv2D)           (None, 350, 6, 8)         3272

Add_max2 (MaxPooling2D)      (None, 175, 6, 8)         0

Add_conv3 (Conv2D)           (None, 175, 6, 8)         3272

Add_max3 (MaxPooling2D)      (None, 87, 6, 8)          0

Add_conv4 (Conv2D)           (None, 87, 6, 8)          3272

Add_max4 (AveragePooling2D)  (None, 43, 6, 8)          0

Add_out (Flatten)            (None, 2064)              0

=================================================================
Total params: 10,632
Trainable params: 10,632
Non-trainable params: 0
_________________________________________________________________
```

The magnetometer has a different frequency and for this reason needs its own branch:

```
Model: "magne-layers"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Magne (InputLayer)           [(None, 200, 3, 1)]       0

Magne_conv1 (Conv2D)         (None, 200, 3, 8)         416

Magne_max1 (MaxPooling2D)    (None, 100, 3, 8)         0

Magne_conv2 (Conv2D)         (None, 100, 3, 8)         1672

Magne_max2 (MaxPooling2D)    (None, 50, 3, 8)          0

Magne_conv3 (Conv2D)         (None, 50, 3, 8)          1672

Magne_max3 (MaxPooling2D)    (None, 25, 3, 8)          0

Magne_conv4 (Conv2D)         (None, 25, 3, 8)          1672

Magne_max4 (AveragePooling2  (None, 12, 3, 8)          0
D)

Magne_out (Flatten)          (None, 288)               0

=================================================================
Total params: 5,432
Trainable params: 5,432
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.5 shows the encoder transferred to the model.
The model was trained for 30 epochs using the Adam optimizer and Sparse Categorical Crossentropy as loss. A batch size of 64 was used.

Results

With 84.28% accuracy, the model achieved a promising classification rate on the CogAge validation data. From now on, this value will serve as a basis to determine whether training the encoder with other data will have a positive effect on the model.

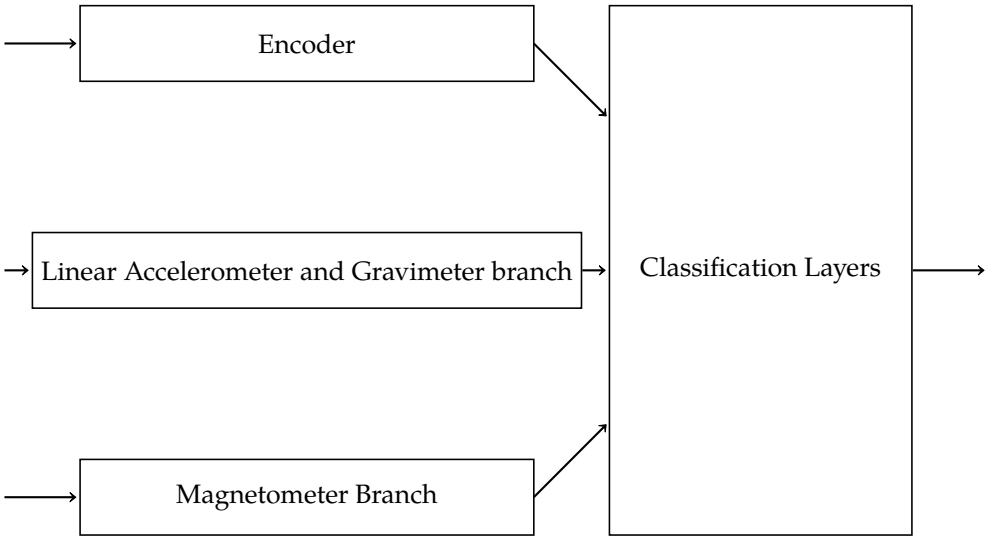

Figure 4.5: Embedded encoder: After the encoder has been trained with the projection head, it is removed. The encoder now is a branch in the multibranch architecture.

4.6 Contrastive Learning with the UCI dataset

Experimental Setup and design choices

The goal of this experiment was to check if it is possible to further increase the accuracy of the classification of the CogAge data as target domain by pre-training the encoder with the UCI data as source. After pre-training the encoder, the latter was inserted into the same multibranch architecture as before. This model was then trained with the CogAge data for fine-tuning.
This experiment is refered as *Contrastive Learning II* in the following sections.

Experimental implementation

Data pre-processing:
Since the format of the UCI dataset differs significantly from the CogAge dataset as already shown in the description of the datasets, some work is needed to first align the data. Since the segments from the UCI data set, in contrast to the CogAge data set, only have a frequency of 50Hz, they first had to be interpolated to the frequency of 200Hz. A simple linear interpolation was used for this purpose. Afterwards segments were prepared in two different ways: Once the segment was used in its interpolated form, in another variant uniform distributed noise was added to it. Here I wanted to check whether adding noise has an influence on interpolated segments in supervised contrastive learning.
Subsequently, as in the experiment before, a generation of positive pairs took place, which were to be used for the pretraining of the encoder.

Pretraining the Model:

As in the previous experiment a projection head was added to the encoder model for pre-training.
In this way, the encoder was trained with the prepared data once in the noise variant, once without noise, for 30 epochs. Subsequently, the encoder was transferred to the complete model. The complete model including the encoder was then trained and validated with the CogAge data (as already explained in the last experiment).

Results

After validating the model using the CogAge test data, it can be seen that the accuracy drops slightly both in the case with noise and in the case without noise compared to the case that only used the CogAge dataset for the source and target. The variant without noise achieved an accuracy of 82.97% and the variant with noise an accuracy of 82.68%. This difference of about 2% is small, but visible. Nevertheless, this result is by far better than the result of the basic transfer learning experiment and slightly outperforms the baseline experiment. It is also interesting to observe that there is almost no difference in accuracy between the variant with noise and the variant without noise.

4.7 Contrastive Learning with the UCI dataset and WISDM dataset

Experimental Setup and design choices

In order to check whether the result could be improved again using further data sets or whether it degrades further, an experiment was carried out here in which an additional data set was mixed with the UCI data on the source domain. The WISDM data set described on page 11 was used for this purpose. The new mixed dataset was then used to train the encoder as before, which was used in the full model afterwards. As before, this experiment was also performed both with noise and without noise.
This experiment is refered as *Contrastive Learning III* in following sections.

Experimental implementation

First, the same modifications as from the previous Contrastive Learning experiment were performed for the UCI dataset. Afterwards the WISDM data had to be aligned with the UCI data so that these two data sets could be mixed. As in the UCI set the WISDM set contains the data of accelerometer and gyroscope sensors.
The WISDM data is sorted into files by subject which had to be combined. The activities were recorded in three minute intervals per activity and had a sampling rate of 20 Hz. The 3 minutes had to be segmented into 4 second blocks and the correct labels had to be applied. Due to the lower sampling rate, an interpolation to the 200Hz of the target data was necessary. After aligning the two datasets they were shuffled and permutation of the

data and labels in unison was applied to make sure that the matching between associated examples and labels is not lost during the mixing operation.
From this large data set, two variations were created as in the experiment before: One with noise, one without noise.
The following steps of this experiment take place from the *Pretraining the Model* step as in the last experiment.

Results

Reaching an accuracy of 81.95% without noise and 82.39% with in this experiment, it can be observed that the addition of an additional source data set has no significant effect on the accuracy. The encoder now trained with the larger data set delivers very exactly the same results as an encoder trained only with the UCI data set when transferred to the target model. Thus, it slightly worsens the accuracy compared to the model trained only on the CogAge data.

5

Discussion

As seen in *Table 5.1* the experiments have shown that it is not a simple problem to create neural networks that can handle and correctly interpret the large variety of data coming from different sensors on mobile devices. Besides sampling rates, different sensors seem to have other characteristics in recording signals that make it difficult to obtain the same results for the same activities but different smartphones.
Furthermore, it was shown that the use of different publicly available data sets requires a significant amount of preprocessing to align them. This is due, among other things, to the fact that CNNs accept fixed size inputs, so the data shape must be adapted to fit to them for all datasets. In this context, it was also shown that adding noise to sensory data in preprocessing does not significantly affect the later classification in the supervisied contrastive learning approach.
It is interesting to see that the contrastive learning approach using only the target dataset with 84.28% achieved higher accuracy than the baseline experiment (using the data from all five sensors) with 80.2%. It achieved the highest result among all the experiments conducted in this thesis. This suggests that the supervised contrastive learning approach can also be good to improve the performances of time-series classification. Even though the contrastive learning approach achieved significantly better results than the basic transfer learning approach, using other human activity recognition datasets led to no improvement in classification compared to the contrastive learning experiment in which only the

Method	Source dataset	Target dataset	Accuracy
Baseline 2 Sensors	none	CogAge	76.86%
Baseline 5 Sensors	none	CogAge	80.20%
Basic Transfer Learning	UCI	CogAge	64.63%
Contrastive Learning I	CogAge	CogAge	84.28%
Contrastive Learning II with noise	UCI	CogAge	82.68%
Contrastive Learning II without noise	UCI	CogAge	82.97%
Contrastive Learning III with noise	UCI + WISDM	CogAge	82.39%
Contrastive Learning III without noise	UCI + WISDM	CogAge	81.95%

Table 5.1: Classification results for the recognition of Bending, Lying, Sitting, Squatting, Standing, Walking on the CogAge dataset.

CogAge dataset was used. Furthermore, it was also observed that the approach of using more data sets and thus increasing the underlying data set did not improve the accuracy with this type of model.
The reasons for this could be manifold. For example, it is possible that the window warping strategy for positive pair generation does not perform well in supervised contrastive learning. Positive pair generation for time intervals is still an area where there is much potential for further development. It is also conceivable that there may be even better methods for standadizing the different data sets, possibly involving more extensive preprocessing and/or the application of filters. On a more general level, it could also be that the selected datasets are too different in significant areas to base them on. The features learned might be too specific to the source dataset, lowering performance on the target. Also, the choice of hyperparamters, batch size, or number of epochs could cause the results to be unpromising.

6

Conclusion and Future Work

As the Covid19 pandemic continues, the utility of contact tracking remains an interesting and relevant topic, especially since it is not inconceivable that other pandemics of this kind might happen in the future. The benefit of activity classification would take us another step in a direction that would allow us to optimally identify infected individuals while reducing false positives and thus living more safely and comfortably in this and future pandemics. Since a contact tracking application should be able to be used by as many people as possible, the heterogeneity of different smartphones is the main problem to achieve a reliable classification of activities.
In this thesis, I investigated how much the heterogenity affects classification when using an ordinary transfer learning approach. Furthermore, I investigated an approach to improve classification that relies on supervised contrastive learning. It turned out that the ordinary transfer learning approach leads to a significant degradation of accuracy. The supervised contrastive learning approach also leads to a degradation of classification compared to a supervised contrastive learning model trained on a single dataset. However, the loss of accuracy with 2% is significantly smaller than with ordinary transfer learning. Furthermore, the supervised contrastive learning model trained on a single dataset achieved a higher accuracy than the baseline CNN with the same dataset.
However, it was not examined in the context of this work whether further methods for the positive pair generation possibly lead to better results in the classification. Also, only linear interpolation was used in the preprocessing to adjust different sampling rates. In the future, it would be interesting to investigate whether there are better methods of preprocessing to more closely match the features of the datasets. It is conceivable that different filtering methods would also have an impact. Investigating further data augmentation techniques to create positive pairs would also be a direction worth looking into. It would also be interesting to investigate other transfer learning methods, such as self-supervised contrastive learning, even though the literature suggests that the supervised contrastive learning approach outperforms the self-supervised learning approach (Khosla et al., 2020).
Due to the variety of data used for training, it is conceivable that the trained models could be used to draw conclusions about subjects upon closer analysis. Training the model in a privacy preserving way, so that it does not leak sensitive information about the training data, would also be an important point to investigate.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

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, pp. 437–442.

Bulling, A., Blanke, U., and Schiele, B. (2014). A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors. ACM Comput. Surv. *46*. ISSN: 0360-0300. DOI: 10.1145/2499621. URL: https://doi.org/10.1145/2499621.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G.E. (2020). A Simple Framework for Contrastive Learning of Visual Representations. CoRR *abs/2002.05709*. arXiv: 2002.05709. URL: https://arxiv.org/abs/2002.05709.

Harris, C.R., Millman, K.J., Walt, S.J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., et al. (Sept. 2020). Array programming with NumPy. Nature *585*, 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering *9*, 90–95. DOI: 10.1109/MCSE.2007.55.

Imort, T. (2022). *Automated Activation of the Acoustic Scene Classification in the Corona-Warn-App*. [Online; accessed 2022-11-16]. URL: https://www.its.uni-luebeck.de/fileadmin/files/theses/Timothy_Imort_Bachelorarbeit_.pdf.

Inc., A. (n.d.). *Google Colaboratory*. [Online; accessed 2022-11-16]. URL: https://colab.research.google.com/.

Jupyter, P. (2014). *Project Jupyter*. [Online; accessed 2022-11-16]. URL: https://jupyter.org/.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised Contrastive Learning. CoRR *abs/2004.11362*. arXiv: 2004.11362. URL: https://arxiv.org/abs/2004.11362.

Le Guennec, A., Malinowski, S., and Tavenard, R. (2016). Data augmentation for time series classification using convolutional neural networks. In ECML/PKDD workshop on advanced analytics and learning on temporal data,

Li, F., Shirahama, K., Nisar, M.A., Huang, X., and Grzegorzek, M. (2020). Deep Transfer Learning for Time Series Data Based on Sensor Modality Classification. Sensors *20*. ISSN: 1424-8220. DOI: 10.3390/s20154271. URL: https://www.mdpi.com/1424-8220/20/15/4271.

Li, F., Shirahama, K., Nisar, M.A., Köping, L., and Grzegorzek, M. (2018). Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. Sensors *18*. ISSN: 1424-8220. DOI: 10.3390/s18020679. URL: https://www.mdpi.com/1424-8220/18/2/679.

Liebenow, J. (2021). *Extending Contact Tracing with Contact Contexts*. [Online; accessed 2022-11-16]. URL: https://mohammadi.eu/dateien/contact_contexts.pdf.

Liebenow, J. (2022). *Differentially Private Aggregation of Mobility Data*. [Online; accessed 2022-11-16]. URL: https://www.its.uni-luebeck.de/fileadmin/files/theses/master_thesis_liebenow_dp_aggregation_of_mobility_data.pdf.

Nisar, M.A., Shirahama, K., Li, F., Huang, X., and Grzegorzek, M. (2020). Rank Pooling Approach for Wearable Sensor-Based ADLs Recognition. Sensors *20*. ISSN: 1424-8220. DOI: 10.3390/s20123463. URL: https://www.mdpi.com/1424-8220/20/12/3463.

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

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research *12*, 2825–2830.

Salama, K. (2020). *Supervised Contrastive Learning*. [Online; accessed 2022-11-16]. URL: https://github.com/keras-team/keras-io/blob/master/examples/vision/supervised-contrastive-learning.py.

SIG-addons (n.d.). *TensorFlow SIG Addons*. [Online; accessed 2022-11-16]. URL: https://www.tensorflow.org/addons.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T.S., Kjærgaard, M.B., Dey, A., Sonne, T., and Jensen, M.M. (2015). Smart Devices Are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys '15. Seoul, South Korea: Association for Computing Machinery, pp. 127–140. ISBN: 9781450336314. DOI: 10.1145/2809695.2809718. URL: https://doi.org/10.1145/2809695.2809718.

Weiss, G.M., Yoneda, K., and Hayajneh, T. (2019). Smartphone and smartwatch-based biometrics using activities of daily living. IEEE Access *7*, 133190–133202.

Zeiler, M.D. and Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. CoRR *abs/1311.2901*. arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.