Fully convolutional networks for segmentation of wearable sensor data

Mihail Douhaniaris

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 30.9.2019

Supervisor

Assistant prof. Fabricio Oliveira

Advisor

Dr Vinay Prabhu

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Mihail Douhaniaris

Title Fully convolutional networks for segmentation of wearable sensor data

Degree programme Mathematics and Operations Research

Major Systems and Operations Research **Code of major** SCI3055

Supervisor Assistant prof. Fabricio Oliveira

Advisor Dr Vinay Prabhu

Date 30.9.2019 **Number of pages** 39 **Language** English

Abstract

Wearable sensor-based activity recognition is an area of pattern recognition dealing with detecting the actions and activities of a human agent using motion sensors worn or incorporated on the agent's body. Current state-of-the-art methods use a sliding window approach to partition the sensor signals into smaller fixed size windows which are then classified into activities by a deep neural network. However, this arbitrary partitioning gives rise to the so-called multiclass window problem where the partitioned window contains data from multiple activities which can result in incorrect classifications and missed activities, particularly when trying to detect sporadic activities which are irregularly occurring and short in duration.

In this work, we propose a novel approach for wearable sensor-based activity recognition based on fully convolutional networks. Our model, which we coin *SensorFCN*, automatically segments a motion sensor signal stream of arbitrary length into activities by dense samplewise classification. Because of this, our approach is unaffected by the multiclass window problem and the model can precisely detect sporadic activities. A public activity recognition dataset consisting of motion and orientation sensor data originating from 67 subjects performing 11 different activities is used to train and validate the proposed model. A series of experiments is performed in order to find the optimal model configuration which achieves an overall sample accuracy of 96.6%. In addition, we perform occlusion sensitivity analysis in order to qualitatively measure our model's robustness to perturbations in the input signals and to better understand the model's predictions. Our work shows that fully convolutional networks are a promising family of models for wearable sensor-based activity recognition and are effective in overcoming the multiclass window problem.

Keywords fully convolutional networks, wearable sensors, activity recognition

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Mihail Douhaniaris

Työn nimi Puettavien anturien datan osiointi täysikonvolutiivisten verkkojen avulla

Koulutusohjelma Mathematics and Operations Research

Pääaine Systems and Operations Research **Pääaineen koodi** SCI3055

Työn valvoja Apulaisprof. Fabricio Oliveira

Työn ohjaaja TkT Vinay Prabhu

Päivämäärä 30.9.2019 **Sivumäärä** 39 **Kieli** Englanti

Tiivistelmä

Puettaviin antureihin pohjautuva toimien tunnistus on hahmontunnistuksen osa-alue, jonka tavoitteena on tunnistaa ihmisen toimia ihmiskehoon kiinnitettävien liikeanturien avulla. Nykyiset menetelmät kirjallisuudessa hyödyntävät liukuvan ikkunan menetelmää pitkän anturisignaalin osioinnissa pienempiin kiinteän kokoisiin ikkunoihin, jotka luokitellaan toimiin syvän neuroverkon avulla. Tämä mielivaltaisen signaalin osiointi on kuitenkin ongelmallinen, sillä osioitu ikkuna voi sisältää useammasta kuin yhdestä toimesta peräisin olevia signaaleja. Tällöin nykyiset menetelmät eivät pysty luokittelemaan oikein kaikkia ikkunan sisältämiä toimia, mikä voi johtaa vääriin luokitteluihin ja menetettyihin toimiin, erityisesti kun yritetään havaita lyhyitä, epäsäännöllisiä toimia.

Tässä työssä esitellään uusi, täysikonvolutiivisiin verkkoihin pohjautuva menetelmä toimien tunnistamiseen. Esittelemämme malli osioi mielivaltaisen pitkän anturisignaalin automaattisesti pienempiin eri toimia sisältäviin osiin, mikä mahdollistaa lyhyidenkin toimien tarkan tunnistuksen. Tämän ansiosta lähestymistapamme ei kärsi moni-ikkunaongelmasta, ja mallilla voi havaita tarkasti myös epäsäännöllisiä toimia. Käytämme mallin opettamiseen julkista, liike- ja suunta-antureista koostuvaa tietuekokoelmaa, joka on kerätty 67:än koehenkilön suorittamista 11:sta eri toimesta. Koesarjojen tuloksena olemme löytäneet 96,6 %:n mittaustarkkuuden tuottavan optimaalisen mallikonfiguraation. Lisäksi olemme suorittaneet peiteherkkyysanalyysin mitataksemme mallin suoriutumista syötesignaalien häiriöistä sekä ymmärtääksemme paremmin mallin ennustamia arvoja. Tämän työn tulokset osoittavat täysikonvolutiivisiin verkkoihin pohjautuvien menetelmien pystyvän ohittamaan muiden antureihin pohjautuvien liikkeentunnistusmallien pullonkaulana pidettävän moni-ikkunaongelman, mikä on merkittävä edistysaskel antureihin pohjautuvan liikkeentunnistuksen alalla.

Avainsanat täysikonvolutiivinen verkko, puettavat anturit, toimien tunnistaminen

Preface

This work was carried out as a research project during my time at UnifyID. This thesis would not have been possible without the help and support of multiple people whom I feel absolutely privileged to have met and be part of this journey. Firstly, I would like to thank John and Kurt for allowing me to be part of such an exciting startup, as well as all the other folks at UnifyID for making it such a fun and inspiring place to work at. Secondly, I would like to thank my advisor Vinay for proposing this project and for being so supportive from day one. I would also like to thank the entire machine learning team at UnifyID for always being willing to discuss and exchange ideas with me. Thirdly, I would like to thank Assistant Professor Fabricio Oliveira for his supervision and for going out of his way to help me graduate on time despite the tight timeline surrounding this project. Finally, I would like to thank my family for always being there for me, and my girlfriend Michelle for all her encouragement and patience during this endeavor.

San Francisco, 19.9.2019

Mihail Douhaniaris

Contents

Symbols, operators and abbreviations

Symbols

f^*	ground truth function
f	artificial neural network
y	ground truth target variable
$\hat{y}$	predicted target variable
$\boldsymbol{x}$	data vector
$\boldsymbol{\theta}$	parameter vector
h	artificial neuron
$\boldsymbol{w}$	weight parameter vector
b	bias parameter
p	probability distribution
J	cost function
L	loss function
ϵ	learning rate
κ	kernel size
n_{filter}	SensorFCN filter parameter
κ_{smooth}	SensorFCN smoothing kernel size parameter

Operators

$\star$	cross-correlation
$\nabla_{\boldsymbol{\theta}}$	gradient with respect to parameter vector $\boldsymbol{\theta}$
$\sum_i$	sum over index i
$\mathbb{E}$	expected value

Abbreviations

GPS	Global Positioning System
GPU	graphics processing unit
IoU	intersection over union
RFID	radio-frequency identification
ReLU	rectified linear unit

1 Introduction

Activity recognition is an area of pattern recognition dealing with predicting a human agent's actions based on sensor data. The types of actions and used sensors differ based on the underlying application. For example, in a fitness tracking application motion and GPS sensors might be used to detect different exercises, such as running or cycling, and in a smart hospital application environmental sensors, such as depth cameras and RFID sensors, might be used to detect the movements and actions of the hospital staff [1, 2]. The emergence of ubiquitous smart devices worn directly on the body (e.g. smartwatches) or incorporated into clothing (e.g. a smartphone placed in a pants' pocket) has enabled a cost effective and non-obstructive way of activity recognition, called wearable sensor-based activity recognition [3–5]. In this type of activity recognition, data from motion and orientation sensors such as accelerometers, gyroscopes and magnetometers is used to predict the activities and gestures of the device's user and the applications range from fitness tracking and assisted living to human-computer interaction.

Wearable sensor-based activity recognition has been an area of active research for over a decade and it falls under the broader category of time series classification [6]. Earlier approaches were mostly based on manual extraction of features from the sensor signals using signal processing methods which were then classified into activities using statistical and classical machine learning methods, such as logistic regression, decision trees and support vector machines [6–8]. A significant drawback to this approach is that manual feature extraction is a laborious task, requiring domain expertise both in signal processing and in the application domain, and that the engineered features are usually specific to the application domain and the used sensor modalities. More recently, advances in deep learning and GPU computing frameworks have sparked the interest of activity recognition researchers [9–14]. Deep learning models automatically learn powerful features from the raw signals based on example activity data, therefore completely circumventing the need for manual feature extraction. Additionally, deep learning based activity recognition methods have been shown to consistently outperform traditional activity recognition methods, making them an attractive alternative to their predecessor methods [4, 5]. Most deep learning based activity recognition methods found in literature utilize a sliding window preprocessing step where the raw sensor signals are divided into smaller subsequences of fixed length, called windows, which are then classified into a single activity. This preprocessing step, however, is prone to the so-called multiclass window problem where a single window contains data from more than one activity, which can result in misclassifications and missed activities. In particular, sporadic activities which are usually short in duration and randomly occurring in the motion sensor signals are especially prone to this type of error.

A promising alternative to sliding window based deep learning models are semantic segmentation models which automatically detect segments of the input data belonging to each class [15]. Originally developed for computer vision, this family of models has not been fully explored for wearable sensor-based activity recognition. In this work, we propose a novel segmentation model for wearable sensor data based on fully

convolutional networks which we coin *SensorFCN*. Our proposed model can process a sensor data stream of arbitrary length and outputs a dense activity segmentation map where each input data sample is mapped to an activity. Because of this, the proposed model is not affected by the multiclass window problem and can precisely detect sporadic activities. In the experimental part of this thesis, we use a public activity recognition dataset to train SensorFCN models with varying model parameters in order to assess the effect of the parameters on activity recognition performance and to find the best performing parameter values. After finding the optimal model configuration, the generalization performance of the model is evaluated on unseen agents using two inference approaches, one using presegmented signal windows of fixed length and one using the full sensor signals of each data collection session. Finally, occlusion sensitivity experiments are carried out for different activities in order to evaluate the robustness of the model to perturbations in the signal inputs and to help better explain the model predictions.

In the following chapter, we give a brief introduction to artificial neural networks, explain what fully convolutional networks are and provide a more detailed overview of activity recognition. In the third chapter, we present the used dataset and the developed SensorFCN model architecture, as well as explain our experimentation procedure. The fourth chapter contains the experiment results and in the final chapter we summarize and discuss this thesis and its main results.

2 Background

2.1 Artificial neural networks

An artificial neural network is a parametric model used to computationally approximate a function f^* that is usually highly nonlinear in nature [16]. For example, in a classification setting where the task is to assign a categorical variable (class) y to a vector $\boldsymbol{x}$ (e.g. a time series or an image) the function of interest would be $y = f^*(\boldsymbol{x})$. An artificial neural network defines the mapping $y = f(\boldsymbol{x}; \boldsymbol{\theta})$ with respect to some parameters $\boldsymbol{\theta}$ that are learned from data.

The basic building block of an artificial neural network is a computational unit called the artificial neuron [17]. An artificial neuron h takes a vector input $\boldsymbol{x}$ and outputs a scalar value $h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = w_1x_1 + \cdots + w_nx_n + b$, where $\boldsymbol{w}$ and b are the neuron weight and bias parameters and $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^n$. An artificial neuron h that takes a 5-dimensional input vector $\boldsymbol{x}$ is depicted in Figure 1.

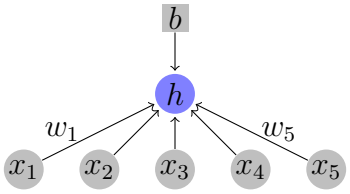

Figure 1: An artificial neuron.

Artificial neural networks are organized in a hierarchically layered structure so that layer outputs are propagated as inputs to deeper layers. This is called the forward pass and it is the principle used to obtain predictions for an input vector $\boldsymbol{x}$ [16]. For example, $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$ represents a network f of depth 3 where $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ are the first, second and third network layers, respectively. The layer that directly processes the input data is called the input layer, the layer that produces the predictions is called the output layer and all intermediate layers are called the hidden layers of the network. In the aforementioned example, $f^{(1)}$ is the input layer, $f^{(2)}$ the hidden layer and $f^{(3)}$ the output layer. Each layer consists of a set of artificial neurons that process prior layer outputs in parallel. Figure 2 shows a fully connected layer that consists of three units h_1, h_2 and h_3, each one with its own set of parameters $\{\boldsymbol{w_1}, b_1\}$, $\{\boldsymbol{w_2}, b_2\}$ and $\{\boldsymbol{w_3}, b_3\}$, respectively. This layer type is called fully connected because each neuron in the layer is path connected via inner product to the full output of its parent layer [16].

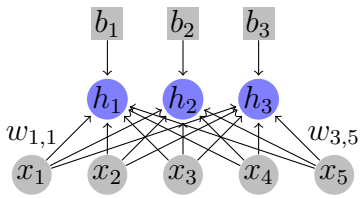

Figure 2: A fully connected layer with three units.

Artificial neural networks constructed purely from fully connected layers are called multilayer perceptrons and are the quintessential deep learning models [16]. Multilayer perceptrons have been responsible for many successes in deep learning research and they form the basis for all modern deep learning models. However, multilayer perceptrons have several shortcomings that hinder their performance in many real world problems. Firstly, fully connected layers always expect a fixed input size and are not invariant to input transformations, such as translation or rotation [17]. For example, in an image classification setting the class prediction should not change when the objects in the image are shifted or rotated. Secondly, the number of learnable parameters in a multilayer perceptron becomes extremely large with large inputs and deep networks which makes training these networks statistically inefficient and computationally expensive.

Convolutional networks are specialized artificial neural networks that exploit the topology of the input data to encode data properties directly to the network [16]. Convolutional networks address the many shortcomings of multilayer perceptrons and have been widely successful in many deep learning tasks, such as image classification, semantic segmentation and speech recognition [17]. While multilayer perceptrons connect a layer's units to every unit of it's parent layer, convolutional networks utilize sparse local connections that only connect units to their topologically close input region. Additionally, in contrast to multilayer perceptrons where each hidden unit posesses its own set of parameters, convolutional networks utilize parameter sharing which refers to using the same weight and bias parameters across a convolutional layer. These two properties, namely sparse local connections and parameter sharing, greatly reduce the number of learnable parameters in the network which results in reduced memory consumption, as well as increased statistical and computational efficiency. Parameter sharing also achieves translation invariance making convolutional networks robust to transformed inputs. Figure 3 shows a convolutional layer with three hidden units h_1, h_2 and h_3 that all share the weight and bias parameters, $\boldsymbol{w}$ and b. In the context of convolutional networks, the weight parameter is often called the kernel, the number of local connections the kernel size and the input region that is path connected to a hidden unit is called the unit's receptive field. Hidden units in a convolutional layer can be formulated as $h_i = (\boldsymbol{w} \star \boldsymbol{x})_i + b$, where $(\boldsymbol{w} \star \boldsymbol{x})_t = \sum_{k=0}^{\kappa-1} w_{k+1} x_{t+k}$ and κ is the kernel size. It is important to note that as the network depth increases the effective receptive field becomes progressively larger, as shown in Figure 4 [16].

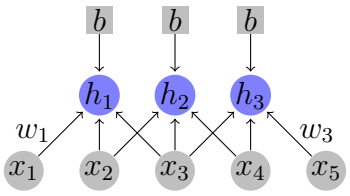

Figure 3: A convolutional layer with a kernel size of 3 units.

In practice, when designing convolutional networks the number of units in a convolutional layer is implicitly determined from the layer kernel size, stride and padding amount. Convolutional layers can be thought of as performing inner products

between the kernel $\boldsymbol{w}$ and slices of the input $\boldsymbol{x_{i:i+\kappa}}$ in a sliding window fashion using a step size of s, called the stride. The hidden units can then be formulated as $h_i = (\boldsymbol{w} \star \boldsymbol{x})_i^s + b$, where $(\boldsymbol{w} \star \boldsymbol{x})_t^s = \sum_{k=0}^{\kappa-1} w_{k+1} x_{s(t-1)+1+k}$. In general, applying convolutions results in fewer output units than input units. To offset this effect and increase the number of output units the input can be padded with zeros at its edges. The effect of the kernel size, stride and padding amount on the number output units in a convolutional layer is shown in Figure 5 [17].

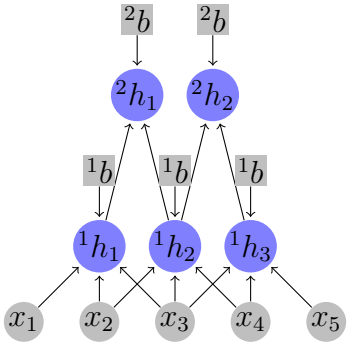

Figure 4: Two convolutional layers with kernel sizes of 3 and 2 units. While the units in the first layer $^1\boldsymbol{h}$ are directly connected to three input elements, the units in the second layer $^2\boldsymbol{h}$ are indirectly connected to four input elements through the first hidden layer connections.

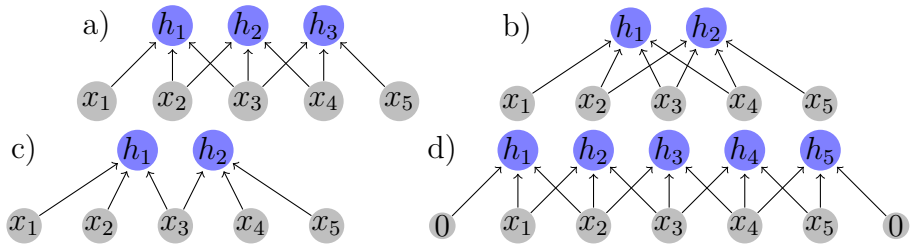

Figure 5: The choice of kernel size, stride and padding amount determine the number of units in a convolutional layer. (a) kernel size: 3, stride: 1, padding: 0. (b) kernel size: 4, stride: 1, padding: 0. (c) kernel size: 3, stride: 2, padding: 0. (d) kernel size: 3, stride: 1, padding: 2.

Another important practical consideration in the design of convolutional layers is the number of input and output channels. Convolutional layers usually have multiple input channels, originating either from the source data (e.g. a multichannel time series or the color channels in an image) or from the multichannel output of a previous convolutional layer. In these cases, each input channel is convolved with its own kernel $\boldsymbol{w}$ and the results are summed in the output units. To increase the capacity of a convolutional layer, additional output channels are usually incorporated, each with their own set of kernels $\boldsymbol{w}$ and biases b [16, 17]. A convolutional layer with multiple input and output channels is shown in Figure 6. Here, the output of channel k can be formulated as $\boldsymbol{h}^k = [h_1^k \; h_2^k \; h_3^k]^T$, where $h_i^k = \sum_c^{C_{in}} (\boldsymbol{w}^{c,k} \star \boldsymbol{x}^c)_i + b^k$ and C_{in} is the

number of input channels. The output of the convolutional layer is called a feature map and can be expressed as $H = [\boldsymbol{h}^1\ \boldsymbol{h}^2\ \ldots\ \boldsymbol{h}^{C_{out}}]$, where C_{out} is the number of output channels in the convolutional layer.

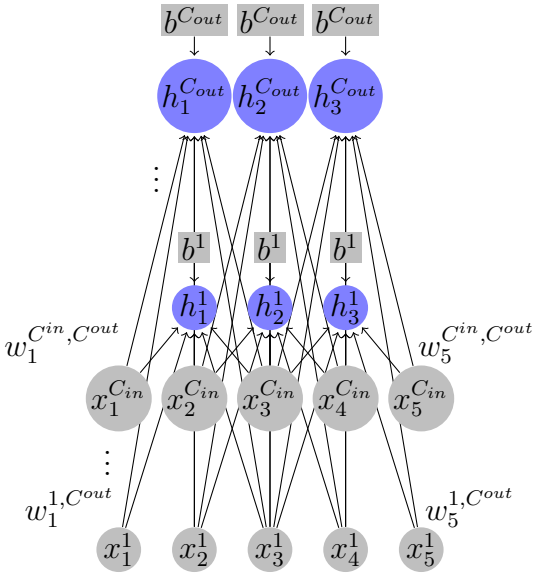

Figure 6: A convolutional layer with C^{in} input and C^{out} output channels.

In the above examples, the hidden units are linear having only inner product and addition operations, $h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$. However, neural networks consisting of strictly linear units can only approximate linear functions. In order to model nonlinear tasks, nonlinear transformations are incorporated in the network units such that $h(\boldsymbol{x}) = g(\boldsymbol{w}^T\boldsymbol{x} + b)$, where g is a nonlinear function called the activation function. The universal approximation theorem states that nonlinear units enable artificial neural networks to act as universal approximators and represent any sufficiently continuous function. There exists a wide range of activation functions the choice of which depends on the unit type, as well as the task the network is trained to perform. The exact design of artificial neural network units is a very active area of research and no strict theoretical guidelines have been developed yet. Among the most common types of hidden units is the rectified linear unit (ReLU) which uses the piecewise linear activation function $g(z) = \max\{0, z\}$ [16, 17].

An artificial neural network defines the distribution $p_{model}(y|\boldsymbol{x};\boldsymbol{\theta})$. The goal is to find the optimal set of network parameters $\boldsymbol{\theta}^*$ so that the model distribution $p_{model}(y|\boldsymbol{x})$ is as close as possible to the true conditional distribution of the data $p_{data}(y|\boldsymbol{x})$. This is done by minimizing a cost function $J(\boldsymbol{\theta})$ that is usually the expected negative log-likelihood of the model distribution over the empirical distribution of the data

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\mathbb{E}_{(\boldsymbol{x},y)\sim\hat{p}_{data}} \log p_{model}(y|\boldsymbol{x}) \\ &= \frac{1}{m}\sum_{i=0}^{m} L(\hat{y}^{(i)}, y^{(i)}), \end{aligned}$$

where $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(m)}, y^{(m)})\}$ is the training data, $\hat{y}^{(i)}$ is the model prediction for $(\boldsymbol{x}^{(i)}, y^{(i)})$ and L is the loss function defined by the model distribution. Artificial neural networks are most commonly trained with stochastic gradient descent using smaller, uniformly sampled subsets of the training data, called minibatches. The training is iterated multiple times over the entire training dataset and each such iteration is referred to as a training epoch. The training procedure for each minibatch of size m' follows four basic steps:

1. Forward pass: $\hat{y}^{(i)} = f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \; \forall \; \boldsymbol{x}^{(i)} \in \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(m')}, y^{(m')})\}$
2. Loss computation: $J(\boldsymbol{\theta}) = \frac{1}{m'} \sum_{i=0}^{m'} L(\hat{y}^{(i)}, y^{(i)})$
3. Gradient computation: $\nabla_\theta J(\boldsymbol{\theta})$
4. Parameter update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla_\theta J(\boldsymbol{\theta})$,

where ϵ is a training parameter called the learning rate. The partial derivatives with respect to each parameter are computed using the chain rule of calculus starting from the output layer and propagating backwards towards the lower layers. This is called backpropagation and it is the most computationally expensive part of both training and inference [16,18]. A widely employed modification to stochastic gradient descent which has been shown to improve model training convergence is stochastic gradient descent with momentum where the update is a linear combination of the gradient and the previous parameter update. In this variant of stochastic gradient descent, the parameter update step becomes $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla_\theta J(\boldsymbol{\theta}) + \alpha \Delta \boldsymbol{\theta}$, where $\Delta \boldsymbol{\theta}$ is the parameter update in the previous step and α is the momentum parameter.

2.2 Fully convolutional networks

Convolutional networks are powerful models in many settings where the task is to approximate a nonlinear function that maps a vector $\boldsymbol{x}$ to a categorical class variable y. Typically, convolutional networks consist of several convolutional layers with a few fully connected layers at the top of the network. Conceptually, these convolutional layers extract deep local features and the fully connected layers map those features to the target classes with the final fully connected layer having one unit for each target class [16]. An example convolutional network used for a three class classification problem is shown in Figure 7.

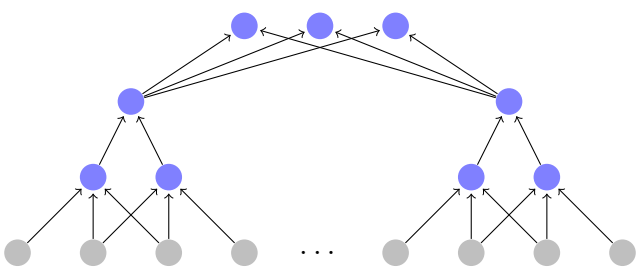

Figure 7: A convolutional network used for classification.

While convolutional networks work extremely well for classifying whole inputs they are not directly applicable for semantic segmentation where the task is to assign a class to every element of an arbitrary-sized input. Even with fixed input sizes, the top fully connected layers would have to be extremely large which would make training such networks computationally intractable. Long et al. were the first to introduce fully convolutional networks for semantic segmentation addressing the problems of prior approaches based on convolutional networks and achieving state-of-the-art results for multiple image segmentation datasets [15]. In their work, Long et al. first introduce the notion of transposed convolution by reversing the forward and backward passes of convolution as shown in Figure 8. They reinterpret fully connected layers as convolutional layers with kernels that span their entire input region, and rearchitecture existing image classification convolutional networks by removing the top fully connected layers and replacing them with transposed convolutional layers to produce dense pixelwise image segmentation maps. Instead of learning a nonlinear function, networks consisting of strictly convolutional layers learn nonlinear filters, which Long et al. call deep filters or fully convolutional networks. Fully convolutional networks can operate on arbitrary-sized inputs and have efficient learning and inference. Figure 9 shows a fully convolutional network where the fully connected layer of the network shown in Figure 7 is replaced by a transposed convolution to produce outputs of the same size as the input.

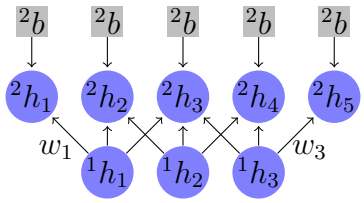

Figure 8: A transposed convolutional layer with a kernel size of 3 units. A transposed convolutional layer reverses the operation of a convolutional layer changing it from many-to-one to one-to-many.

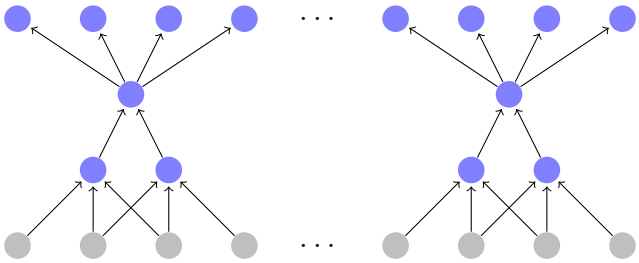

Figure 9: A fully convolutional network used for segmentation.

2.3 Activity recognition using wearable sensors

Activity recognition refers to the problem of predicting a human agent's actions based on sensor data. The target actions can either be lower level actions or activities,

such as simple gestures or bodily movements, or higher level activity descriptions which provide some additional context related to the agent or their environment. Low-level activity examples include "walking", "standing" and "sitting" whereas examples of high-level activities include "working at desk", "exercising" and "cooking". Activity recognition has gathered increasing research interest due to the recent emergence of ubiquitous sensor networks and advances in data mining and machine learning [7–14, 19–23]. Application areas range from fitness tracking and interactive games to assisted living, human behavior analysis and video surveillance [3–6].

Activity recognition methods can be divided into three main categories based on the utilized sensor modalities: wearable, ambient and hybrid sensor-based activity recognition [3, 5]. Wearable sensor-based activity recognition is used to predict low-level gestures and activities by utilizing sensor data from devices incorporated on the agent's body, such as mobile phones, smartwatches and fitness trackers [3,5]. The most commonly used sensors in this category of activity recognition are the sensors in inertial measurement units, namely accelerometers, gyroscopes and magnetometers. The widespread availability of these sensors in many consumer devices has enabled wearable sensor-based activity recognition to become the most prominent and easiest to adopt activity recognition method [4]. Ambient sensor-based activity recognition methods use data from fixed sensors placed in the acting agent's environment (e.g. homes or workplaces), and they are typically used to predict high-level activities. The used sensors are either mounted cameras that directly capture the agent's actions or lower level sensor devices, such as proximity and pressure sensors, RFID tags, as well as Bluetooth and Wi-Fi devices, that capture the agent's interactions with the environment and enable inferring the agent's actions indirectly. Although ambient sensors are less obstructive than wearable sensors, they usually come with higher deployment costs and the set of activities that can be recognized is more limited [4]. Additionally, in contrast to wearable sensor-based activity recognition systems, such systems only work in the spatially fixed environment where the sensors are deployed. Hybrid activity recognition methods combine wearable and ambient sensors and although they generally improve activity recognition performance, they suffer the same drawbacks as ambient sensor-based systems [3].

There has been extensive research of wearable sensor-based activity recognition methods over the last decade, as shown in Figure 10. However, studies vary greatly in their characteristics, such as the target activities, the on-body locations of the used sensors and the recognition methods. The target activities can generally be categorized into three types: 1) periodic activities which are prolonged activities with a natural period or cycle, such as "walking", "running" or "cycling", 2) static activities which are prolonged activities that involve little or no movement, such as "sitting", "standing" and "lying" and 3) sporadic activities which are short in duration and often interspersed with other activities. Examples of the latter include transitions between activities like "sitting-to-standing" and "standing-to-sitting" and one-off activities and gestures like "jumping", "falling" or "taking-phone-out-of-pocket" [6].

Another important characteristic of a wearable sensor-based activity recognition system is the location and number of the on-body sensors. The sensor placement depends on the intended application and the types of activities the system is aiming

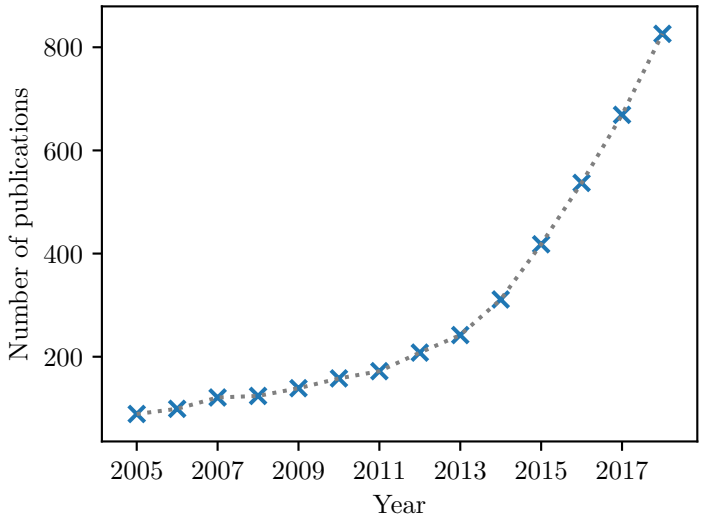

Figure 10: Number of publications returned by the ScienceDirect query using the keywords "wearable sensors activity recognition".

to predict [3]. For example, some body locations might be better fitted for predicting periodic activities than static or sporadic activities. Common locations in literature include the thigh, wrist, arm and foot. Utilizing multiple sensors in different body positions generally increases activity recognition performance and the amount of activities that can be recognized [3]. However, incorporating multiple sensor devices increases deployment costs and introduces complexity related to sensor fusion.

Recognition methods have three main defining characteristics related to their generalization, how the recognition is carried out and whether the method explicitly models the states of the agent or the world. Firstly, recognition models can be classified to either agent-specific or agent-agnostic [6]. Agent-specific models are trained for a specific agent using data provided by the agent during an "enrollment" period whereas agent-agnostic models are trained with data originating from multiple agents and try to generalize predictions to unseen agents. Usually agent-specific models exhibit better performance than agent-agnostic models but have higher deployment costs and introduce friction by requiring the agent to explicitly provide enrollment data [6]. Siirtola et al. propose a novel method of personalizing agent-agnostic models by using limited agent-specific data. Their study shows promising but relatively modest improvements in prediction performance compared to the agent-agnostic model, and the authors suggest that with more research and testing the method could yield even higher improvements in recognition performance [23]. Secondly, activity recognition can either be carried out continuously or in segments. Continuous recognition models use the sensor signals directly and automatically detect the activities in the data stream whereas segmented (sometimes also referred to as windowed) recognition models assume that the activities are split into smaller activity segments by an independent oracle and the model classifies the segments into one of the activity classes. Finally, wearable sensor-based activity recognition

models can be stateful, meaning that the state of the agent or their environment is explicitly incorporated in the model, or stateless, meaning that the predictions are based explicitly on the signals of the sensors without modeling the state of the world [6]. Stateless models are the dominant approach to wearable sensor-based activity recognition as models of this family require less domain knowledge and result in simpler recognition systems.

Generally, building activity recognition models entails four basic steps: 1) preprocessing the data of the deployed sensors, 2) extracting meaningful features from the collected data, 3) training the activity recognition models and 4) testing the models on new data [3]. Preprocessing usually includes signal filtering, data normalization, partitioning (also referred to as segmenting) and resampling. Signal filtering is applied to the raw sensor signals with the aim of reducing signal noise and other artifacts. Several filtering techniques have been used in prior works, such as low-pass and high-pass Butterworth filters, median filters and moving average filters. Data normalization is performed to help with model training convergence and increase performance, and common approaches include standard normalization to zero mean and unit variance as well as Min-Max scaling. Partitioning entails dividing the raw signal to smaller sections to be processed by the recognition model later down the pipeline. Partitioning a continuous sensor stream is a difficult task and plays an important role in recognition performance, particularly for sporadic activities [6]. Partitioning approaches include energy-based partitioning, rest-position partitioning and partitioning using external context sources, but the method that is most commonly used is sliding window based partitioning where the sensor data is divided into time windows, ranging from under 1 s to 10 s and over [3]. Inertial sensors such as accelerometers, gyroscopes and magnetometers don't typically sample perfectly uniformly so often resampling is performed to produce uniformly sampled signals in order to analyze signals in the frequency domain with techniques such as Fast Fourier transform. Generally, higher sampling rates yield higher prediction accuracy but come with higher energy requirements.

Earlier activity recognition works utilized hand-crafted features derived from partitioned signal windows to predict the activities of interest [6–8]. These features include either time-domain features, such as signal magnitude area, standard deviation, median, skewness and autoregressive coefficients, or frequency-domain features, such as spectral energy, entropy and dominant frequency, which are obtained from transforming the partitioned windows using methods such as Fast Fourier transform or discrete wavelet transform. These features are then fed into classification algorithms like decision trees, support vector machines, k-nearest neighbors and naive Bayes. However, developing good representative features by hand is a laborious task and often such features result in unsatisfactory prediction performance and don't generalize well to unseen users and new sensor modalities [4].

More recently, advances in GPU computing frameworks and deep learning have sparked interest among activity recognition researchers and practitioners. Deep learning based activity recognition models usually follow the same basic steps described above with the exception that deep learning models do not require hand-crafted features. Instead, deep learning models jointly perform feature extraction and activity

recognition by automatically learning powerful and robust features from the data. Deep learning based activity recognition models have been shown to consistently outperform conventional machine learning methods using hand-crafted features [8–13]. A common approach for deep learning based activity recognition is to use convolutional networks for labeled sliding windows of sensor data. Hammerla et al. explored deep convolutional and recurrent network architectures with different model configurations and showed that convolutional neural networks perform well for periodic activities, such as "walking" or "running", whereas recurrent neural networks are better for sporadic activities [12]. Ordóñez et al. propose a deep learning based framework for activity recognition using a hybrid network structure utilizing convolutional and LSTM recurrent units. Their system uses convolutional layers to automatically extract features from windowed segments of multichannel motion sensor data from multiple body locations which then get fed into the recurrent layers that model the temporal dependencies of the extracted features. They show that their system outperforms activity recognition systems based solely on convolutional networks using similarly windowed signals [11].

While deep learning based models using sliding windows have shown promising results, particularly for periodic and static activities, their prediction performance depends on the arbitrary temporal partitioning of the sensor data which is particularly problematic for sporadic activities [11]. Additionally, sliding window segmentation gives rise to the so-called multiclass window problem where a window contains data from multiple classes. In these cases, both data labeling as well as predictions are ill-defined. Fully convolutional networks are relatively unexplored in the context of wearable sensor-based activity recognition. However, fully convolutional networks offer a straightforward way for continuous activity recognition and automatic activity segmentation by dense samplewise prediction of an arbitrary-sized sensor data stream. Zhang et al. readapted a biomedical image segmentation model based on fully convolutional networks [21]. In their work, sliding windows of triaxial accelerometer signals are first mapped into a multichannel images with a single pixel column which are then fed into the image segmentation model for training and activity recognition. Their study shows promising results but mostly focuses on the adaptation of an existing image segmentation model to the activity recognition task. In addition, they used a random train/validation/test split of windows, meaning that the sets contain data originating from the same set of subjects. In this work, we develop a novel segmentation model specifically for wearable sensor-based activity recognition which utilizes one-dimensional convolutions and thus no extra steps are required to convert the sensor signals into image-like objects. We perform model training experiments in order to find the best performing model architecture parameters and evaluate the difference between windowed and continuous activity recognition on an independent set of test subjects in order to quantify the agent-agnostic performance of the model. Finally, we perform occlusion sensitivity experiments for different activities to assess our model's robustness to perturbations in the input signals and to help explain the model predictions.

3 Research material and methods

3.1 The MobiAct dataset

The MobiAct dataset is a publicly available dataset for wearable sensor-based activity recognition and fall detection [24]. The dataset consists of three distinct parts and contains data collected from 67 subjects with more than 3200 trials. The first part contains segments of *activities of daily living* where subjects perform a single activity per trial, the second part contains longer segments of *scenarios of daily living* where subjects perform multiple activities sequentially in each trial, and the third part contains short segments of various types of *falls*. This study uses the first and second parts of the MobiAct dataset which contain the low-level activities shown in Table 1.

Activity	Label	Type	Subjects	Duration
Standing	STD	Static	60	300 min
Walking	WAL	Periodic	61	305 min
Jogging	JOG	Periodic	61	91 min
Jumping	JUM	Periodic	61	91 min
Stairs up	STU	Periodic	61	72 min
Stairs down	STN	Periodic	61	73 min
Stand-to-Sit	SCH	Sporadic	61	36 min
Sitting	SIT	Static	19	19 min
Sit-to-Stand	CHU	Sporadic	19	11 min
Car step-in	CSI	Sporadic	60	35 min
Car step-out	CSO	Sporadic	60	36 min

Table 1: Table of activities used in this study.

The data in each trial is collected using a commercial Android smartphone placed in either one of the subjects' front trouser pockets, in any orientation freely chosen by the subject. The data consists of the recordings of two hardware motion sensors, the accelerometer and the gyroscope, as well as one additional software orientation sensor which is derived from the aforementioned two hardware sensors. All three sensors are triaxial and independently measure the device's motion or orientation with respect to each Android coordinate axis, as shown in Figure 11 and Table 2.

3.2 Experimental procedure

All sensor data is interpolated to a sampling rate of 50 Hz using first order splines and the trials are then split into non-overlapping windows of 3 s, resulting in windows of 150 individually labeled samples. Finally, each sensor channel is normalized as

$$x' = -1 + 2\frac{x - \min(x)}{\max(x) - \min(x)},$$

where x is the sensor channel data, and $\min(x)$ and $\max(x)$ are the minimum and maximum ranges of the sensor channel, as shown in Table 2.

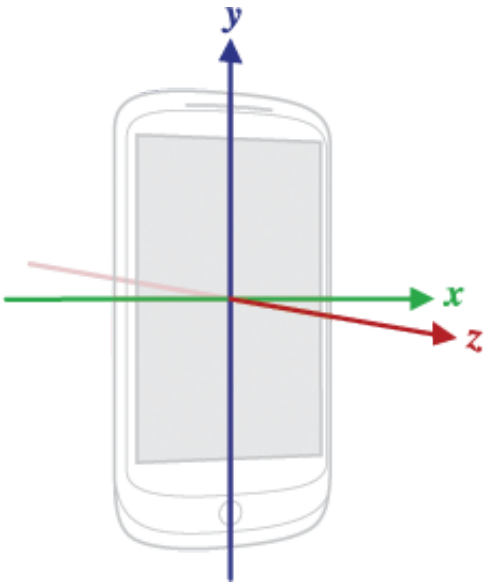

Figure 11: The Android motion sensor coordinate system.

Sensor	Channel	Description	Units	Data range
Accelerometer	x	Acceleration along the x-axis.	m/s^2	[-20, 20]
	y	Acceleration along the y-axis.	m/s^2	[-20, 20]
	z	Acceleration along the z-axis.	m/s^2	[-20, 20]
Gyroscope	x	Rate of rotation around the x-axis.	rad/s	[-10, 10]
	y	Rate of rotation around the y-axis.	rad/s	[-10, 10]
	z	Rate of rotation around the z-axis.	rad/s	[-10, 10]
Orientation	Azimuth	Angle around the z-axis.	degrees	[0, 360]
	Pitch	Angle around the x-axis.	degrees	[-180, 180]
	Roll	Angle around the y-axis.	degrees	[-90, 90]

Table 2: The Android motion and orientation sensors used in the MobiAct dataset.

In this work, we propose a novel fully convolutional network for wearable sensor-based activity recognition which we coin *SensorFCN*. Our model can process a sensor signal of arbitrary length and produces dense samplewise activity predictions, thus automatically segmenting the sensor signals into sections of activities. The SensorFCN architecture consists of three components, namely the *encoding*, the *decoding* and the *scoring* component, shown in Figure 12. The encoding component consists of three convolutional layers followed by ReLU activations. These layers downsample their inputs by applying one dimensional non-padded convolutions and encode high-level information about the sensor signals. The first convolutional layer convolves each input sensor channel with kernels of size κ and produces an output feature map with n_{filter} dimensions. The second and third convolutional layer subsequently double the dimensionality of the feature map using kernels of sizes 5 and 3, respectively. The decoding network component consists of a single transposed convolution layer with kernel size $\kappa + 6$, chosen such that the produced layer outputs are upsampled back to the original length of the input sensor signals.

In addition, the decoding component maps the high dimensional $(4 \times n_{filter})$ feature map produced by the encoding component to a lower dimension (n_{filter}). Finally, the scoring layer consists of a convolutional layer that applies a unit convolution to the decoding component output feature map and maps it to n_{class} channels where each channel corresponds to a target activity. The final activity segmentation map consists of samplewise activity predictions which are obtained by the arg max of the channel dimension of the scoring component output. The SensorFCN models and the experiments discussed below are implemented with PyTorch, an open source deep learning framework for the Python programming language, and all computations are performed using a Tesla K80 GPU.

All SensorFCN models in this work are trained with stochastic gradient descent against a samplewise categorical cross-entropy loss. The loss is weighted with class weights that are inversely proportional to the total activity durations shown in Table 1 in order to correct the activity class imbalance in the source dataset. Training is performed for 50 epochs in minibatches of 32 windows using a momentum of 0.9 and an initial learning rate of 0.001 which is reduced by a factor of 0.9 every 5 epochs. The subjects are randomly split into separate sets for model training, validation and testing containing 53, 7 and 7 subjects, respectively. The train, validation and test sets then consist of the windows belonging to the corresponding subjects. The generalization to new agents is evaluated by the validation and test performances using four common semantic segmentation metrics: sample accuracy, mean accuracy, mean IoU and frequency weighted IoU [15]. Let n_{ij} be a sample of class i predicted to belong to class j, n_{class} be the number of classes and $t_i = \sum_j n_{ij}$ be the total number of samples of class i. The reported metrics can then be formulated as

- Sample accuracy: $\sum_i n_{ii} / \sum_i t_i$
- Mean accuracy: $(1/n_{cl}) \sum_i n_{ii}/t_i$
- Mean IoU: $(1/n_{cl}) \sum_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$
- Frequency weighted IoU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$.

The experimental part of this work consists of a series of experiments aiming to quantify the effects of the SensorFCN network architecture parameters on the activity recognition and segmentation performance. In each experiment, a single network parameter is varied and the four performance metrics are evaluated on the validation set. The model is then iteratively refined by selecting the best performing parameter value for the next set of experiments. Firstly, the input kernel size κ is varied and a SensorFCN model is trained $\forall\ \kappa \in \{5, 15, 25, \ldots, 135\}$. All three Android sensors are used and the network filter parameter n_{filter} is set to 16. Secondly, the effect of the choice of the input sensors is quantified by training a SensorFCN model for each sensor combination while keeping the network filter parameter at 16. All three channels of a chosen sensor are used, meaning that the number of input channels in the network is always 3, 6 or 9, depending on whether the number of chosen sensors is 1, 2 or 3, respectively. The third experiment investigates the effect of the network filter parameter n_{filter} on the model performance by training a SensorFCN

model $\forall\ n_{filter} \in \{16, 32, 64, 128\}$. In the fourth experiment, we introduce a new network component at the end of the network which we call the *smoothing* component. Intuitively, the samplewise predictions produced by the network should not change too frequently as the time difference between two subsequent samples is only 0.02 s. This new component is appended to the network after the scoring component and intends to reduce the prediction variability and smooth out the produced activity labels. The smoothing component consists of a convolutional layer with n_{class} filters and a kernel size of κ_{smooth}. In order to maintain the correct output size, this layer pads the scoring component output with $\kappa_{smooth}/2$ zeros. The effect of the smoothing kernel size on the model performance is evaluated by training a SensorFCN model $\forall$ $\kappa_{smooth} \in \{5, 15, 25, \ldots, 135\}$.

After finding the best parameter values for the SensorFCN model, the generalization performance to unseen agents is evaluated by computing the sample accuracy on the test set using two different approaches. The first approach performs inference on the fixed length activity windows, which is analogous to a windowed activity recognition system where the input sensor signals are first partitioned into smaller windows which are then passed on to the recognition pipeline. The second approach performs inference on a trial level, which is analogous to a continuous activity recognition system where the activity recognition is performed on the input signal in its entirety.

Finally, we perform occlusion sensitivity analysis on segments of different activities where parts of the input sensor signals are occluded and the changes to the model's output are monitored. Originally proposed by Zeiler and Fergus, these methods are widely used in literature to quantify the robustness of deep learning models with respect to changes in their input space, as well as to help explain which parts of the input contribute most to the model predictions [25]. In our occlusion sensitivity experiments, a segment of input sensor data representing a specific activity is first forward propagated in the network and then the output of the smoothing layer is passed on to a softmax normalizer in order to obtain the normalized unoccluded prediction scores for the input segment. The softmax normalization is performed along the output channel dimension so that all channels sum up to 1. The softmax normalized can be expressed as

$$\text{Softmax}(\mathbf{x}_i) = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x_j})},$$

where $\mathbf{x}$ is a vector containing the n_{class} output channels. Then, we slide an occlusion kernel of $\kappa_{occlusion}$ zeros with a stride of 1 and pass the smoothing layer's output to the softmax normalizer to obtain the normalized scores of the occluded input signals. The occlusion sensitivity is then defined as $1 - s_{occluded}/s_{original}$, where $s_{occluded}$ and $s_{original}$ are the softmax normalized scores of the activity for the occluded and the unoccluded input sensor signals, respectively. The final occlusion sensitivity for the input segment is obtained by averaging the occlusion sensitivities across all occlusion positions.

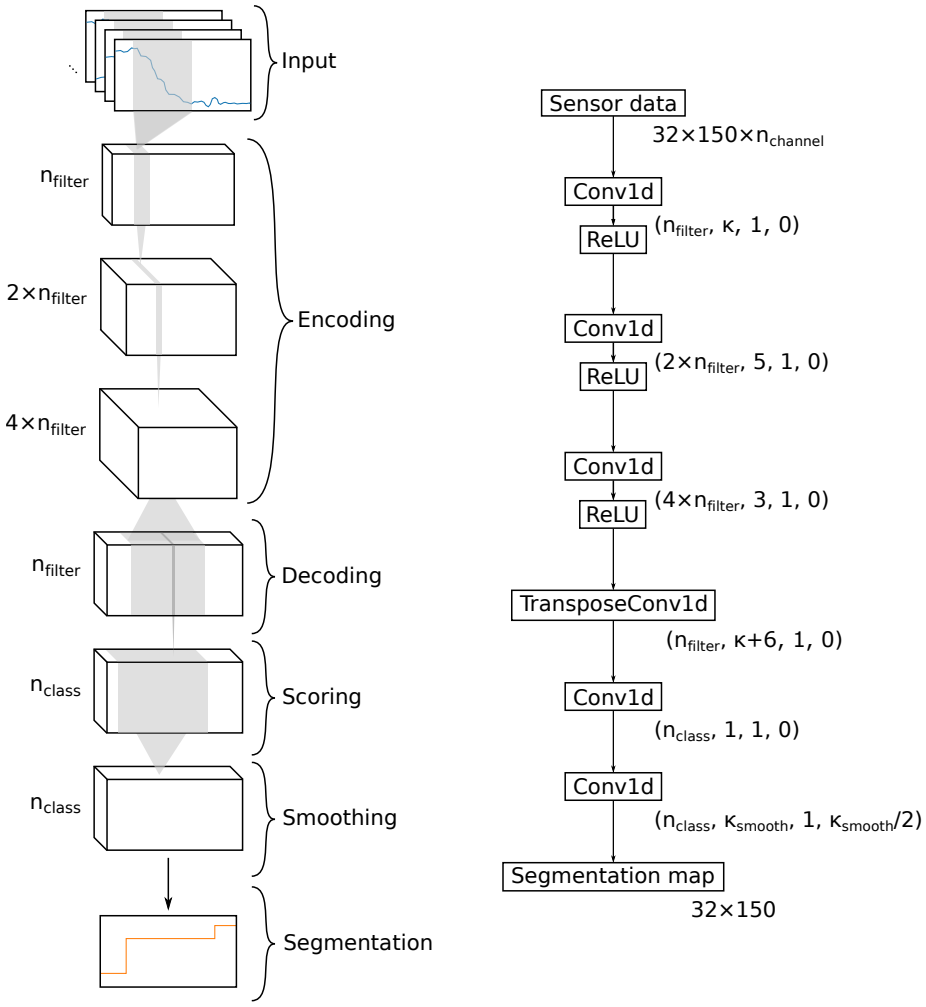

Figure 12: The SensorFCN network architecture. (Left) Each block represents a feature map produced by a single layer within a network component. The widths and depths of the blocks represent the temporal length and the dimensionality of the feature maps, respectively. The feature map lengths first decrease as one dimensional non-padded convolutions are applied in the network and then increase back to the original length of the input signal via a transposed convolution operation. (Right) The network layer types and parameters. The input sensor data is processed in tensors of shape (minibatch size $\times$ window length $\times$ number of input sensor channels) and the produced output contains the samplewise activity predictions for each window in the minibatch. The convolutional layer parameters are shown in tuples of 4 elements containing the number of filters, the kernel size, the stride and the amount of zero padding.

4 Results

4.1 SensorFCN architecture

In this section, we present the results of the SensorFCN model training experiments where a single network architecture parameter is varied with the aim of finding the best performing network parameter values on the MobiAct dataset. The first experiment entailed training SensorFCN models and varying the input kernel size κ such that $\kappa \in \{5, 15, 25, \ldots, 135\}$. In this experiment, the trained models use all three Android sensors included in the MobiAct dataset: the accelerometer, the gyroscope and the orientation sensor. The filter parameter n_{filter} was set to 16 and the models do not utilize the SensorFCN smoothing component. The training and validation sample accuracy, mean accuracy, mean IoU and frequency weighted IoU are shown as a function of the input kernel size in Figure 13. All four metrics follow the same pattern of first increasing as the input kernel size is increased and then decreasing when the input kernel size is increased further. The input kernel size achieving the best performance on the validation set across all four metrics is $\kappa = 55$ which corresponds to a temporal window of slightly over a second on the resampled MobiAct data at 50 *Hz*. The resulting validation sample accuracy for $\kappa = 55$ is 0.873, the mean accuracy is 0.745, the mean IoU is 0.656 and the frequency weighted IoU is 0.813.

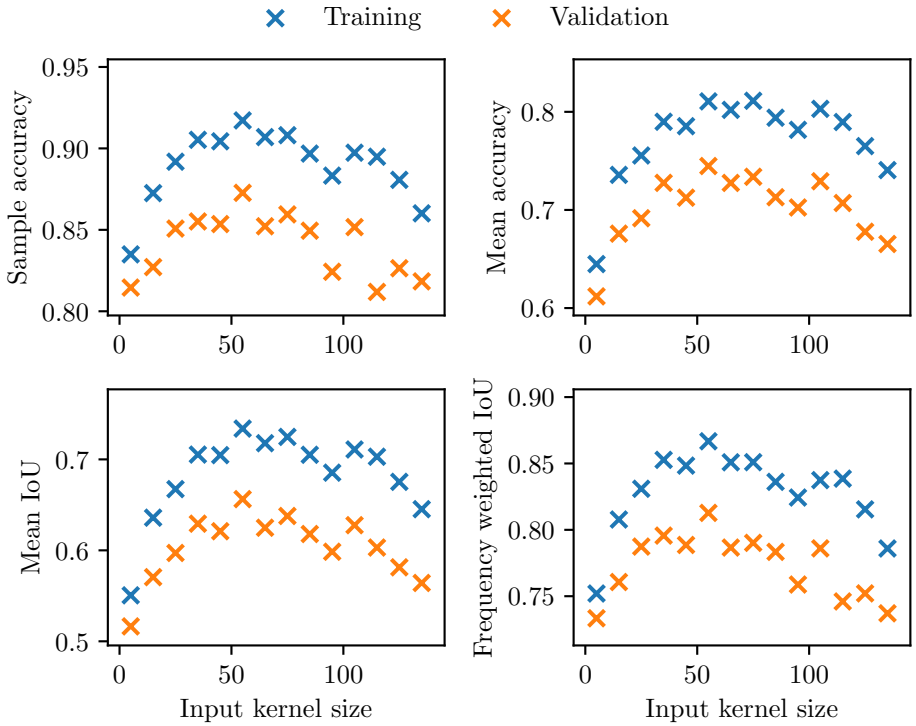

Figure 13: The training and validation metrics as a function of the input kernel size.

Figure 14 shows the validation sample accuracy as a function of the input kernel size broken down for each activity type. The figure reveals that larger input kernel sizes are particularly helpful in the recognition of periodic and sporadic activities. Smaller kernel sizes are preferable for static activity recognition because a smaller receptive field is sufficient to capture the full information of the static signal whereas larger receptive fields are more robust to small changes and thus require more samples to detect a change in the signal state.

In the next experiment, a SensorFCN model was trained for each of the seven different combinations of the Android sensors as the model inputs. The trained models do not utilize the smoothing component of the SensorFCN network architecture and the filter parameter and input kernel size were set to $n_{filters} = 16$ and $\kappa = 55$, respectively. The training and validation sample accuracy, mean accuracy, mean IoU and frequency weighted IoU for all seven Android sensor combinations are shown in Figure 15. The SensorFCN model utilizing the accelerometer is the best performing out of the models utilizing only one Android sensor and including the gyroscope or the orientation sensor brings a small increase in recognition performance. The best performing model is the one utilizing the accelerometer and the gyroscope resulting in a validation sample accuracy of 0.881, a mean accuracy of 0.754, a mean IoU of 0.667 and a frequency weighted IoU of 0.827 which is an improvement over the previous best model utilizing all three Android sensors.

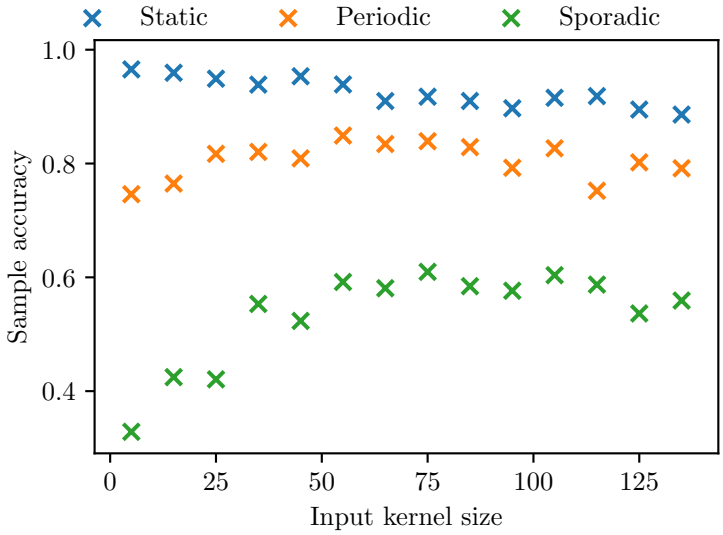

Figure 14: The training and validation sample accuracy for each activity type as a function of the input kernel size.

The third experiment entailed training SensorFCN models with varying filter parameters n_{filter}. Four SensorFCN models were trained with $n_{filter} \in \{16, 32, 64, 128\}$ using the accelerometer and gyroscope Android sensors as the model inputs. The input kernel size was kept at $\kappa = 55$ and the models do not utilize the SensorFCN smoothing component. The sample accuracy, mean accuracy, mean IoU and frequency weighted IoU for the trained models are shown in Table 3. The table shows that

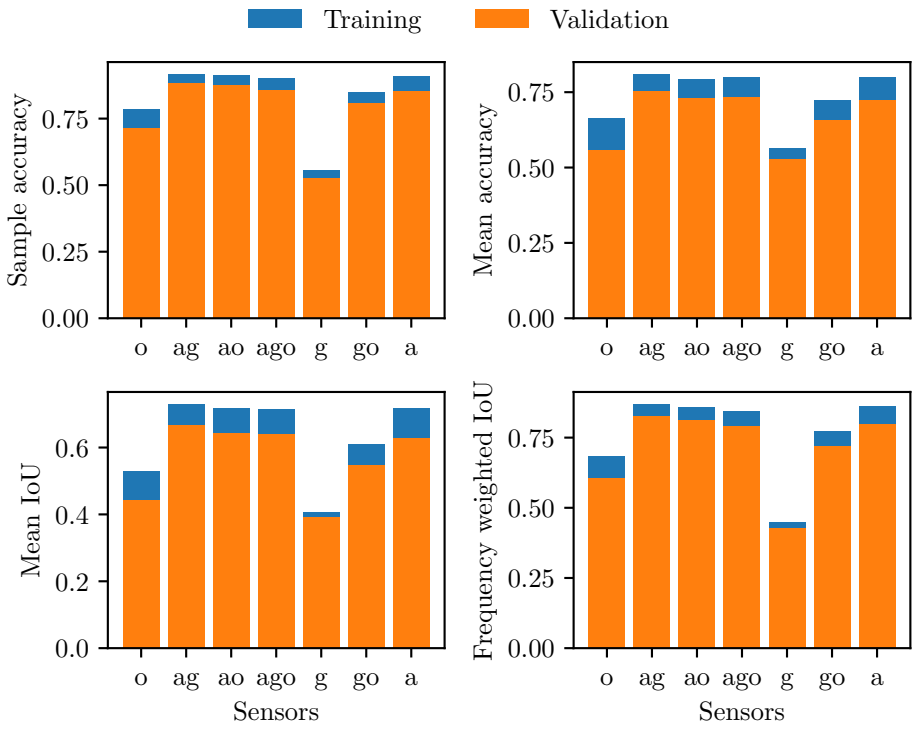

Figure 15: The training and validation metrics for the models trained with all different combinations of the Android sensors as the model inputs.

increasing the filter parameter of the SensorFCN network architecture tends to result in increased activity recognition performance. However, higher n_{filter} parameter values result in larger models in terms of the number of trainable model parameters, as shown in the last column of the table, which increases model training and inference times. Table 4 shows the break down of the validation sample accuracy for each activity type for all n_{filter} parameter values used in this experiment. The table shows that sporadic activity recognition benefits the most from increasing the SensorFCN filter parameter. The filter parameter value resulting in the best performance is 128 which brings the validation metrics up to 0.895 for the sample accuracy, 0.793 for the mean accuracy, 0.704 for the mean IoU and 0.846 for the frequency weighted IoU.

n_{filter}	Sample acc.	Mean acc.	Mean IoU	F.W. IoU	Parameters
16	0.874	0.738	0.65	0.818	76763
32	0.884	0.769	0.679	0.831	295851
64	0.891	0.778	0.69	0.841	1161035
128	0.895	0.793	0.704	0.846	4599435

Table 3: The validation metrics for the models with different filter parameters.

The fourth and final of the SensorFCN model training experiments involved introducing the smoothing component to the SensorFCN architecture and training models with varying smoothing kernel sizes κ_{smooth} such that $\kappa_{smooth} \in \{5, 15, 25, \ldots, 135\}$.

n_{filter}	Static	Periodic	Sporadic
16	0.939	0.858	0.533
32	0.95	0.86	0.602
64	0.94	0.88	0.594
128	0.94	0.886	0.627

Table 4: Validation sample accuracy of the trained models with different filter parameter values broken down for each activity type.

The trained models used the Android accelerometer and the gyroscope as inputs with a kernel size of $\kappa = 55$ and a filter parameter of $n_{filter} = 128$. The training and validation sample accuracy, mean accuracy, mean IoU and frequency weighted IoU are shown in Figure 16 as a function of the smoothing kernel size κ_{smooth}. The figure shows that including the smoothing component boosts model performance and the best performing smoothing kernel size is $\kappa_{smooth} = 115$, achieving a 0.909 sample accuracy, 0.832 mean accuracy, 0.744 mean IoU and 0.865 frequency weighted IoU on the validation set. Figure 17 shows the validation sample accuracy for each activity type as a function of the smoothing kernel size. The figure shows that adding the smoothing component to the SensorFCN model results in a significant increase in the recognition performance of sporadic activities.

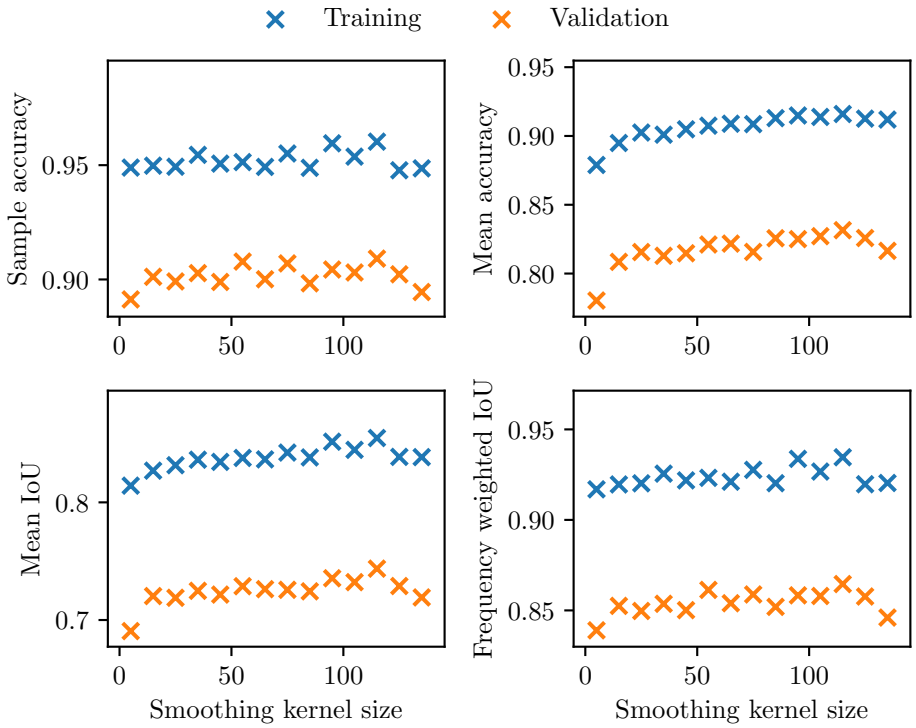

Figure 16: The training and validation metrics as a function of the smoothing kernel size.

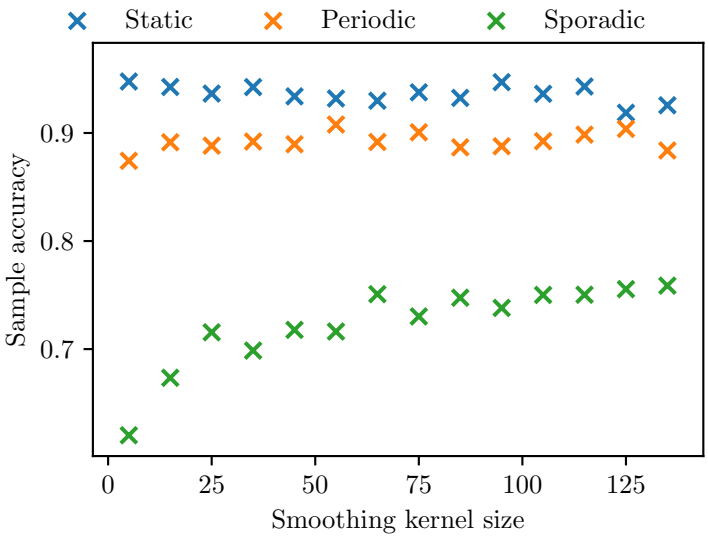

Figure 17: The validation sample accuracy for each activity type as a function of the smoothing kernel size.

4.2 Generalization and explainability

In this section, we evaluate the generalization performance of the SensorFCN model using the best performing network parameter values determined in the model training experiments of the previous section. The used model is the SensorFCN model using an input kernel size of $\kappa = 55$, a filter parameter value of $n_{filter} = 128$ and the accelerometer and gyroscope data as the model inputs. In addition, the evaluated model utilizes the smoothing component with a smoothing kernel size of $\kappa_{smooth} = 115$. The generalization of the model is tested using two inference methods using a separate test set consisting of the activity data of the agents that were not included in the model training and validation phase. The first inference method involves performing forward passes on a window level, similar to how a windowed activity recognition system processes the output of an independent oracle that presegments the activity data. The second inference method performs the forward pass on a trial level, similar to how a continuous activity recognition system processes the full sensor data stream and automatically finds the activities within the data. The test sample accuracy on an activity type level for both the windowed and continuous inference approaches are shown in Table 5. The table shows that performing inference on the full trial data is better than on a window level, yielding a 0.966 test sample accuracy on the trial level and 0.951 on a window level. Additionally, trial level inference results in much higher sample accuracy for sporadic activities than window level inference.

The sample confusion matrix for each activity evaluated on the test trials is shown in Figure 18. Overall most activities show high sample accuracies with the periodic activities of walking (WAL) and jogging (JOG) having the highest sample accuracy of 0.99 and the sporadic activities of transitioning from standing to sitting (SCH) and stepping into a car (CSI) having the lowest sample accuracy of 0.79 and 0.81,

	Windowed	Continuous
Static	0.958	0.959
Periodic	0.964	0.979
Sporadic	0.77	0.851
Overall	0.951	0.966

Table 5: Sample accuracy on the test set performing inference on the activity window level (windowed) and on the activity trial level (continuous).

respectively. Some of the activity recognition confusion occurs between activities that share similar motion characteristics. For example, walking up the stairs (STU) is often confused with walking on a flat surface (WAL), both of which involve taking steps. Similarly, taking the stairs down (STN) gets confused with jogging (JOG) which both involve taking steps with higher impacts and acceleration values. However, a significant portion of the misclassifications of sporadic activities seem to stem from the somewhat ambiguous ground truth transition points between activities. For example, the transition from standing to sitting (SCH) is often confused with sitting (SIT). This is not because these two activities are similar per se but rather because the predicted transition moment between the two activities is either too early or too late with respect to the ground truth which brings the sample accuracy down, especially considering the sporadic activities are short in length.

Figure 19 shows an example 50 second long segment of a test trial where the agent first walks up to their car, steps in and sits there for a few seconds, and finally steps out of the car. The Android accelerometer and gyroscope are used as the model inputs and the ground truth and predicted activities are shown in the bottom part of the figure with a blue and orange line, respectively. The used model was the SensorFCN model with input kernel size $\kappa = 55$, filter parameter $n_{filter} = 128$ and smoothing kernel size $\kappa_{smooth} = 115$. We see that the model produces dense samplewise activity predictions and the alignment between the ground truth and the predicted labels is generally within 1 s of the correct one.

Occlusion sensitivity analysis with an occlusion kernel size $\kappa_{occlusion} = 50$ was performed on three sections of the example trial covering three different activities: walking, standing and sitting. Figure 20 shows the occlusion sensitivity of each input sensor channel on a 7 s segment of walking and we see that the accelerometer y-axis is the most sensitive input to occlusion with an average occlusion sensitivity of 0.26. Figure 21 shows a zoomed-in occlusion sensitivity plot for the accelerometer y-axis which highlights the parts of the walk that are more sensitive to occlusion and which the model more strongly associates with walking. Figure 22 shows the input occlusion sensitivity for two segments of sitting and walking. The figure shows that the accelerometer y-axis is the most important model input for recognizing standing with an average occlusion sensitivity of 0.12 and the accelerometer z-axis is the most important for sitting with an average occlusion sensitivity of 0.05. These axes are the ones parallel to the gravitational acceleration vector during the two activities with the phone in the agent's pocket, as shown in Figure 11.

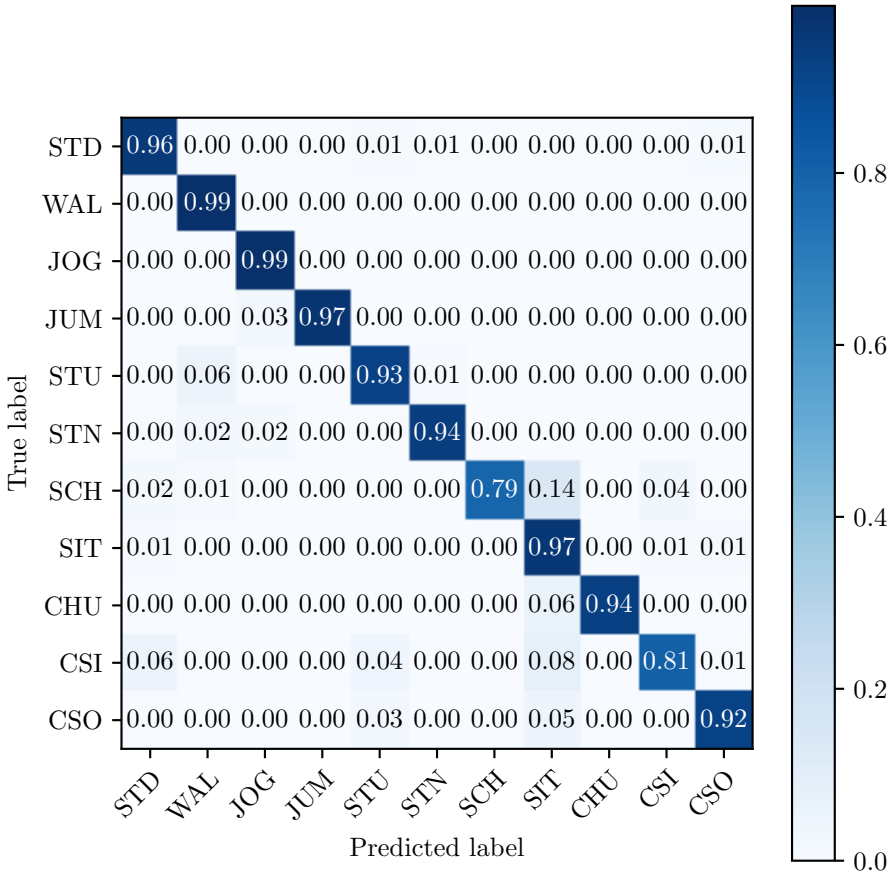

Figure 18: Sample confusion matrix for the test agents evaluated on a trial level.

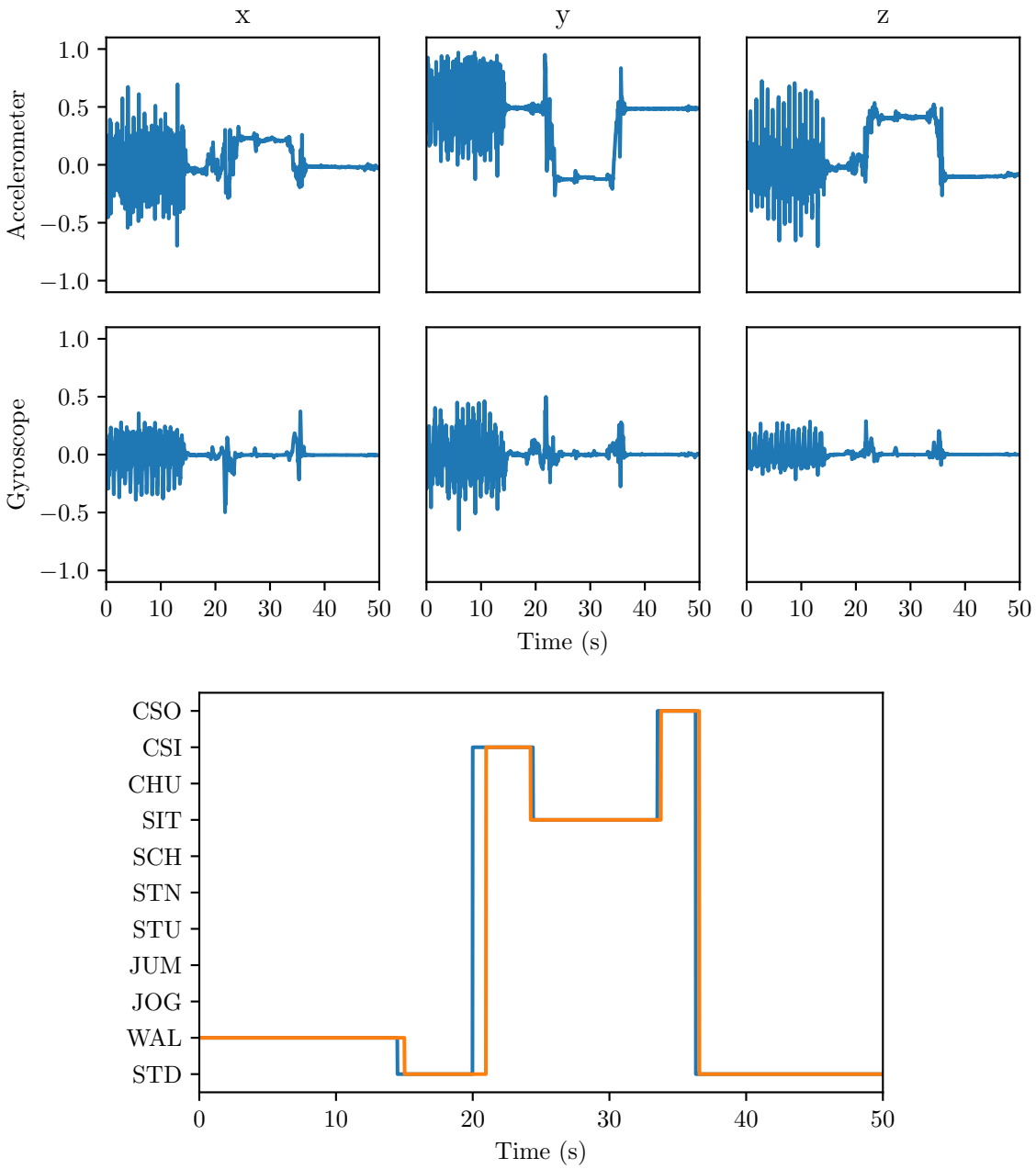

Figure 19: (Top) Android accelerometer and gyroscope data of an agent walking up to their car, getting in and then getting out again. (Bottom) Ground truth (blue) and predicted (orange) activity labels.

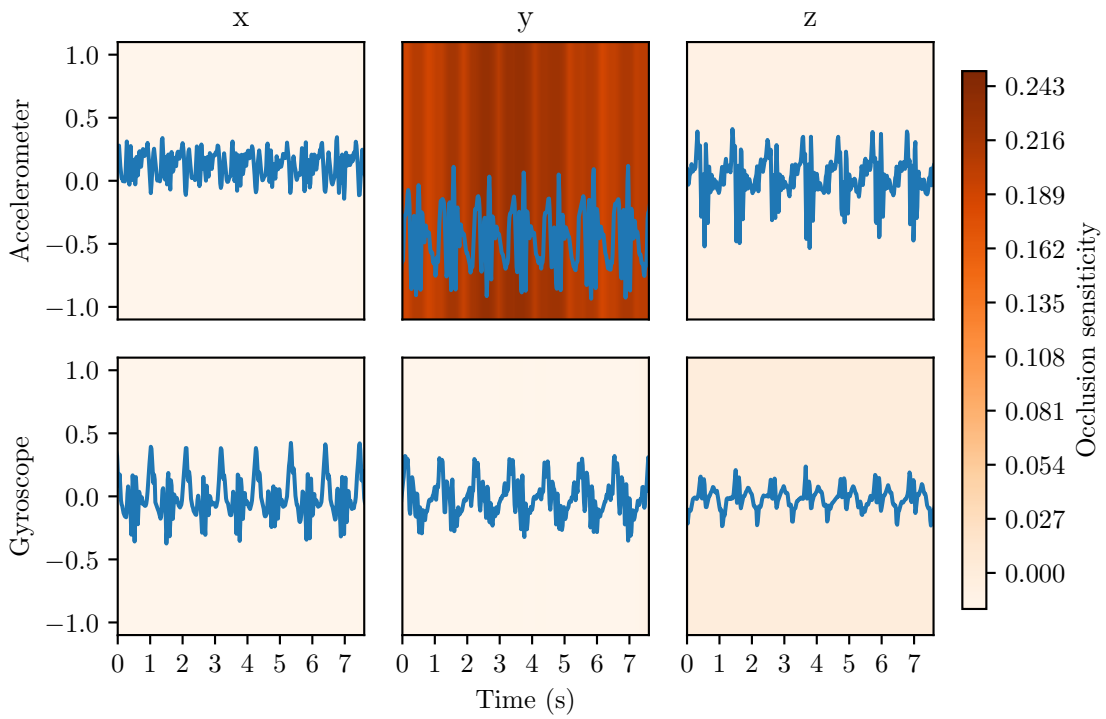

Figure 20: Occlusion sensitivity for a segment of sensor data of walking.

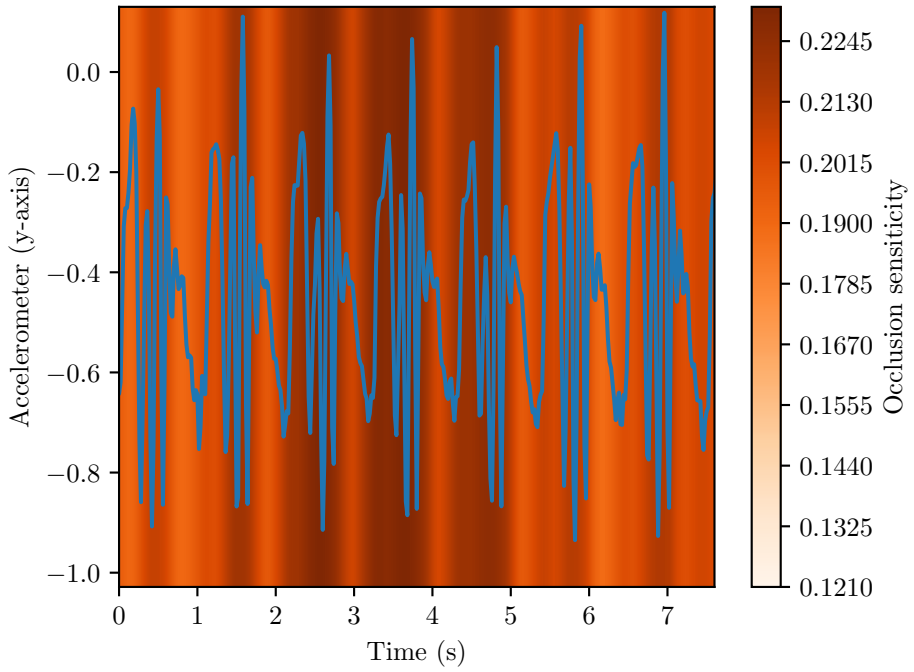

Figure 21: A zoomed-in version of the accelerometer y-axis in Figure 20. The sections with higher occlusion sensitivity are the parts of the gait cycle associated with the heel strikes which the model associates with walking.

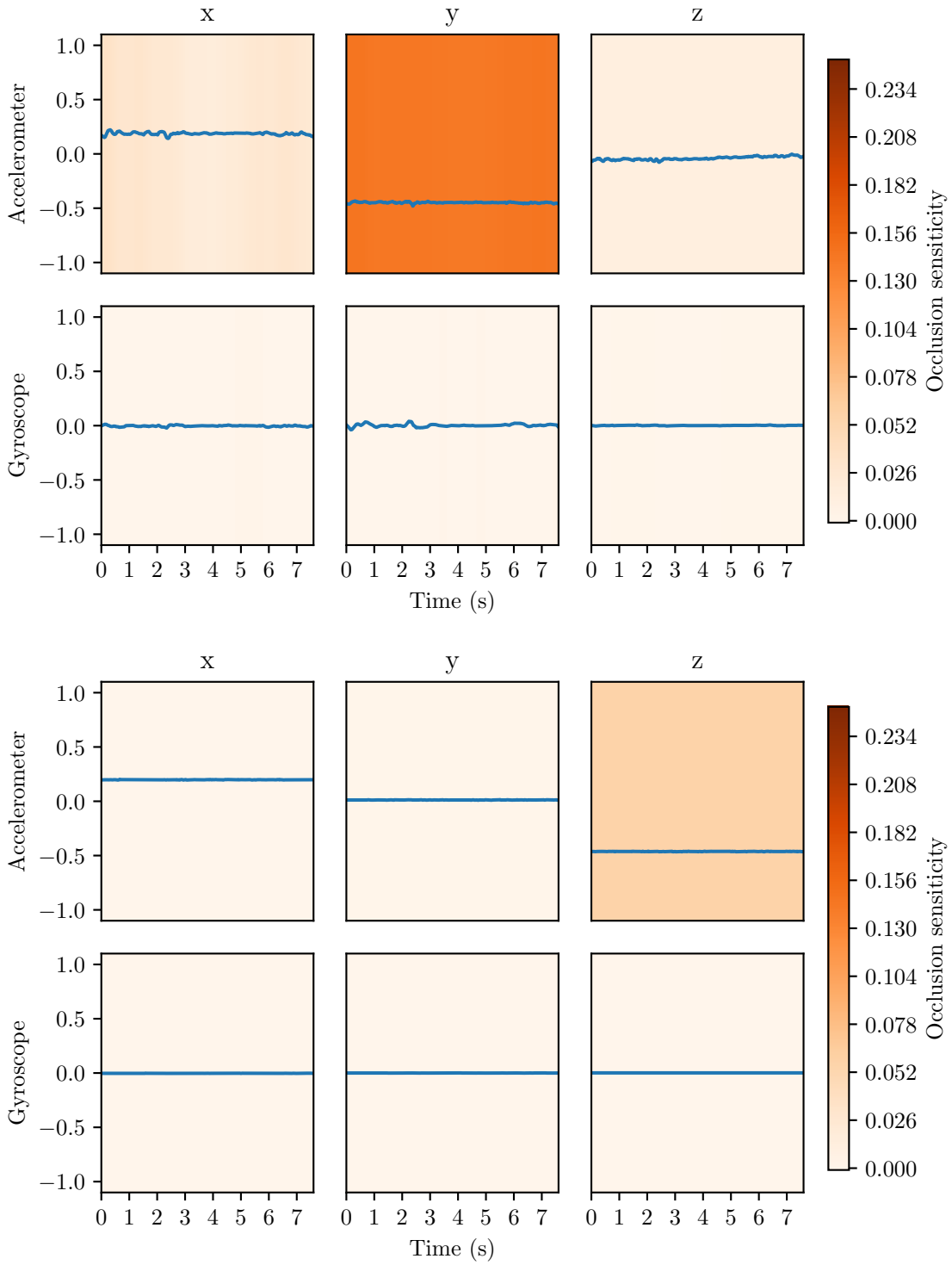

Figure 22: Occlusion sensitivity for a motion sensor data segment of standing (top) and sitting (bottom).

5 Summary

Wearable sensor-based activity recognition is an area of pattern recognition with the aim of predicting the actions of a human agent based on data obtained from sensors worn or incorporated on the agent's body with applications ranging from fitness tracking and assisted living to human-computer interaction [1, 2]. The state-of-the-art methods found in literature utilize sliding windows to partition the sensor signal streams into smaller fixed size windows which are then classified into activities by a deep neural network [9–14]. However, this approach can result in instances where a signal window contains multiple activities, in which case these methods fail to correctly classify the activities in the window and potentially result in missed activities. This so-called multiclass window problem is especially problematic when trying to detect short-lived and irregularly occurring sporadic activities. In this work, we proposed a novel approach to wearable sensor-based activity recognition using fully convolutional networks. Our model, which we coin *SensorFCN*, can process a stream of motion sensor data of arbitrary length and outputs a samplewise activity segmentation map. Because of this, the proposed model is not affected by the multiclass window problem and can precisely detect sporadic activities. The MobiAct dataset consisting of 67 subjects performing 11 different activities was used to train and validate the proposed model. The data is collected using a commercial Android smartphone placed in the subjects' pants' pockets and contains the readings of three sensors: the accelerometer, the gyroscope and the orientation sensor.

The first set of experiments included training and validating SensorFCN models with different model parameters using mutually exclusive sets of subjects for model training and validation in order to assess the parameters' effect on the activity recognition performance and to find the best performing model parameters. We found that varying the input kernel size of the model has a noticeable impact on activity recognition performance, particularly for sporadic activities. The trained models utilizing an input kernel size in the range of 35 – 85 samples, which corresponds to an input kernel spanning roughly 1 s on the resampled data, had similar performance to each other. Additionally, we found that the accelerometer is the most important input sensor and that the model utilizing both the accelerometer and the gyroscope showed the best activity recognition performance out of all the models using the different sensor combinations. The proposed SensorFCN architecture performs reasonably well even with a relatively small number of parameters. However, increasing the number of filters in the model increases recognition performance, particularly for sporadic activities. Finally, we introduced a smoothing component to the proposed SensorFCN network architecture. This component consists of a convolutional layer at the end of the network, after the decoding and the scoring network components, which improved activity recognition performance significantly. We found that larger smoothing kernel sizes increase sporadic activity recognition performance while leaving static and periodic activity recognition performance relatively unaffected.

The second part of the experiments included performing inference on a test set which consisted of motion sensor data from a set of subjects that were held out during the model training and validation experiments. Two different inference approaches

were evaluated, one using presegmented windows of sensor signals and the other one using the full signals of each data collection session. We showed that performing inference on longer segments results in increased activity recognition performance which resulted in a 96.6% sample accuracy on the test set. In addition, we performed occlusion sensitivity experiments where parts of the input motion sensor signals were occluded in order to measure the model's robustness to changes in the input data and to better understand which input signals contribute most to the model predictions for different activities. The occlusion sensitivity experiments showed again that the accelerometer is the most important model input. In particular, occluding the accelerometer axis that is parallel to the gravitational acceleration seems to affect recognition performance most, which is explained by the fact that this axis contains the phone orientation information with respect to the ground. Occlusion sensitivity seems to be slightly higher for periodic activities than for static activities but in general, the occlusion sensitivity scores are relatively low even when a large occlusion kernel was used, showing that the trained models are robust to perturbations in the input motion sensor signals.

Wearable sensor-based activity recognition is a promising way of detecting human activities based on the motion sensor signals found in many every day smart devices such as smartwatches and smartphones. Our proposed SensorFCN model is able to overcome the multiclass window problem persistent in many of the state-of-the-art approaches found in literature which enables accurate detection of short-lived and randomly occurring sporadic activities. The accurate and reliable detection of such actions is useful not only in traditional activity recognition applications, but also in developing new ways of human-computer interaction based on motion gestures. Our study focused on exploring the feasibility of fully convolutional network architectures for wearable sensor-based activity recognition and while we achieved good generalization performance on the MobiAct dataset the resulting optimal model parameters might not be the universally best performing ones. In order to get a more robust understanding of how the model parameters affect the activity recognition performance and which parameter values result in the best performance, a k-fold cross-validation strategy could be employed where the model training experiments are repeated for different splits of training and validation subjects and the results are then averaged across the different splits. Additionally, the model could be tested using more public activity recognition datasets [26–28]. Although these experiments would result in more robust optimal model parameter values, such methodology is more expensive in terms of GPU time and beyond the scope of this study. Future research topics could include exploring the importance of network depth by increasing the number of convolutional layers in the encoding component of the network and observing the change in activity recognition performance. Another future research direction could be assessing the on-device inference performance of the SensorFCN model and exploring different model pruning strategies to further reduce model size.

References

[1] Alejandro Galán-Mercant, Andrés Ortiz, Enrique Herrera-Viedma, Maria Teresa Tomas, Beatriz Fernandes, and Jose A. Moral-Munoz. Assessing physical activity and functional fitness level using convolutional neural networks. *Knowledge-Based Systems*, page 104939, August 2019.

[2] Albert Haque, Michelle Guo, Alexandre Alahi, Serena Yeung, Zelun Luo, Alisha Rege, Jeffrey Jopling, Lance Downing, William Beninati, Amit Singh, Terry Platchek, Arnold Milstein, and Li Fei-Fei. Towards Vision-Based Smart Hospitals: A System for Tracking and Monitoring Hand Hygiene Compliance. *arXiv:1708.00163 [cs]*, August 2017. arXiv: 1708.00163.

[3] Yan Wang, Shuang Cang, and Hongnian Yu. A survey on wearable sensor modality centred human activity recognition in health care. *Expert Systems with Applications*, 137:167–190, December 2019.

[4] Henry Friday Nweke, Ying Wah Teh, Mohammed Ali Al-garadi, and Uzoma Rita Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, September 2018.

[5] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, March 2019.

[6] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors. *ACM Comput. Surv.*, 46(3):33:1–33:33, January 2014.

[7] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical Human Activity Recognition Using Wearable Sensors. *Sensors*, 15(12):31314–31338, December 2015.

[8] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-Aware Human Activity Recognition Using Smartphones. *Neurocomputing*, 171:754–767, January 2016.

[9] Wenchao Jiang and Zhaozheng Yin. Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1307–1310, New York, NY, USA, 2015. ACM. event-place: Brisbane, Australia.

[10] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, June 2015.

[11] Francisco Javier Ordóñez and Daniel Roggen. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1):115, January 2016.

[12] Nils Y. Hammerla, Shane Halloran, and Thomas Ploetz. Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables. April 2016.

[13] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235–244, October 2016.

[14] Andrey Ignatov. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Applied Soft Computing*, 62:915–922, January 2018.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*, November 2014.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[17] Jorma Laaksonen. CS-E4890 Deep Learning: Deep feedforward neural networks, October 2017.

[18] Antti Keurulainen. CS-E4890 Deep Learning: Optimization, November 2017.

[19] Ye Liu, Liqiang Nie, Li Liu, and David S. Rosenblum. From action to activity: Sensor-based activity recognition. *Neurocomputing*, 181:108–115, March 2016.

[20] Rubén San-Segundo, Henrik Blunck, José Moreno-Pimentel, Allan Stisen, and Manuel Gil-Martín. Robust Human Activity Recognition using smartwatches and smartphones. *Engineering Applications of Artificial Intelligence*, 72:190–202, June 2018.

[21] Yong Zhang, Yu Zhang, Zhao Zhang, Jie Bao, and Yunpeng Song. Human activity recognition based on time series analysis using U-Net. *arXiv:1809.08113 [cs, stat]*, September 2018. arXiv: 1809.08113.

[22] Pekka Siirtola, Ella Peltonen, Heli Koskimäki, Henna Mönttinen, Juha Röning, and Susanna Pirttikangas. Wrist-worn Wearable Sensors to Understand Insides of the Human Body: Data Quality and Quantity. In *The 5th ACM Workshop on Wearable Systems and Applications - WearSys '19*, pages 17–21, Seoul, Republic of Korea, 2019. ACM Press.

[23] Pekka Siirtola, Heli Koskimäki, and Juha Röning. From User-independent to Personal Human Activity Recognition Models Exploiting the Sensors of a Smartphone. *arXiv:1905.12285 [cs]*, May 2019. arXiv: 1905.12285.

[24] Charikleia Chatzaki, Matthew Pediaditis, George Vavoulas, and Manolis Tsiknakis. Human Daily Activity and Fall Recognition Using a Smartphone's Acceleration Sensor. In Carsten Röcker, John O'Donoghue, Martina Ziefle, Markus Helfert, and William Molloy, editors, *Information and Communication Technologies for Ageing Well and e-Health*, Communications in Computer and Information Science, pages 100–118. Springer International Publishing, 2017.

[25] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901 [cs]*, November 2013. arXiv: 1311.2901.

[26] Daniela Micucci, Marco Mobilio, and Paolo Napoletano. UniMiB SHAR: a new dataset for human activity recognition using acceleration data from smartphones. *arXiv:1611.07688 [cs]*, November 2016. arXiv: 1611.07688.

[27] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R. Millán, and Daniel Roggen. The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, November 2013.

[28] Mi Zhang and Alexander A. Sawchuk. USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 1036–1043, New York, NY, USA, 2012. ACM. event-place: Pittsburgh, Pennsylvania.