Information Processing and Management 59 (2022) 103104

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Predicting future sedentary behaviour using wearable and mobile devices

Martín Santillán Cooper [a], Marcelo G. Armentano [b,*]

[a] Facultad de Ciencias Exactas, UNICEN, Argentina
[b] ISISTAN Research Institute (CONICET-UNICEN) Campus Universitario, Paraje Arroyo Seco, Tandil (7000), Argentina

ARTICLE INFO

Dataset link: Sedentery behavior prediction (Original data)

Keywords:
Sedentary behaviour prediction
Machine learning
User modelling
Wearable and mobile devices

ABSTRACT

Sedentarism is a common problem that can affect human health and wellbeing. Predicting sedentary behaviour is an emerging area that can benefit from data collected from sensors available in ubiquitous devices, such as wearables and smartphones. In this paper, we present an approach aiming at predicting the sedentary behaviour of a user from data collected from sensors installed in wearable/mobile devices. We compare personal and impersonal models using a real-life dataset consisting of sensing data of 48 users during 10 weeks. We found that impersonal models using Deep Neural Networks were able to accurately predict the subject's future sedentary behaviour.

1. Introduction

The definition of sedentary behaviour has been evolving over the years, at the same time so did the way of measuring it. The Sedentary Behaviour Research Network[1] defines sedentary behaviour as any waking behaviour characterized by an energy expenditure $\leq$ 1.5 METs (Metabolic Equivalent of Tasks) while in a sitting or reclining posture. MET measures the intensity of an activity in multiples of resting energy expenditure. Examples of sedentary activities are watching television (1.0 MET), eating while sitting (1.5 MET) or playing video games (1.0 MET) and driving (1.3 MET).

Research done in the area demonstrates strong and consistent associations between sedentary time and diabetes, cardiovascular disease and all-cause mortality (Carter, Hartman, Holder, Thijssen, & Hopkins, 2017; Wilmot, et al., 2012). However, the reported associations were largely independent of physical activity. Thus, it is important to note that sedentary behaviour does not represent the opposite of physical activity and that it is possible for an individual to have high levels of both moderate to vigorous physical activity (MVPA) and sedentary behaviour. In general, the sedentary time has shown to be detrimentally associated with health outcomes and with markers of metabolic risk across diverse population groups. In addition, it has been highlighted the importance of not only stimulating MVPA but also reducing sedentary time, since sedentary behaviour is a risk factor for mortality independent of MVPA (Koster, et al., 2012).

Other research line focused on the associations between short breaks in sedentary time with metabolic outcomes (Paing, et al., 2018) and with optimization of cognitive operations (Falck, Davis, & Liu-Ambrose, 2017; Felez-Nobrega, Hillman, Dowd, Cirera, & Puig-Ribera, 2018; Magnon, Vallet, & Auxiette, 2018). Benatti and Ried-Larsen (2015) claimed that there is enough evidence to show the positive effects of breaking up prolonged time spent sitting on metabolic outcomes.

Methods for measuring sedentary behaviour can be classified as subjective (self-report questionnaires or diaries) and objective (using sensors of ubiquitous devices). Subjective methods are being surpassed by new technologies that can provide, for all

* Corresponding author.
E-mail addresses: mscooper@exa.unicen.edu.ar (M. Santillán Cooper), marcelo.armentano@isistan.unicen.edu.ar (M.G. Armentano).
[1] https://www.sedentarybehaviour.org/

https://doi.org/10.1016/j.ipm.2022.103104
Received 27 February 2022; Received in revised form 27 August 2022; Accepted 25 September 2022
Available online 17 October 2022
0306-4573/© 2022 Elsevier Ltd. All rights reserved.

population groups, second-by-second information on posture, movement (or lack of movement) and patterns within and between days (Atkin, et al., 2012).

The mobile health (mHealth) area attracted researchers attention in recent years thanks to the decreasing size of mobile devices, along with their increasing computing power and connectivity. MHealth can be defined as "the use of mobile communications and emerging network technologies for health care". MHealth has the ability to benefit communication between patients and health entities, provide health services and promote preventive health behaviours (Dutta, Kaur-Gill, Tan, & Lam, 2018). In this way, there is great potential for mHealth technologies to redesign almost all facets of healthcare (Steinhubl, Muse, & Topol, 2015). In particular, mHealth applications have been shown to have the potential to promote changes in sedentary behaviour and physical activity (Yerrakalva, Yerrakalva, Hajna, & Griffin, 2019).

Although mobile devices can be considered as one of the causes of sedentary behaviour (He & Agu, 2016b), they also offer new opportunities to prevent it. Nowadays, portable mobile devices, such as smartphones, smartwatches and fitness trackers are equipped with a wide variety of sensors that can be used for human activity and behaviour analysis. The use of objective methods to assess sedentary behaviour is growing in popularity as the costs of portable mobile devices decrease and are easier to use. In 2018, 91% of people between 18 and 49 years old in the US own a smartphone.[2] In this context, these devices can be seen as an opportunity for developing complex objective methods for measuring sedentary behaviour.

Many smartphone applications have been implemented with the aim of alerting the user when sedentary behaviour is recognized (Fahim, Baker, Khattak, & Alfandi, 2017; Grundgeiger, et al., 2017; He & Agu, 2014). Predicting future sedentary behaviour can help to enable preventive interventions such as reminders and suggestion for different activities based on Theory of Planned Behaviour (TPB) (Ajzen, 1991). TPB postulate that a subject is more likely to participate in recommended interventions to reduce sedentary behaviours if such activities are included in their plans. Following this idea, our hypothesis is that if we were able to predict at time t that a subject will be sedentary in time t+1, we could recommend activities aiming at changing his/her sedentary routine in a long term. This kind of interventions can lead to better opportunities for changing the subjects' behaviour to healthier outcomes.

We define future sedentary behaviour prediction (FSBP) as predicting whether the physical activity of a user will surpass, on average, 1.5 METs in the near future or not. The problem of predicting the future sedentary behaviour has previously been addressed by analysing only the inactive/stationary time of a subject and the performance of several models has been measured and compared. However, although MET is a standard metric in the area of health for measuring the intensity of an activity in terms of energy expenditure, the use of this metric for predicting sedentary behaviour by using wearables and mobile devices remains unexplored. In this paper, we present a novel approach for predicting the future sedentary behaviour of a subject in terms of its MET level (Sedentary Behaviour Research Network, 2012). In addition, we present a comparison between different models for FSBP.

To sum up, the research question that we address in this article is whether it is possible to predict the future sedentary behaviour of a subject, based on *lifelogging* data collected from the observation of values obtained from multiple sensors of wearable and mobile devices.

The remaining of this article is organized as follows. In Section 2.1 we present some related work in the use of wearable and mobile devices to predict human activity, focusing particularly on sedentary behaviour prediction. Section 2.2 defines Personal and Impersonal models and Section 2.3 describes the predictive models used in this research. Section 3 describes the dataset used for our experiments, detailing the data cleaning procedure and the definition of features from the raw data. In Section 4, we present the experiments performed to evaluate the proposed models, and in Section 5 we analyse in detail the results obtained. Finally, in Section 6, we present some insights and lessons learned from the experiments and in 7 we describe some of the limitations we faced and future work.

2. Background and related work

This Section is organized to cover three different aspects. First, in Section 2.1, we review the literature on the use of the sensors installed in mobile and wearable devices for studying different aspects of human behaviour. We specifically focus on research conducted in the fields of smart healthcare and activity prediction. Then, in Section 2.2, we present the two types of user models that we analysed in the context of FSBP: personal and impersonal. Finally, in Section 2.3, we describe the predictive models that we built and compared for the task of FSBP.

2.1. Mobile devices for smart healthcare and activity prediction

As mobile and wearable devices, became popular, many studies have been carried out to find usage patterns that allow correlating, inferring and predicting different types of human behaviours in the context of health and wellbeing. Harari, et al. (2017) demonstrated the viability of using smartphone sensing methods to track health-related behaviours in the context of students' daily lives. They identified a number of significant correlations between objective sensing data from smartphones and mental wellbeing and academic performance outcomes. Gong, et al. (2019), in a collaboration between engineers and psychologists, demonstrated a statistically significant relationship between smartphone sensing data and social anxiety levels of college students. From the experiments performed, the authors concluded that there is a significant difference in movement behaviours tied to social

[2] http://www.pewresearch.org/fact-tank/2018/09/28/internet-social-media-use-and-device-ownership-in-u-s-have-plateaued-after-years-of-growth/

anxiety level and that these behaviours vary across different semantic locations. These results open new possibilities for passively monitoring behavioural markers of social anxiety through the integration of the accelerometer and the GPS data. Kanjo, Younis, and Ang (2019) adopted a deep learning approach for emotion classification trained and tested using a dataset collected from smartphones and wearable devices in a real-world study. They achieved a classification accuracy of 95% using hybrid models composed by Convolutional Neural Network and Long Short-term Memory Recurrent Neural Network. Wu, Boukhechba, Cai, Barnes, and Gerber (2018) used Bluetooth encounter networks to predict their cognitive stress levels. The results indicated a potential value of incorporating Bluetooth encounter data into mental health monitoring practice via mobile sensing technology. Zia, Tadayon, McDaniel, and Panchanathan (2016) studied the viability of neural networks in predicting Freezing of Gait trained and tested with data collected from wearable devices. The aim of this research was to analyse if wearable devices could be used to help Parkinson's patients live safer and more independent lives. Particularly, they used a class of neural networks known as Layered Recurrent Networks and reached 42%, 89%, and 57% precision in predicting Freezing of Gait for the three participants in the study.

Activity Prediction is an area of research that consists of hypothesizing information about the activities that a subject will be involved in the near future. In this context, our research is focused on predicting future sedentary behaviour. Several authors have addressed a similar problem. Q. He and E. Agu proposed several models for FSBP. First, they proposed a frequency domain algorithm for detecting recurrent sedentary patterns from activity time-series data. In their experiment, subjects who exhibited recurrent sedentary behaviours yielded periodic functions with a Mean Square Error as low as 0.003817 for predicting recurrent sedentary behaviours (He & Agu, 2016a). In a second study, the same authors explored whether the contexts that can be sensed by users' smartphones can be used to predict their future sedentary behaviours reliably, in order to enable more effective interventions based on the theory of planned behaviour. By using logistic regression, they were able to classify user context variables to predict if the subject will be "very sedentary" in the next hour with a precision of 73.1% and a recall of 87.7% (He & Agu, 2016b). It is important to notice that the reported figures correspond to the prediction of only the "very sedentary" class. Parallelly, they proposed an approach to automatically discover patients' temporal patterns of sedentary behaviour from raw activity logs by using an autoregressive model with maximum entropy method. They have been able to model the three important individual determinants of sedentary behaviours: time, daily rhythm, and past sedentary habits (He & Agu, 2016c). Finally, they focused on detecting the prevailing rhythms of sedentary behaviours and modelling the cyclical rhythm and linear rhythm in Lefebvre's philosophy using periodic functions (history-free) and linear functions (history-dependent) respectively (He & Agu, 2017). In spite of the fact that these authors built several interesting models in their research, it is very important to note that they measured sedentary behaviour in terms of the average of sedentary records per hour. Instead, we follow the definition of the Sedentary Behaviour Research Network and we approximated the MET level thanks to the Compendium of physical activities (Ainsworth, et al., 2011). In addition, to the best of our knowledge, a comparison between personal and impersonal models (in terms of the data used to train the models) has been never done in the context of FSBP.

2.2. Personal and impersonal models

In this paper, we aimed at comparing personal and impersonal models for FSBP, taking into account the advantages and disadvantages of each of them. Lockhart and Weiss (2014), define impersonal models as those that "use training data from a panel of users that will not subsequently use the model". This definition implies that the training and test sets have no users in common. On the other hand, personal models are trained only with information taken from the same user for whom the model is intended. Therefore, personal models require a training phase to collect labelled data from each user. The training and test data come from the same user but contain different and disjoint examples.

Both, personal and impersonal models have its own advantages and disadvantages. On the one hand, impersonal models have the advantage that they can include data from many users to train the model. In consequence, it is possible to train impersonal models with large amounts of data collected from other users and complex non-linear non-parametric functions can be searched for FSBP. Additionally, impersonal models can be built once for all users, as they can be considered universal. Finally, impersonal models have the advantage of not suffering the problem of cold-start (Schein, Popescul, Ungar, & Pennock, 2002). The cold-start problem refers to the situation in which a new user is added to a system and there is almost no information about him/her. Impersonal models can be used to generate predictions to a new user without any additional labelled training data or model regeneration. Among the disadvantages of impersonal models, we can highlight that building a model with data from many users and using it to classify activities of a target user may be prone to introduce noise due to the diversity among users. In consequence, users with particular behaviour patterns should have a better performance with personal models than with impersonal models.

On the other hand, personal models have the advantage of matching the idiosyncrasies of each user, at the cost of requiring each user to provide training (labelled) data. However, personal models suffer from the cold-start problem since the amount of data available for new users is scarce.

2.3. Predictive models

The problem addressed in this article is that of predicting future sedentary behaviour using Deep Learning models trained on *lifelogging* data. Energy expenditure is measured in METs, which is the standard measurement in the scientific community that studies health in relation to physical activity. In the field of health, an agreement has been reached among researchers in determining as sedentary activity all that activity whose associated MET is less than or equal to 1.5. As we defined in Section 1, the prediction of future sedentary behaviour (FSBP) is defined as the task of predicting the MET value of the physical activity performed by a user in the near future in order to study their future sedentary behaviour. Although we could consider this task as a binary classification problem by considering 1.5 MET as the threshold for sedentary behaviour, we preferred to approach a regression problem and predict the MET level for the next time bucket by using information of the previous observed behaviour, without imposing any decision boundary. In this article, we evaluate the feasibility of using different *deep learning* architectures to approach this problem.

The first architecture that we analysed is *Multilayered Perceptron (MLP)*, which is the simplest architecture in the field of deep learning. The main disadvantage of using this type of network for sequence modelling is that it completely ignores the topology of the input, that is, timesteps and features are represented in the same dimension, so it is not possible to instruct the network to take these representations into account. In other words, the input variables could be presented in any order without affecting the training result. In this work, MLP was considered in order to evaluate whether using architectures that take into account the sequential structure of the data lead to an improvement in the predictive accuracy of the models.

Recurrent Neural Network (RNN) use architectures where the input of the network are sequences. RNN units maintain an internal state that is propagated through time. This internal state represents the memory of the units and can act as a representation of everything that the network has seen in the input stream until a given moment. In short, RNNs have as input a vector that represents a sequence (for example: a sentence) and the neurons or units process each element of that vector, one at a time and iteratively. RNNs are computationally represented by cyclic graphs, where cycles represent the influence of the current value of a variable on its value at a future time. Thus, the order in which the input stream is processed is important and must respect its natural order (for example, that of time, or from left to right in the case of text processing) since the network is designed to take advantage of that order. Differently, in MLPs each neuron receives all the information *at the same time*, preventing the sequential order from being considered. Another important difference with respect to MLPs is that RNNs share the same parameters for each component of the sequence, where MLPs have different parameters for each part.

Convolutional Neural Networks (CNN), are very common in image processing and have several differences with respect to MLPs. On the one hand, CNNs take clearly structured data as input, in which nearby variables are strongly correlated (for example, images). On the other hand, this type of network greatly reduces the number of parameters needed to be trained, since these are shared by all the neurons of a given layer. These parameters are called *filters*, and by applying the convolution to the network input or to an intermediate layer, they extract local features that are then successively combined to generate global features. Therefore, such filters can be interpreted as features that grow in complexity with each layer of the network. Commonly, between each convolution layer, a method called *pooling* is applied, which reduces the number of neurons and, therefore, the resolution of the processed data, obtaining translation invariance and improving the generalization capacity of the network.

Temporal Convolutional Networks (TCN), are a type of CNN architecture with certain characteristics that make it especially applicable to tasks where the model input is a sequence. Bai, Kolter, and Koltun (2018) carried out a systematic evaluation of convolutional and recurrent architectures. The results showed that the common association between sequence modelling and RNNs should be reconsidered, since TCNs were the architectures that achieved the highest performance in the benchmarks. The most important differences between generic TCNs in Bai et al. (2018) and traditional CNNs are :

- TCNs do not use pooling layers, so the input and output dimensions are the same.
- TCNs always apply the convolution of a single dimension (that of time).
- The convolutions carried out by the TCN are causal, that is, the convolutions never consider future information.
- They use dilations to increase the receptive field exponentially and thus be able to cover long dependencies on sequential data.

There are several advantages of TCNs over RNNs that lead to a better performance in different benchmarks (Bai et al., 2018):

- Unlike RNNs where predictions for a future timestep have to wait until all previous predictions have been carried out, convolutions can be carried out in parallel since the same filter is used for all of them. This allows both training and evaluation of the network to be carried out more quickly.
- It is easy to adjust the receptive field of the TCN from the variation of the size of the filter and the value of the dilations, which allows a better control of the size of the model memory, allowing easy adaptation for different domains.
- Since in TCNs the backpropagation path is different from the temporal direction of the sequence, they do not suffer from the so-called exploding/vanishing gradient problems.
- When training, the RNNs we must keep in memory the partial results of each layer, which may require a lot of memory, especially for long sequences. This fact does not occur in TCNs because the filters are shared by the entire layer.
- TCNs can, like most RNNs, receive variable-length sequences.

In Bai et al. (2018) two disadvantages of the use of TCN are also shown:

- At evaluation time, TCNs must keep the entire sequence in memory to make a prediction. Instead, RNNs need only keep a timestamp of the entire sequence in memory.
- If we want to change the domain from an already trained model, for example, to perform fine-tuning, the receptive field required by the new task to be performed may be greater than the receptive field of the already trained model, so the model may not perform well on the new task.

When training a Deep Learning model, there are different hyperparameters that must be adjusted to ensure that the model reaches the highest possible performance. For this, different values must be tested for each hyperparameter, comparing the performance obtained by training the model with each of them. There are different ways to perform the comparison of the different values of the hyperparameters. In this article, we use Bayes Optimization was used (Agnihotri & Batra, 2020) to perform the tuning or search for the optimal hyperparameters.

Hyperparameters can be settings related to the learning algorithm or the architecture of the model. Therefore, each model will have specific hyperparameters. The hyperparameters that are part of all the models used are detailed next:

- *Number of layers*: Deep Learning models consist of a sequence of layers, where each layer is made up of neurons. This hyperparameter defines how deep the model is, in relation to the number of intermediate relationships it will be able to learn.
- *Number of neurons*: each layer of the model consists of neurons, where each neuron performs some processing from the output of neurons in the previous layer. The number of neurons in one layer may differ from the number of neurons in another layer. In addition, the input of a neuron and the processing carried out by it are highly dependent on the type of model it belongs to.
- Activation function: is the function that is applied to the output of neurons belonging to the hidden layers of the neural network. The activation function is intended to introduce non-linearities to the neural network. That is, let z be the non-linear activation function. Typically, the value of a neuron is the sum of its inputs weighted by their respective weights, i.e., a real number. Therefore, $z : z(x) = y$ is a function such that $x \in \mathfrak{R}$ and $y \in \mathfrak{R}$, where z cannot be expressed as a linear function, of the form $ax + b$. The activation function that has become the standard nowadays is called ReLU (Rectified Linear Unit) where $z(x) = max(0, x)$. This function has the property of being "almost" linear, since it is composed of two linear functions, although it is still non-linear. In this way, it shares certain properties with linear functions, which makes models using it easy to optimize using gradient-based optimization methods (Goodfellow, Bengio, & Courville, 2016).
- *Number of epochs*: number of iterations the complete training dataset is run through in the training process.
- *Batch size*: number of training cases that will be taken from the training dataset to perform one iteration of the learning algorithm. The larger the batch size, the fewer the number of iterations that must be performed at each epoch. The learning algorithm divides the training dataset into n batchs. Then, for each batch, the model error is computed and the parameters are updated using gradient-based optimization. The advantage of updating the parameters at the end of each batch rather than after traversing the entire training dataset is that it allows the parameters to be updated more frequently, which allows the optimization function to converge more quickly. If the number of batch is 1, the number of parameter updates are the same as the number of epochs.
- *Learning rate*: defines the factor by which the model parameters are updated relative to the computed gradient at each iteration. Let θ be the model parameters, δ the computed gradient and ϵ the learning rate, then the model parameters are updated as $\theta \leftarrow \theta - \epsilon\delta$ at the end of each iteration.

3. Dataset analysis and processing

In this section, we describe the dataset used to validate our models, the pre-processing applied to the raw data, and the features extracted.

3.1. Dataset description

We analysed and preprocessed the StudentLife Dataset (Wang, et al., 2014) in order to prove the validity of our models. The dataset was collected from 30 undergrads and 18 graduate students over a 10-week term in spring 2013. The students consisted of 38 males and 10 females. Two of them were first-year, 14 second-year, 6 third-year, and 8 fourth-year Bachelor's students. There were also 13 first-year and 1 second-year Master's student, and 3 Ph.D. students. Participants were racially diverse, with 23 Caucasians, 23 Asians, and 2 African-Americans. Available data included:

- **Activity data**, including activity duration (total time that the user moves per day), indoor mobility and the total travelled distance (i.e., outdoor mobility) per day;
- **Conversation data**, including conversation duration and frequency per day;
- **Sleep data**, including sleep duration, sleep onset and waking time; and finally
- **Location data**, including GPS, inferred buildings when the participant is indoors, and the number of co-located Bluetooth devices.

The StudentLife Dataset was collected using the StudentLife app, which is a smartphone application and sensing system that automatically infers human behaviour in an energy-efficient manner. StudentLife app uses a continuous sensing engine, which balances the performance needs of the application and the resource demands of continuous sensing on the phone.

Different types of sensing information were logged with different frequencies, depending on the sensor type and the global workload of the phone. For example, activity logs are sampled every 2–3 s in 1 of every 4 min, smartphone app usage was logged every 20 min, and location logs were sampled every 10 min.

The StudentLife Dataset has been used in several fields, both in human activity learning and prediction (Chen, Wang, Zhou, & Campbell, 2014) and in mental wellness assessment (Harari, et al., 2017; Saeb, Lattie, Schueller, Kording, & Mohr, 2016; Wang, et al., 2017). This dataset is, to the best of our knowledge, the best suited for studying and analysing models for FSBP, due to its 10-weeks duration (which let us analyse how sedentary behaviour change along time) and its wide variety of sensing data. Furthermore, this dataset was used in several works for FSBP.

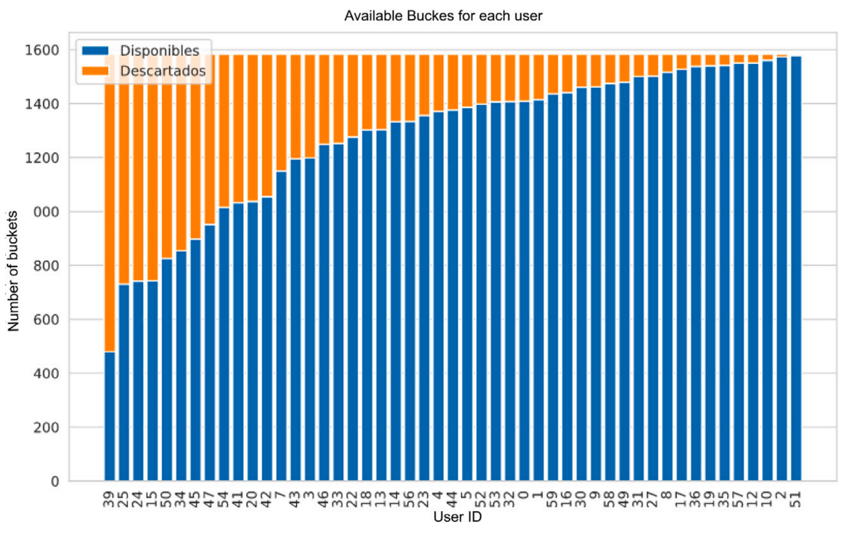

Fig. 1. Number of available and discarded buckets for each user.

3.2. Data cleaning

All of the sensing data in the StudentLife Dataset was tagged with a timestamp. Although this is the case for most continuous sensing data types in the dataset, some records have another format, in which there are two timestamps that indicate the start and end times of an event. When referring to each type of sensing data, we will refer to Discrete Sensing Type (DST) and Interval Sensing Type (IST). For example, *activity* records (stationary, walking, running or unknown) are DST records in the dataset, while *light* or *conversation* records are IST.

We decided to discretize the time-series into one-h buckets since this is the granularity used in most of the related work for sedentary prediction. Then, all the features that were generated from the raw dataset correspond to a particular user/hour combination. For example, a specific 1-hour-bucket might correspond to subject 10 and hour 2013-04-24 19:00–20:00. We also considered a discretization in half-hour buckets to compare the impact on learning model performance of roughly doubling the number of cases generated from the original dataset. The disadvantage of this is that there will be more buckets that will have to be discarded for not containing records of any of the data types. As a consequence, the models that are trained from the use of several consecutive buckets (for example: CNN and RNN) will be affected because there will be more cases in which a sequence of buckets must be discarded due to the absence of one or more records.

Fig. 1 shows the number of resulting buckets available for each user and the number of buckets discarded due to the lack of records for some variable, after purging the preprocessed dataset. As can be seen, there is great variability regarding the number of buckets available for each user. The user with the fewest number of buckets available is user 39, with 479 buckets, while the user with most buckets available is user 51, with 1579.

In the dataset, students' activity records are labelled as *stationary*, *walking*, *running* and *unknown*. We discarded the physical activity logs labelled with unknown, since they do not provide enough information regarding the activity that was being performed by the subjects. Then, we only kept the buckets in which there were available logs of the physical information of the subject since the sedentary level is computed from it. Since our goal is to predict future sedentary behaviour we also removed the buckets for which we had no physical activity information of the next hour.

For the remaining buckets, we computed the sedentary level, in terms of MET, from the physical activities performed by each subject. We established the MET level of each physical activity according to the Compendium of physical activities[3] (Ainsworth, et al., 2011). For those activities with "stationary" labels in the dataset, we set a MET value of 1.3 (corresponding to activities such as "sitting quietly", "lying quietly", "doing nothing", "lying in bed awake", "listening to music (not talking or reading)" in the Compendium). For those activities labelled as "walking" in the dataset, we set a MET value of 5.0 (corresponding to activities such as "carrying 15-pound load – e.g. suitcase – level ground or downstairs" in the Compendium). Finally, for activities labelled as "running" in the dataset, we set a MET value of 8.3, which correspond to activities such as "running, 5 mph (12 min/mile)" in the Compendium.

Once we associated MET values to each type of activity, the information available on the activity (ordinal categorical variable) was transformed into a numerical variable. Then, we defined how the physical activity information was going to be summarized – expressed in METs – for each bucket, so that the result was a real number. That is, since all the available data is grouped into

[3] https://sites.google.com/site/compendiumofphysicalactivities/

buckets, a representative MET value must be computed for each of them. To do this, the MET level of a bucket is defined as the average MET value of each of the activity logs belonging to that bucket. For example, if a 1-hour-bucket contains 100 stationary logs, 10 walking logs and 10 running logs, the MET level of that bucket is calculated as $(100 * 1.3 + 10 * 5 + 10 * 8.5)/120 = 2.2$.

As we explained previously, a behaviour is normally considered sedentary if it exceeds the value of 1.5 METs. The previous statement is adapted in this article as follows: *the behaviour of a subject for a given bucket is considered sedentary if the MET average level for that bucket is greater than 1.5*.

It is worth noticing that the low granularity in the type of physical activity specified in the StudentLife Dataset represents a limitation, since the Compendium of Physical Activities associated MET levels to activities in a higher granularity. Then, the MET value associated to each activity in the StudentLife Dataset is the best approximation that could be made with the information available.

3.3. Features definition

Since we discretized the available data into buckets, we had to hypothesize which information was going to be useful to predict the next hour sedentary behaviour. The features used were chosen based on the data available in the StudentLife Dataset and on previous research (He & Agu, 2016b).

Although subjects in the dataset were students, we avoided to include data that might only be present in this kind of subjects, for example, the time remaining for a scheduled exam, if he/she is attending to a given class, etc. Therefore, in order to make our approach more general, we only defined features that might be computed from data collected from other population.

We describe next the smartphones sensed context variables selected and computed by processing the StudentLife Dataset.

GPS features

- locationVariance: used to measure the variability in a participant's GPS location, computed as described in Saeb, et al. (2015) as
  $locationVariance = \log(\sigma^2_{lat} + \sigma^2_{long})$.
- locationMean: similar to locationVariance, but w.r.t. mean
  $locationMean = \log(mean_{lat} + mean_{long})$
- speedMean: average instant speed, as computed in Saeb, et al. (2015)
  $speedMean = \frac{1}{N}\sum^{N}\sqrt{(\frac{lat_i - lat_{i-1}}{t_i - t_{i-1}})^2 + (\frac{long_i - long_{i-1}}{t_i - t_{i-1}})^2}$
- speedVariance: variance of instant velocity, as computed in Saeb, et al. (2015)
- totalDistance: the total distance travelled, computed as
  $totalDistance = \sum^{N}\sqrt{(lat_i - lat_{i-1})^2 - (long_i - long_{i-1})^2}$

Time features

- pastMinutes: number of minutes elapsed from the beginning of the day
- remainingMinutes: number of remaining minutes in the day
- secondSine: sine transformation of the number of seconds elapsed from the beginning of the day
  $secondSine = \sin(\frac{2\cdot\pi\cdot seconds}{24\cdot 60\cdot 60})$
- secondCosine: cosine transformation of the number of seconds elapsed from the beginning of the day
  $secondCosine = \cos(\frac{2\cdot\pi\cdot seconds}{24\cdot 60\cdot 60})$
- weekDaySine: sin transformation of the day of the week
  $weekDaySine = \sin(\frac{2\cdot\pi\cdot dayofWeek}{7})$
- weekDayCosine: sin transformation of the day of the week
  $weekDayCosine = \cos(\frac{2\cdot\pi\cdot dayofWeek}{7})$

These last 4 features are intended to give the models the ability to understand the cyclical nature of time-related variables. Without performing this transformation, machine learning models do not have the necessary information to understand the fact that between the first possible value and the last there is the same distance as between any other two consecutive values. That is, in the case of days of the week, the difference between Tuesday and Wednesday is the same as between Sunday and Monday.

Physical activity features

- stationaryCount: the number of activity records classified as *stationary* in each bucket;
- walkingCount: the number of activity records classified as *walking* in each bucket;
- runningCount: the number of activity records classified as *running* in each bucket;
- totalActivityCount: the total number of records in each bucket;
- activityMajor: the most frequent activity in each bucket;

Audio features

- silenceCount: the number of audio records classified as *silence* in each bucket;
- voiceCount: the number of activity records classified as *voice* in each bucket;
- noiseCount: the number of activity records classified as *noise* in each bucket;
- totalAudioCount: the total number of records in each bucket;
- numberOfConversations: the number of conversations that the student had in each bucket;

Other features

- wifiChanges: the number of wi-fi changes in each time bucket.

Interval sensing type features

- isLocked: whether the smartphone was locked;
- lockedRatio: the percentage of the bucket during which the smartphone was locked
- isInDark: whether the smartphone was in a dark place (for example in a pocket);
- darkRatio: the percentage of the bucket during which the smartphone was in dark;
- isInConversation: whether the user was or not in a call;
- conversationRatio: the percentage of the bucket during which the user was in a call;
- conversationNumber: the number of different calls of the user in the bucket;
- isCharging: whether the smartphone was being charged;
- chargingRatio: the percentage of the bucket during which the smartphone was being charged;

To compute the ratios, each interval that exceeds the bucket size is divided iteratively into new smaller intervals, until all the intervals have a size less than or equal to the bucket size. Then, for each interval, the proportion it occupies of the bucket to which it belongs is calculated. Then, all the intervals are divided into groups according to the user's id and the bucket they belong to (for example: if the buckets are one-hour long and the interval starts at 14:34:20, the floor function is used to obtain the corresponding bucket, which returns 14:00:00). Finally, we sum the proportions of each interval belonging to each particular group.

The objective variable is *sLevel*, and represents the MET value for each bucket.

As a result, we obtain a preprocessed dataset in the form of a matrix with a dimension of 76,032 rows and 32 columns for 1-h time buckets and 152,064 rows by 32 columns for the half an hour buckets.

3.4. Lags definition

The next processing step is to generate the datasets with lags. In this processing stage, the dataset is prepared to be used to train and test the deep learning architectures. Two parameters are involved in this step: the number of lags and the value of the periods. Different values for these parameters will be evaluated in order to study which of them achieves the best performance for each architecture.

Each buckets b is considered to represent the present moment. Next, we find the list of buckets older than b by considering the number of lags and the value of the period. This list will define the information that the models will use to predict the MET level of the present moment, that is, the MET value of b. The selected buckets are then concatenated to the current bucket, while removing all features from the current bucket except for the target variable. The number of time units that each record in the dataset spans is $\#lags \times granularity$. The time interval covered by each record of the dataset is given by $\#lags \times granularity \times period$. Therefore, the time interval covered by each record is greater than or equal to the number of time units for which it has information. Another important issue to notice is that the number of features in each record is $\#characteristics \times \#lags$, so the number of features grows directly with the number of lags. This can cause problems if the number of training examples is low, where the dimensional space is so large that not enough training cases are available to have regularly spaced examples so that the dimensional space is relatively occupied. Thanks to the introduction of the period, the models can obtain information from the distant past without increasing the number of features.

Fig. 2 shows three examples of records from different datasets with lags. Green buckets represent buckets used by the model, while the pink bucket represents the present moment (the one from which the model will try to predict the MET level). In Fig. 2a the dataset has eight lags and a period of 1. In Fig. 2b the dataset has two lags and a period of 2. In Fig. 2c the dataset has four lags and a period of four. Assuming a granularity of 1 h, we can easily see the effect of using different values for the period. For example, although in Fig. 2a the dataset has twice as many lags as in Fig. 2c, the time interval that is covered in Fig. 2c is 16 h, that is half that in Fig. 2a. The buckets selected in each case are concatenated and given as the input to the model to make a prediction of the MET b_p, lets call it $\hat{y}$. The prediction is then compared to the real MET value to compute an error function, such as MSE.

In practice, the process of generating the datasets with different lags is carried out as a translation of the entire dataset in a single operation and not iteratively, since in this way the process is parallelizable and faster. Then, a list with different times, for example [16, 12, 8, 4] in the case of Fig. 2c, is generated from the number of lags and the value of the period. Then, for each element of that list, the translation of the dataset is performed and, finally, all the datasets are concatenated.

As an output of this pre-processing stage, a dataset is obtained where each record is made up of the target variable that represents the present moment and the features of one or more buckets from the past. The number of rows will vary according to the number of lags and the value of the period. Different combinations will have differences in the number of available rows. Also, the number

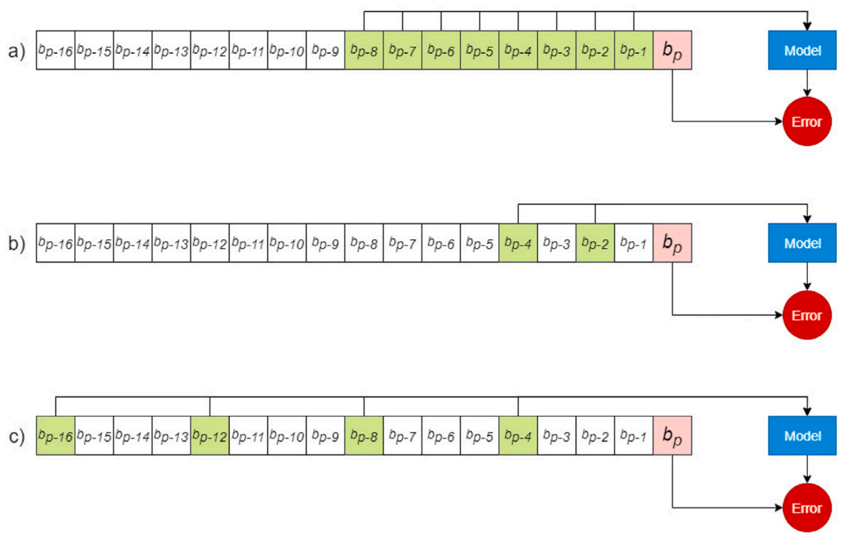

Fig. 2. Example of time buckets with different lags. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of lags will determine the number of columns in the matrix. At this point, the dataset is represented by an array of $n \times (36 \times \#lags) + 1$ dimensions, where n is the number of rows. For example, if the number of lags is 4 and the period is 2, we have an array of $57,493 \times 145$ for a granularity of one hour and $117,295 \times 145$ for a granularity of 30 min.

4. Experiments

In this section, we describe the experiments that were carried out to test our predictive models.[4] We first describe the methodology (Section 4.1), the selection of prototype users (Section 4.2), the training, testing and validation procedure (Section 4.3) and the evaluation metrics (Section 4.4) used. Finally, in Section 4.5 and Section 4.6, we describe the results obtained.

4.1. Methodology

It would be ideal for the tuning process to be carried out for each different combination of the aspects involved, which is not feasible due to the high number of models that should be evaluated. For this reason, we decided to take certain experiments as proxies for the others. The decisions that we made are as follows:

- Both personal and impersonal models are explored.
- The tuning process is performed for each type of proposed architectures (MLP, CNN, TCN and RNN).
- Besides, two representative users will be selected. This selection is more complex than the others, and it will be detailed in Section 4.2.

In this way, the number of total architectures resulting from the tuning process will be $2 \times 4 \times 2 = 16$. We decided to give 100 iterations to each of the tuning processes. Once the 100 iterations have been carried out, the set of hyperparameters for which the model achieved the highest performance is taken. The function that the Bayes optimization algorithm tries to minimize in this case is the model error for an experiment. That is, the complete pipeline of a normal experiment is carried out and the average of the performance achieved in each of its iterations is taken.

In this article, we intend to evaluate different aspects related to the FSBP task with a deep learning approach: the type of neural network, different time features when grouping records in the dataset (number of lags, granularity and period) and the nature of the models (personal vs impersonal). A list of values is then proposed for each aspect and each of those values will be tested for each of the users. Table 1 shows all the categories in which the experiments can be classified, as well as the possible values that each of these characteristics can take. The number of experiments that we should perform to evaluate different combinations of the characteristics is then $4 \times 2 \times 4 \times 3 \times 2 \times 48 = 9216$. For these reason, to reduce the last dimension (that of users), we select prototype users for evaluating the prediction models, as detailed in Section 4.2.

[4] All the python code used in this article is available at https://github.com/martinscooper/sedentery-behavior-prediction.

Table 1
Experiments to be performed.

Category	Network architecture	Model nature	Time features		
			Lags	Periods	Granularity (minutes)
Values	MLP, RNN, CNN, TCN	Personal, Impersonal	1, 2, 4, 8	1, 2, 4	30, 60

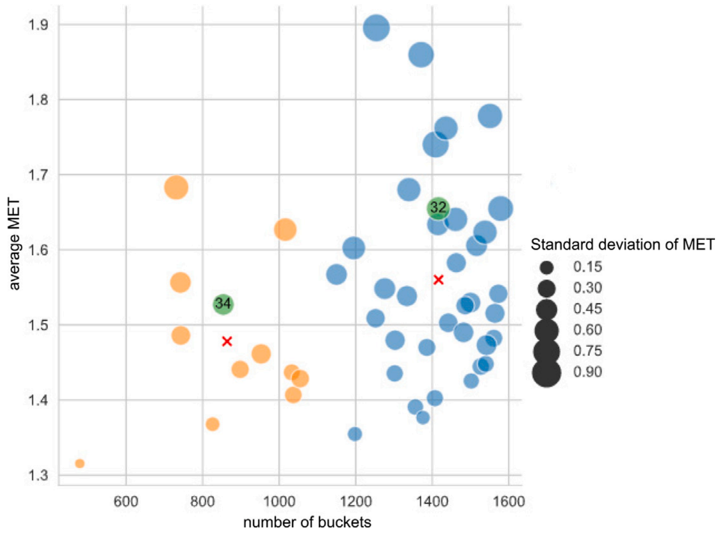

Fig. 3. Users clustering and prototype users. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. Selection of prototype users

The idea behind selecting prototype users is to represent different energy expenditure patterns when finding the best hyperparameters for different architectures and datasets, while greatly reducing the time required for the tuning process.

To select prototype users, we used the unsupervised K-Means machine learning method, which seeks for finding centroids from certain features calculated for each user. The features that we selected to cluster users are the average MET, the MET standard deviation, and the number of 1-h buckets available. K-means generates two user groups based on the given features. The user closest to the centroid of each group is then taken as the prototype user.

Fig. 3 shows the results of clustering the users into two groups. The x-axis represents the number of 1-h buckets available, while the y-axis represents the average MET. The radius of each circle of the plot represents the standard deviation of the MET value. The larger the circle, the larger the standard deviation of the MET level for each user in each bucket. Selected users are displayed in green: user 34, which represents users with low average METs and low number of available buckets, and user 32, which represents users with high number of buckets and high average METs. Finally, the red crosses show the exact location of the centroids computed by K-Means. Notice that, for clarity, only the first 2 dimensions of the centroids are plot, so it is not apparent from the plot that users 32 and 34 are the users closest to each centroid.

4.3. Models training, testing, and validation

Since each record in the dataset have an associated timestamp (*time-series*), classical techniques for splitting the dataset into training and testing, such as hold-out or cross-validation, are not suitable for this problem. Classical evaluation methods assume that the variables in the dataset are independent and identically distributed, which cannot be assumed in the case of time-series, since records cannot be exchanged with each other without changing the distribution of the dataset. Furthermore, the evaluation method to be used must be adapted to the actual way in which the data are recorded and made available to the models. In other words, all the data used to train a model must be prior to the data used to test it. In this way, a real situation is simulated in which future data are not yet available. Finally, the method to be used must be adaptable to both personal and impersonal models. This is necessary because the data used by the models to be trained and tested may or may not belong to the same dataset. A common method used to evaluate models trained from time-series datasets, is known as *Time Series Split*. Time Series Split consists of dividing the dataset into k partitions, respecting the temporal order, where each partition has the same number of cases. Then, $k-1$ iterations

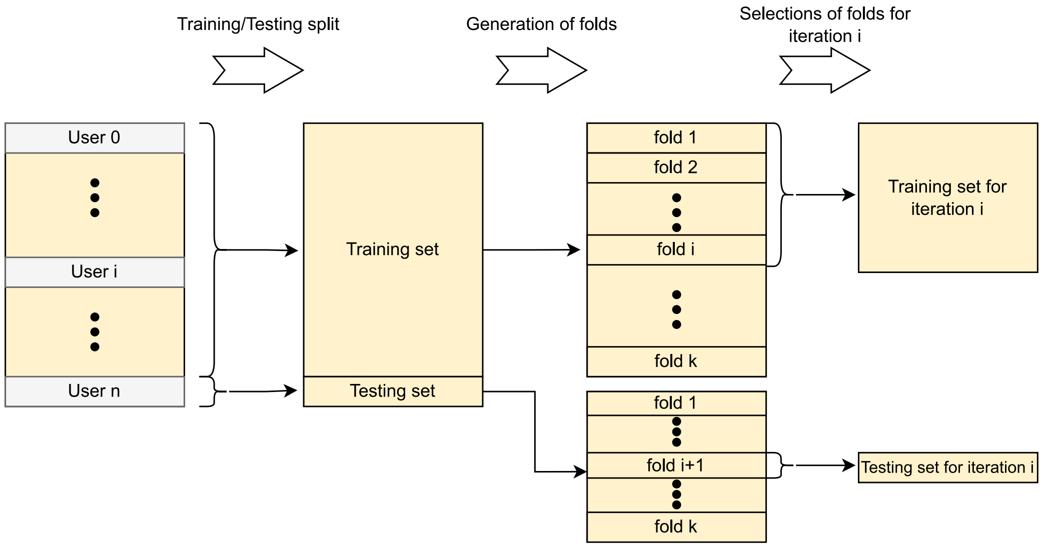

Fig. 4. Validation procedure for impersonal models.

are carried out in which the model is re-trained, with the particularity that in each iteration the set of training cases is a superset of the set of cases of the previous iteration. More precisely, the set of training cases of iteration i is equal to the union of the training and test sets of iteration $i-1$. Then, in iteration i the first i partitions are used to train the model and partition $i+1$ is used to test it. The final result of this validation method is the average performance achieved by the model on the test dataset at each iteration. In some cases, the standard deviation can be a useful statistic.

Time Series Split is not suitable for our purposes for two main reasons. First, it assumes that cases are observations in fixed time intervals, which is not applicable to datasets obtained from the preprocessing pipeline, where two time intervals of equal length may contain a different number of cases. Second, in order to include both personal and impersonal models, it is necessary that the evaluation method can perform training and testing of the models from two different datasets. This requirement is given in the case of impersonal models, where the training dataset belongs to a dataset that includes data on many users, while the testing dataset contains data on a single user.

Therefore, a new method based on Time Series Split is proposed to face the described problems. The proposed evaluation method differs in the following from the Time Series Split method:

- The criterion for generating the partitions is that they must represent the same time interval and not the same number of cases. For this, the minimum and maximum dates of the dataset are calculated and the length of that time interval is divided by the number of iterations, thus obtaining the time interval for each partition. Once these partitions are obtained, we proceed in the same way as in Times Series Split.
- To allow the method to adapt to personal and impersonal models, it receives two different datasets as parameters, one for training and one for testing. Thus, in the case of personal models, the training and testing datasets belong to the same user, while in the case of impersonal models the training dataset contains the data of all users except the target user while the testing dataset contains the data of the target user.

The steps followed by the validation algorithm vary according to whether the model is personal or impersonal, as this determines from which users the training and test data come. Fig. 4 shows the validation procedure for an impersonal model. In this case, we start with the entire dataset, which has the data of all users. The first step consists of selecting the training and test datasets, where the latter corresponds to the data of one user and the former to the data of all other users. In this case, the test dataset corresponds to the data of user n and the training dataset to the data of the rest of the users. Then, k partitions of each of the datasets are generated. The partitions with the same index of both datasets correspond to the same time interval. Finally, the training and test datasets for iteration i are determined. The training dataset for iteration i will be composed of the union of all partitions with index less than or equal to i, of the form $\bigcup_{t=1}^{i} Partition_t$, of the full training dataset. The test dataset of iteration i will be composed of the $i+1$ partition of the complete test dataset.

Fig. 5 shows the validation procedure for a personal model. Similar to impersonal models, we start from the entire dataset and select the subset of records corresponding to the target user. Next, the k folds are generated. Finally, for iteration i the first i folds are selected to train the model and fold $i+1$ is used for testing purposes.

4.4. Evaluation metrics

The metric used to evaluate the performance of the proposed models is the Mean Squared Error (MSE), which is a quantitative performance measure commonly used to measure the error between the actual value and the estimated value. In this context MSE

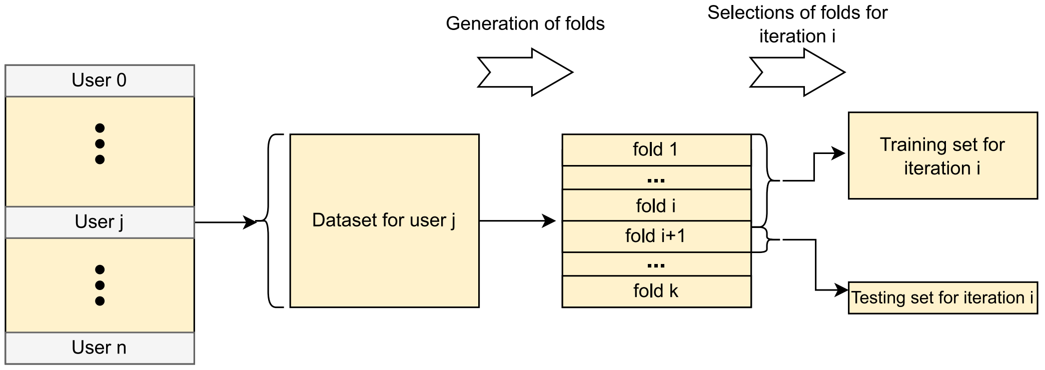
Generation of folds
Selections of folds for iteration i
User 0
User j
User n
Dataset for user j
fold 1
...
fold i
fold i+1
...
fold k
Training set for iteration i
Testing set for iteration i

Fig. 5. Validation procedure for personal models.

Table 2
Result of the tuning process for MLP.

User	Model	Nodes	Layers	Batch norm?	Dropout	Epochs	Batch	MSE
34	personal	256	16	Yes	0.800	64	8	0.252
	impersonal	32	2	No	0.743	64	128	0.211
32	personal	4	4	No	0.368	64	8	0.361
	impersonal	32	2	Yes	0.374	64	128	0.308

is equal to the sum of the squared errors (Eq. (1)).

$$MSE = \frac{1}{N} \sum_{t-1}^{n} (Y_t - \hat{Y}_t)^2 \tag{1}$$

where Y_t is the actual MET value at time t and $\hat{Y}_t$ is the forecasted MET value at time t. Compared to the Mean Absolute Error (MAE), MSE penalizes those errors of greater magnitude.

It is interesting to note that the metric selected to evaluate the models is the same that the loss function which is optimized when training these models.

4.5. Results of the tuning process

This section shows the results obtained in the tuning process used to find the best combinations of hyperparameters for each of the proposed architectures. In this process we used Bayes Optimization, which uses a Gaussian Process to decide which combination of hyperparameters is more convenient to adjust in each iteration. To this aim, we used the implementation provided by the Scikit-Optimize[5] library. Each tuning process is given up to 100 evaluations of different hyperparameter configurations. These processes can be terminated before 100 evaluations in case the surrogate model converges. Finally, the result of the tuning process will be the hyperparameter configuration that has achieved the best performance.

Each of the proposed architectures has different hyperparameters to be adjusted, so we will present a subsection for each architecture. Likewise, there are certain hyperparameters that are shared by all the architectures: the number of epochs and the size of the batch, since they are not related to the architecture of the model but to the number of times the training dataset is iterated and every how many training cases the model parameters are updated. In all cases, Adam was used as the optimization function and MSE as the loss function. In total, 16 model experiments were selected. In this section, they are presented categorized by the type of neural network, while for each type of neural network the results of the associated models are shown. The other categories that vary are the nature of the model (personal or impersonal) and the prototype user (user 32 or 34).

4.5.1. MLP

MLP architectures are the simplest of all, so they share some hyperparameters with other architectures. The hyperparameters that were adjusted for the MLPs were the number of dense layers, the number of units per layer, whether or not to use Batch Normalization, and the dropout value. Table 2 shows the results of the tuning process associated with the MLPs.

[5] https://scikit-optimize.github.io/stable/

Table 3
Result of the tuning process for CNN.

User	Model	Filters	Kernel size	Dropout (conv)	Dense nodes	Dropout (Dense)	Epochs	Batch	MSE
34	personal	256	4	0.00	16	0.80	64	8	0.269
	impersonal	256	16	0.80	64	0.306	64	256	0.210
32	personal	128	8	0.041	8	0.589	64	8	0.383
	impersonal	4	16	0.00	64	0.80	32	256	0.305

Table 4
Result of the tuning process for TCN.

User	Model	Filters	Kernel size	Dropout	Skip connect?	Use batch norm?	Epochs	Batch	MSE
34	personal	16	16	0.800	yes	yes	256	8	0.258
	impersonal	64	4	0.800	no	yes	64	128	0.216
32	personal	4	4	0.800	no	yes	256	8	0.384
	impersonal	4	16	0.506	no	yes	256	128	0.307

Table 5
Result of the tuning process for RNN.

User	Model	Layers	LSTM units	Dropout(LSTM)	densenodes	Dropout(dense)	Epochs	Batch	MSE
34	personal	2	4	0	64	0.447	16	8	0.245
	impersonal	4	4	0	1	0.800	32	8	0.210
32	personal	4	128	0	64	0.561	8	8	0.342
	impersonal	2	4	0	32	0	8	8	0.300

4.5.2. CNN

In the case of CNN, the hyperparameters that were adjusted were the number of filters, the size of the kernel and the dropout applied to the convolution layer. Typically, this architecture is used in combination with one or more dense layers. Therefore, we decided to add an optional dense layer as the last layer of this architecture. Consequently, the number of dense units also had to be tuned, as well as the dropout value for this layer. Table 3 shows the results of the tuning process associated with the CNNs.

4.5.3. TCN

Since TCNs are similar to CNNs, they share certain hyperparameters. In this architecture, the hyperparameters that were adjusted were the number of filters, the kernel size, the dropout value, whether or not to use Batch Normalization, and whether or not to use skip connections. Table 4 shows the results of the tuning process associated with the TCN.

This architecture presents an important difference with respect to the others: it is necessary that the receptive field is equal to or greater than the number of lags in the dataset to ensure that the model is effectively using all the lags. Therefore, the "number of lags" hyperparameter was not tuned during the tuning process. Instead, this hyperparameter is adjusted in each experiment based on the number of lags and the size of the kernels. As reported in Bai et al. (2018), architectures do not considerably change their performance as long as their receptive field has been correctly determined. The equation that determines the number of delays is presented in Eq. (2).

$$\#delays = \left\lceil \log_2\left(\frac{\#lags}{kernel\ size} + 1\right) \right\rceil \tag{2}$$

By applying Eq. (2) we ensure that the number of lags is determined so that the receptive field always spans the variable number of lags. It should be noted that although TCNs allow the concatenation of more than one residual block, only one is used in the experiments performed. This decision was made taking into account that the use of more than one residual block is not justified in an experiment where the largest number of lags is 8.

4.5.4. RNN

Finally, for RNN the following hyperparameters were adjusted: number of recurring layers, number of LSTM units per layer and the value of the dropout applied in each layer. As in CNN architectures, the last layer is allowed to be an optional dense layer, so the number of units and the dropout value of this last layer must also be tuned. Table 5 shows the results of the tuning process associated with the RNNs.

4.5.5. Conclusions about the tuning process

As a conclusion, it can be observed in Tables 2 to 5 that there is no clear difference between the hyperparameters of the personal models and those of the impersonal ones. In the same way, there is no clear difference between prototype users. For example, it could be assumed that impersonal models would benefit from the use of deeper or broader architectures, that is, with a greater

Table 6
Ranking of architectures compared by MSE.

	1st	2nd	3rd	4th
RNN	6	5	20	17
CNN	11	9	9	19
TCN	21	21	3	3
MLP	10	13	16	9

(a) min MSE

	1st	2nd	3rd	4th
RNN	25	10	10	3
CNN	11	16	9	12
TCN	6	12	14	16
MLP	6	10	15	17

(b) average MSE

number of units and layers. In Table 2 we see that the personal architecture for the user 32 is composed of 16 layers with 256 units each, while the impersonal architecture for the same user is composed of 2 layers with 32 neurons each. In this case, the difference between the number of hyperparameters is high and counter-intuitive. Therefore, for the task addressed in this article, it is not observed that the greater the amount of data, the greater the complexity of the network. For better known tasks such natural language modelling, the greater the amount of available data to train the models, the greater the complexity of the models needed to learn from that data.

4.6. Prediction results

All DNN architectures were designed and implemented using the Keras[6] deep learning library for Python. Prediction models were trained using a server equipped with an AMD Ryzen 7 2700 CPU and a Titan XP GPU. Given the high number of experiments performed for different combinations of prediction models, users, number of lags, periods, granularity and nature of the model (personal vs impersonal) we present results in a form of a ranking for each user, summarized in a single table.

4.6.1. Comparing architectures with respect to MSE

In this section we present the ranking of the experiments classified by the neural network architecture and compared by the MSE using the users for the individual rankings over the total set of experiments. Table 6a shows the results for the minimum MSE, while and Table 6b shows the results for the average MSE as aggregation function. Each cell of both tables show the number of times a given type of neural network obtained the first, second, third and fourth place in the ranking. The sum of each row or column is 48, the total number of users, since each individual ranking was built for each user.

As can be seen in Table 6a, TCN models obtained the lowest MSE for 21 of the 48 users, while CNN and MLP performed better in 11 and 10 cases, respectively. Finally, RNNs obtained the lowest MSE than the other networks in only 6 cases. It is interesting to notice that TCNs obtained the first or second lowest MSE for 42 of the 48 users. On the other hand, Table 6b shows the results for the average MSE. Very different results can be observed, since in this case the RNN networks obtained the lowest average MSE for 25 of the 48 users, while the CNNs performed better in 11 and 10 cases, respectively. Finally, the MLPs and the CNNs reached a lower average MSE in 6 cases each. The difference between the results of both tables is related to the fact that the total set of experiments is considered. We observed that in the impersonal models there was an exponential growth of the error, phenomena known as *exploding gradient*. This fact particularly affected the TCN in the personal models. This fact will be discussed in greater detail later in Section 5.6.

Next, we built the same ranking with the difference that we filtered by different granularities. There is no significant difference between 1-h and 30-min granularities if the minimum MSE is used as the aggregation function, where the advantage of the TCN continues to hold (Table 7). On the other hand, if the average MSE is used as the aggregation function, the advantage of the RNN over other architectures continues (Table 8).

Next, we analysed the difference according to the nature of the models (personal and impersonal) separately. Table 9 shows the ranking of the personal models using the minimum MSE and the average MSE as aggregation function, while Table 10 shows the ranking of the impersonal models.

In Tables 9 and 10 we can observe a clear difference between the models according to their nature. For personal models, RNNs obtained the best performance. This occurs in 21 or 28 of the cases when using the minimum or average MSE, respectively. On the other hand, for impersonal models, TCNs lead the rankings. The reasons for this difference are addressed in the discussion (Section 5.2).

[6] https://keras.io/

Table 7
Ranking of architectures with 1-h granularity compared by MSE.

	1st	2nd	3rd	4th
RNN	6	6	18	18
CNN	11	8	11	18
TCN	22	20	3	3
MLP	9	14	16	9

(a) min MSE

	1st	2nd	3rd	4th
RNN	19	11	8	10
CNN	11	12	11	14
TCN	14	13	12	9
MLP	4	12	17	15

(b) average MSE

Table 8
Ranking of architectures with 30-mins granularity compared by MSE.

	1st	2nd	3rd	4th
RNN	5	5	19	19
CNN	10	12	11	15
TCN	22	15	6	5
MLP	11	16	12	9

(a) min MSE

	1st	2nd	3rd	4th
RNN	27	9	5	7
CNN	9	20	14	5
TCN	3	7	13	25
MLP	9	12	16	11

(b) average MSE

Table 9
Ranking of architectures for personal models compared by MSE.

	1st	2nd	3rd	4th
RNN	21	15	7	5
CNN	5	15	15	13
TCN	17	7	10	14
MLP	5	11	16	16

(a) min MSE

	1st	2nd	3rd	4th
RNN	28	13	5	2
CNN	9	21	10	8
TCN	7	6	10	25
MLP	4	8	23	13

(b) average MSE

4.6.2. Personal vs. impersonal models

In this section we evaluate personal and impersonal models compared by the MSE using the users for the individual rankings on the total set of experiments, using the minimum and the average respectively as aggregation function. As an example, the first cell of Table 11a indicates that for 5 of the 48 users personal models achieved the best performance. In both tables, a clear predominance of impersonal models over personal models is observed.

A similar ranking is shown in Tables 12, except that we show each iteration separately. We can observe how personal models increase their accuracy compared to impersonal models as more data is available for training. This is better perceived when using the minimum MSE as aggregation function. It is worth noticing that iterations are considered individually and not cumulatively. For example, Iteration 3 comprises only the performance of the models in that iteration. This implies that models that have achieved high performance in one iteration may not have high performance in the others.

Table 10
Ranking of architectures for impersonal models compared by MSE.

	1st	2nd	3rd	4th
RNN	3	6	18	21
CNN	11	8	12	17
TCN	23	20	2	3
MLP	11	14	16	7

(a) min MSE

	1st	2nd	3rd	4th
RNN	5	5	15	23
CNN	13	14	7	14
TCN	24	11	7	6
MLP	6	18	19	5

(b) average MSE

Table 11
Personal vs. impersonal models compared by MSE.

	1st	2nd
Personal	5	43
Impersonal	43	5

(a) min MSE

	1st	2nd
Personal	0	48
Impersonal	48	0

(b) average MSE

Table 12
Personal vs. impersonal models compared by MSE in each iteration.

	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd
personal	8	40	14	34	18	30	19	29	23	25
impersonal	40	8	34	14	30	18	29	19	25	23

(a) min MSE

	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd
personal	1	47	5	43	4	44	10	38	16	32
impersonal	47	1	43	5	44	4	38	10	32	16

(b) average MSE

4.6.3. Training time required

In this section we analyse the ranking of the experiments according to the neural network used, compared by the time required for training, using the users for the individual rankings and the minimum and the average as aggregation function. In Table 13, the ranking is made on the total set of experiments. In Table 14 we only consider impersonal models, while in Table 15 we only considered personal models. As an example, the first cell of Table 13a indicates that for 48 (i.e. all) users a personal model was trained in the shortest time. As expected, there is an important difference in terms of the time required to train personal with respect to impersonal models.

Table 13 shows an interesting fact: even though for all users RNN models were trained in the shortest time, when the average is used as the aggregation function, CNNs were trained in the shortest time on average. This happens because, for few training data, RNNs could be trained quickly (0.13 ± 0.035 minutes), compared to the CNNs (0.285 ± 0.08) minutes (Table 15). However, when the amount of data of training grows, RNNs increase the time required to be trained (4.895 ± 5.24 minutes), while the CNNs even required less time (0.21 ± 0.08 minutes) thanks to their high parallelization capacity (Table 14). In all the tables it can be seen that the TCN is in the last position in terms of the time required to be trained (2.3 ± 1.04 minutes globaly, 1.83 ± 0.69 minutes for personal models and $2,75 \pm 1.13$ minutes for impersonal models).

Table 13
Time required to train the models using MSE.

	1st	2nd	3rd	4th
RNN	48	0	0	0
CNN	0	48	0	0
TCN	0	0	0	48
MLP	0	0	48	0

(a) min MSE

	1st	2nd	3rd	4th
RNN	0	0	36	12
CNN	48	0	0	0
TCN	0	0	12	36
MLP	0	48	0	0

(b) average MSE

Table 14
Time required to train the impersonal models using MSE.

	1st	2nd	3rd	4th
RNN	0	0	30	18
CNN	48	0	0	0
TCN	0	0	18	30
MLP	0	48	0	0

(a) min MSE

	1st	2nd	3rd	4th
RNN	0	0	36	12
CNN	48	0	0	0
TCN	0	0	12	36
MLP	0	48	0	0

(b) average MSE

Table 15
Time required to train the personal models using MSE.

	1st	2nd	3rd	4th
RNN	48	0	0	0
CNN	0	12	36	0
TCN	0	0	0	48
MLP	0	36	12	0

(a) min MSE

	1st	2nd	3rd	4th
RNN	48	0	0	0
CNN	0	17	31	0
TCN	0	0	0	48
MLP	0	31	17	0

(b) average MSE

4.6.4. Results for prototype users

In this section, we present in more detail the results obtained for prototype users. Remember that user 34 represents the group of users with the lowest energy expenditure and the fewest number of available buckets, while user 32 represents the group of users with the highest energy expenditure and the largest number of available buckets. We show the configuration of the experiments that obtained the lowest MSE for each prototype user, as well as the MSE and the time required for training each of the iterations. In addition, we present the corresponding predictions graphs for both users. In the case of user 34, the model that obtained the lowest MSE was a TCN architecture for an impersonal model, with a granularity of 1 h, using 8 lags, and a period of 4. The average MSE obtained was 0.1328, while the average training time was 1.9 min. Table 16 shows the MSE values and training time for each iteration.

Fig. 6 shows the predictions made by the model that obtained the lowest performance for user 34. Different iterations are indicated by dashed vertical lines. The red horizontal dotted line indicates the threshold of 1.5 MET that indicates sedentary behaviour. We can observe that there were few available buckets and few records with non-sedentary behaviour. Consequently, the model is unable to correctly identify those moments in which the user had a high level of MET. In these cases, where most buckets

Table 16
MSE and time required to train the bet performing model for user 34.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
MSE	0.195	0.044	0.140	0.107	0.178
time	0.713	1.393	2.029	2.614	2.844

Table 17
MSE and time required to train the bet performing model for user 34.

	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
MSE	0.110	0.221	0.186	0.569	0.414
time	0.938	0.973	2.934	2.391	3.892

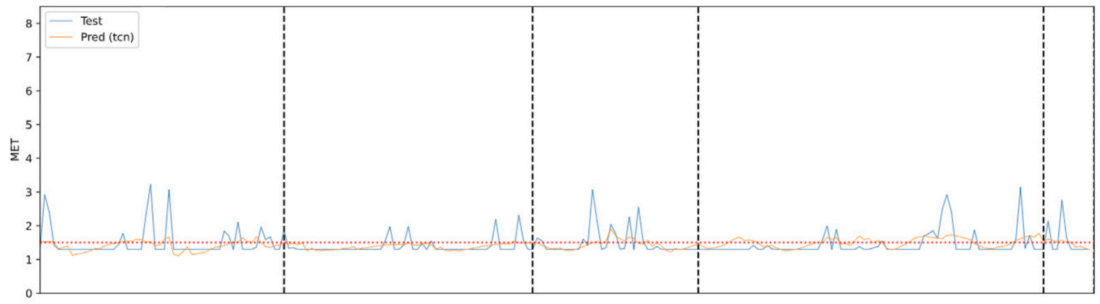

Fig. 6. Predictions for user 34.

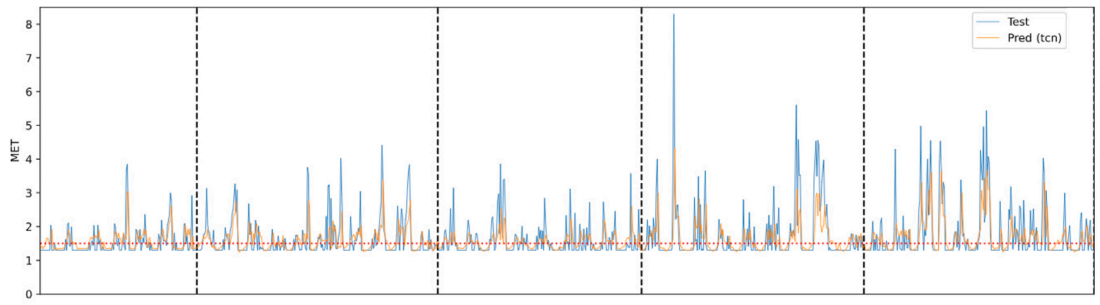

Fig. 7. Predictions for user 32.

correspond to sedentary behaviour, we can see the importance of using MET as an error function, since buckets with non-sedentary behaviour should be penalized.

In the case of user 32, the model that obtained the lowest MSE was the impersonal model, with a granularity of 1 h, using 8 lags, and a period of 1. The architecture corresponded to a TCN. The average MSE obtained was 0.3 while the average training time was approximately 2.2 min. Table 17 shows the values of MSE and training time for all iterations.

Fig. 7 shows the predictions made by the model that obtained the lowest performance for user 32. Compared to Fig. 6, there are more available buckets of non-sedentary behaviour, so the model was able to make better predictions when a peak of high energy expenditure was observed.

5. Discussion

In this section we will discuss in detail the results presented in Section 4. In all cases, since samples to be compared are considerably large, Student's test was run to analyse the statistical differences of MSE obtained by the different experiments. To test homoscedasticity, we used the Levene's test. For those, experiments in which samples did not result to have equal variances (heteroscedasticity), we used the Welch's version of the test. We organized this section in different subsections to analyse the impact of the selected prototype users (Section 5.1), the difference between personal and impersonal models (Section 5.2), and the performance of different deep learning architectures (Section 5.3). We then analyse the impact of selecting different granularity (Section 5.4) and number of laps (Section 5.5) for processing the available records. Finally, we discuss the problem of the *exploding gradient* that we faced for some models (Section 5.6) and we present some advantages and disadvantages of the proposed approach with respect to related work (Section 5.7).

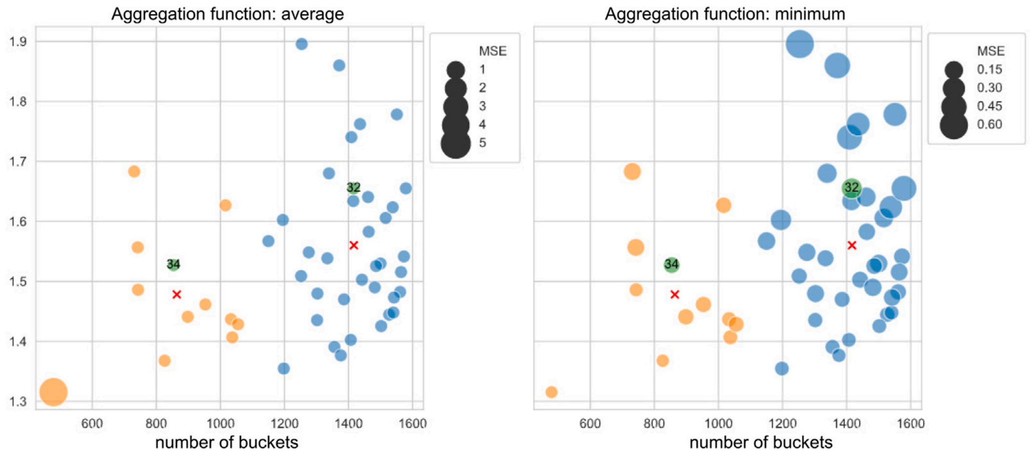

Fig. 8. MSE for each user.

5.1. *Impact of prototype users*

In Section 4.2 we discussed the selection of two prototype users from the dataset that represent groups of user with different energy expenditure. In this section we analyse the impact of this strategy with respect to analysing every individual user in the tuning process. Fig. 8 shows the MSE obtained for each individual user using the minimum and average MSE as the aggregation function. For example, in the plot on the left of Fig. 8, the radius of the circle corresponding to user 32 represents the average MSE obtained for the models built for that user. In the plot on the right of Fig. 8, the radius of the circle corresponding to user 32 represents the minimum MSE obtained for that user.

We found no significant difference in the MSE obtained for different users when we use the average as an aggregation function. This fact mainly occurs because there was one user in particular whose average MSE was very high. In the right plot of Fig. 8 we can observe a clear pattern that differentiates both groups of users. On the one hand, there is not significant difference between the radii of the circles for users in the group represented by user 34 (users with low average and standard deviation of METs and few buckets available). On the other hand, in the group represented by user 32 (high average and standard deviation of MET and greater number of available buckets) we observe that the lower the average MET, the lower the resulting MSE. This may be due to the fact that it was easier for the model to predict sedentary behaviour as there are not many cases of non-sedentary behaviour. We can also observe that it was difficult for the models to make accurate predictions for those users with higher average METs, compared to users with low average METs.

In summary, no pattern emerges from the results of the experiments to show that the choice to perform the tuning process from certain prototype users has had negative consequences for users farther away from the centroids of each group.

5.2. *Personal and impersonal models*

In this section, we discuss the differences between personal and impersonal models in terms of their overall performance and over different iterations. The main difference between the two models is that in the case of impersonal models we have information about all users except the one we want to make predictions about. In the case of personal models, the opposite is true: only information about the user for whom predictions are to be made is available. In practice, this leads to a very important difference: impersonal models have many more case studies than personal models. At the same time, though, only personal models have data that represent the idiosyncrasies of the target user. How much benefit this type of model can derive from this fact will depend on how similar the sedentary behaviour is among the different users, since, if most of the users have similar behaviours, the personal model should not be able to perform better than the impersonal model. Then, it is necessary to analyse how these practical consequences impacted the experiments.

As shown in Fig. 8, the average MET for the entire set of users varies widely, from 1.3 to 1.9. Fig. 9 shows the predictions for user 46. The model that achieved the best performance for user 46 is an impersonal model, using an MLP architecture, with a granularity of 60, 1 lag and a period of 1. In Fig. 9, we see that the user has many peaks of energy expenditure. The model is able to predict such peaks but not precisely their magnitude. Fig. 10 shows in detail the first iteration, where it is more clearly seen that the sedentary and non-sedentary moments are correctly predicted. Since the model user for this predictions is impersonal, it was trained from the data of all other users, there being none whose energy expenditure is so high (since user 46 has the highest average MET).

It is logical to analyse, then, whether personal models for user 46 are able to predict more accurately the energy expenditure peaks. The personal model with the highest performance was ranked in 11th position. This model uses an RNN architecture, with a granularity of 1 h, 8 lags and a period of 1. Figs. 11 and 12 show the predictions for this model analogously as shown above with

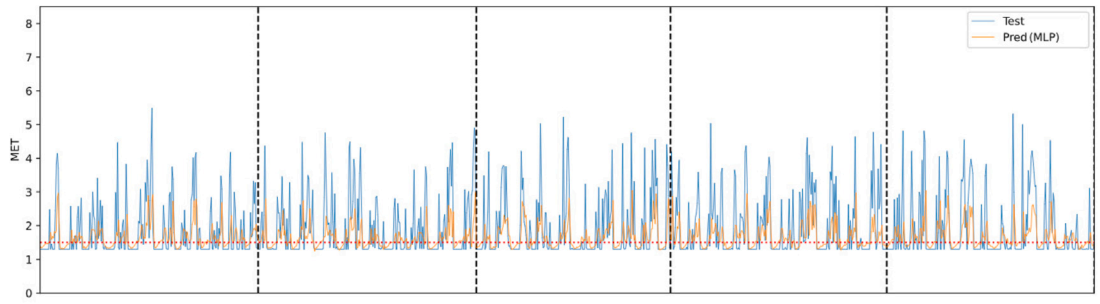

Fig. 9. Predictions for user 46.

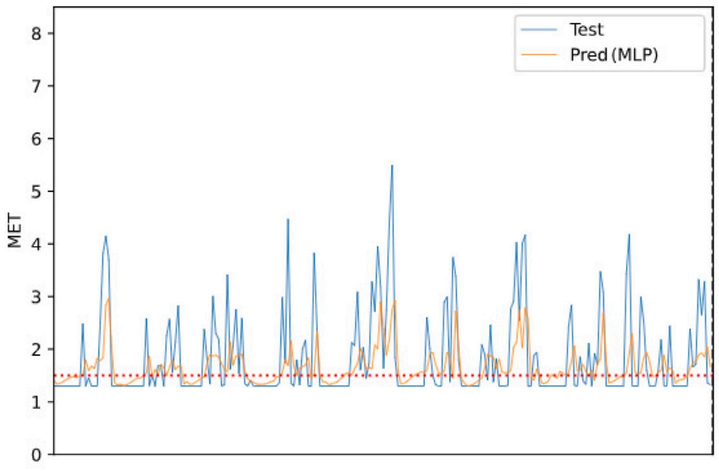

Fig. 10. Predictions for user 46, first iteration.

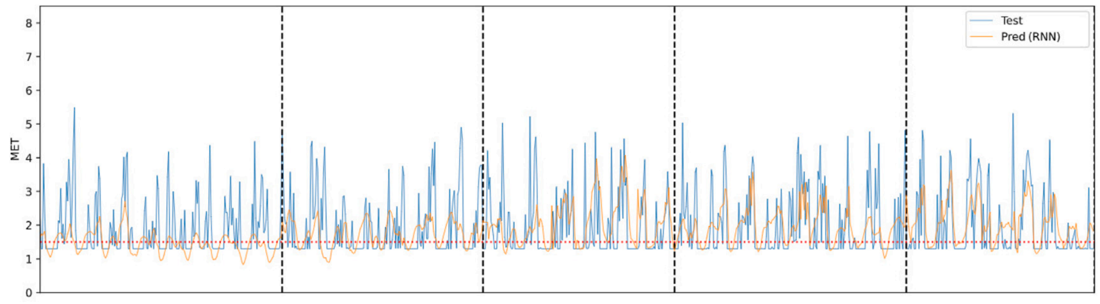

Fig. 11. Predictions for user 46 for the best personal model.

the impersonal model. In Fig. 11 it can be seen how for the first iterations the model predictions are highly inaccurate, due to the small number of training cases available so far. However, from iteration 3 onwards, the model starts to make better predictions. We can also observe, in the right image of Fig. 12, that the model is able to predict more accurately the energy expenditure peaks than in the case of the impersonal model, although the model did not get it right in certain cases where it predicts a high MET when the user has sedentary behaviour.

The MSE values for the predictions of the best impersonal model for each iteration were 0.514, 0.727, 0.776, 0.774 and 0.799, respectively. The MSE values obtained for the best personal model in each iteration were: 0.733, 0.892, 0.874, 0.714, 0.606, respectively. This confirms the information plotted in Figs. 11 and 12. The MSE values for the first iterations is worse in the case of personal models. Then, in the last two iterations, the personal model performs better than the impersonal models.

The fact that the performance of the personal models improves as the amount of data they hold about the user increases (in this case, over different iterations) is observed at a general level, and may be due to the fact that in later iterations the personal models already have sufficient data from which the idiosyncrasies of each user can be modelled. Table 12, showed that as the iterations

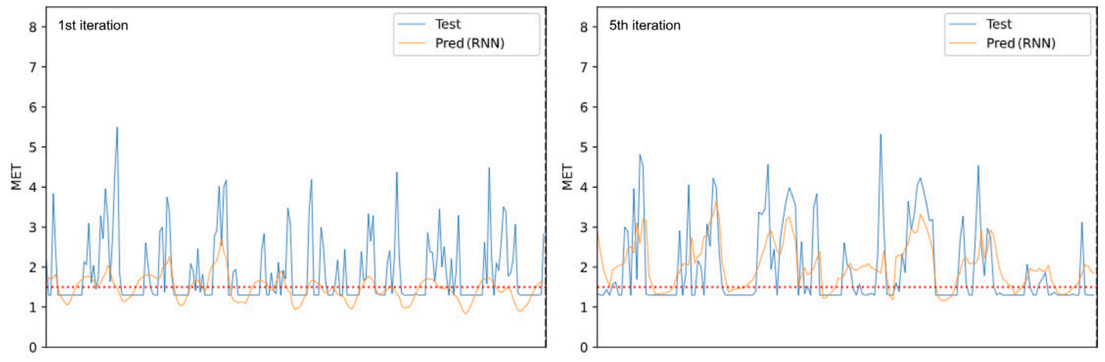

Fig. 12. Predictions in the first and last iterations of the best personal model for user 46.

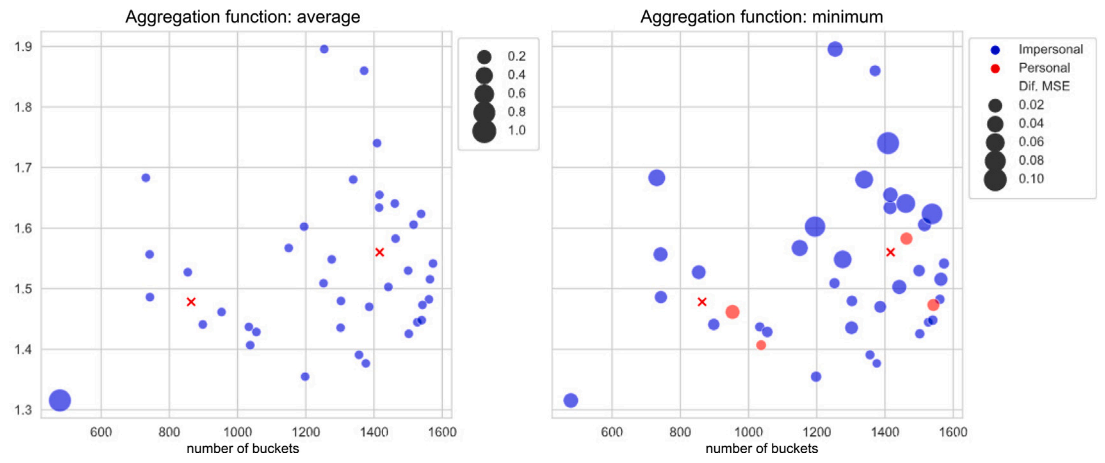

Fig. 13. Difference between MSE values for personal and impersonal models, for all users.

progress, the performance of the personal models increase compared to the impersonal models (based on how many users have achieved the best performance). This is an important fact to observe, since at the overall level the impersonal models achieved higher performance than the personal models for 43 of the 48 users, reducing to 25 of the 48 users in the last iteration. The fact that at the general level the impersonal models have prevailed over the personal models might be due to the low performance that the latter achieved in the first iterations, due to the low amount of training data. It is important to clarify in this last observation that the analysis performed to compare the personal and impersonal models was carried out on the basis of the different iterations taken individually. As an example, although for the last iteration –for which the models have the largest amount of data available for training – the personal models achieved the best performance for almost the same number of users as the impersonal models, this does not mean that this is the case at the global level. It is often the case that a personal model outperforms the impersonal models in a single iteration, not in several consecutive iterations. Moreover, if we look at the ranking of the models, within the first tens, the vast majority are impersonal models, even if the first position of the ranking is obtained by a personal model.

Fig. 13 shows, for each user, the model that achieved the best performance based on the nature of the model (personal or impersonal). The radius of the circles represents the difference between the MSE of the best personal model and the best impersonal model – based on the mean in the left plot and based on the minimum in the right plot –. In the left plot of Fig. 13, we observed that, for all users, the best model was impersonal. Moreover, in the plot on the right of Fig. 13 we see that for all but 4 users, impersonal models performed better than personal models. Due to the exploding gradient problem that many personal models suffered from (see Section 5.6) the differences between the two when analysing the first image is huge. On average, the difference in the MSE values when using the average as the aggregation function was 20,854,079. The only neural network architecture whose personal models were not affected by the exploding gradient was the RNN. The above average drops to 0.04 if the models in which an RNN was used are filtered out.

If we analyse the image on the right of Fig. 13, the difference between both types of models is between 0.0012 and 0.112 and it is similar in most cases. However, this difference increases for users with higher average MET and more buckets. Specifically, this phenomenon occurs for users belonging to the upper side of the cluster associated to user 32. This may be due to the fact that impersonal models for these cases have enough training cases with users with high energy expenditure to learn from, while personal models are the ones that fail the most in these cases, especially in the first iterations.

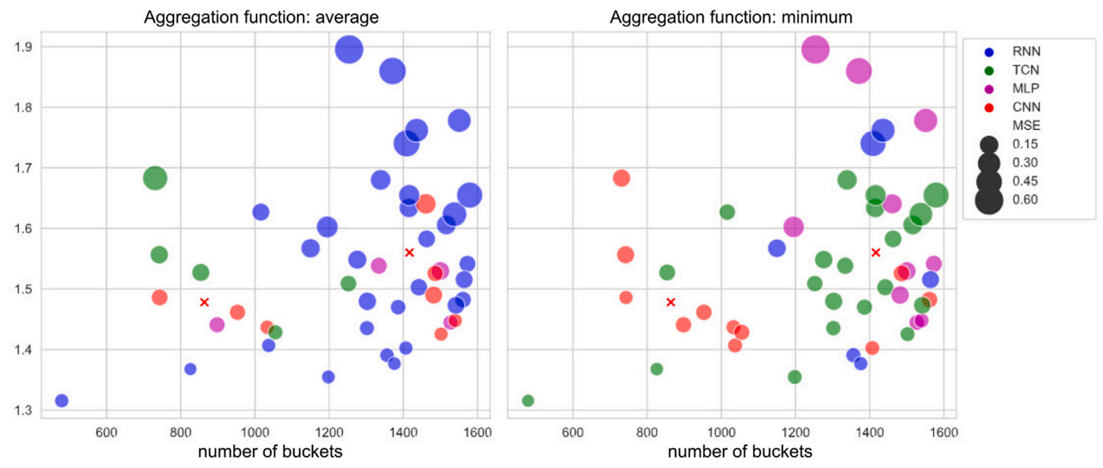

Fig. 14. Best performing neural network architectures for each user.

5.3. Performance of different architectures

In this section, we analyse the performance of the different types of neural networks based on how they performed in the groups created from the prototype users. To begin with, we show in Fig. 14, for each user, the neural network that performed best based on the average (left plot) and minimum (right plot) MSE of each user's set of experiments.

In the left plot of Fig. 14 RNNs obtained the lowest average MSE for the majority of users. Despite this, such a majority does not hold for the cluster of user 34 (low average and standard deviation of MET and low number of available buckets). The advantage of RNNs can be explained from what will be discussed in Section 5.6. RNNs turned out to be the most stable architecture for personal models.

We can observed a pattern that differentiates the two groups of users is the plot on the right of Section 5.6. On the one hand, in the group corresponding to user 34 (low average and standard deviation of MET and low amount of available buckets) CNNs obtained the best performance for 8 of the 12 users. On the other hand, in the group corresponding to user 32 (high average and standard deviation of METs and greater number of available buckets), TCNs were the best neural network for most users.

Fig. 14 was created with the total set of experiments although, as we previously noticed, RNNs obtained the best performance in most users because it was the most stable neural network. Then, it is interesting to analyse which neural networks performed best in each case for the personal and impersonal models separately. Fig. 15 shows the best performing neural networks for each user taking into account only the personal models, using the average (left plot) and the minimum (right plot) as aggregation functions. For users belonging to the group with the highest energy expenditure and the largest number of available buckets, RNNs performed better in most cases, even better than the analysis shown in Fig. 13. This indicates their good predictive ability for more heterogeneous datasets with respect to the higher presence of buckets corresponding to non-sedentary behaviour, at the same time as a small amount of data. Moreover, they did not suffer from the exploding gradient problem. On the other hand, for the group represented by user 34, in half of the cases TCNs obtained the best average MSE and in the other half RNNs, while TCN obtained the lowest MSE for most of the users when considering the minimum MSE as the aggregation function.

Fig. 16 shows the best performing neural networks for each user taking into account only the impersonal models, using the average (left plot) and the minimum (right plot) as aggregation functions. We can observe that both plots in Fig. 16 are very similar. In this case, the models that obtained the lowest average MSE are also those that obtained the lowest minimum MSE. We can observe in both plots that TCNs performed better for impersonal models, although CNNs obtained the best performance for users in the cluster of user 34.

5.4. Granularity

One of the categories defined at the time of designing the experiments is granularity. Granularity defines the time interval from which the features representing each bucket are computed. The values defined for the experimentation were 30 min and one hour. Therefore, the datasets built with a granularity of 30 min have approximately twice as many training cases as those with a granularity of 1 h. Lower granularity allows for better temporal accuracy in predicting future sedentary behaviour, while the features used to train the model have a smaller amount of data to perform the computations. The Student's test was performed to compare the MSE of these two types of models (30-min and 1-h granularity) and no statistical difference was found. However, as can be seen in Fig. 17, the models with 1-h granularity have a lower MSE for most users if the mean is used as the aggregation function. If the same test is performed comparing the time required for the models to be trained, a statistical difference is found (p-value = 1.30e−35). Those feature values that vary between experiments that increase the number of records in the training datasets translate into a considerable increase in the time required to train the models. Therefore, it is interesting to note that, depending on which feature is the most important to minimize, one granularity or the other can be considered.

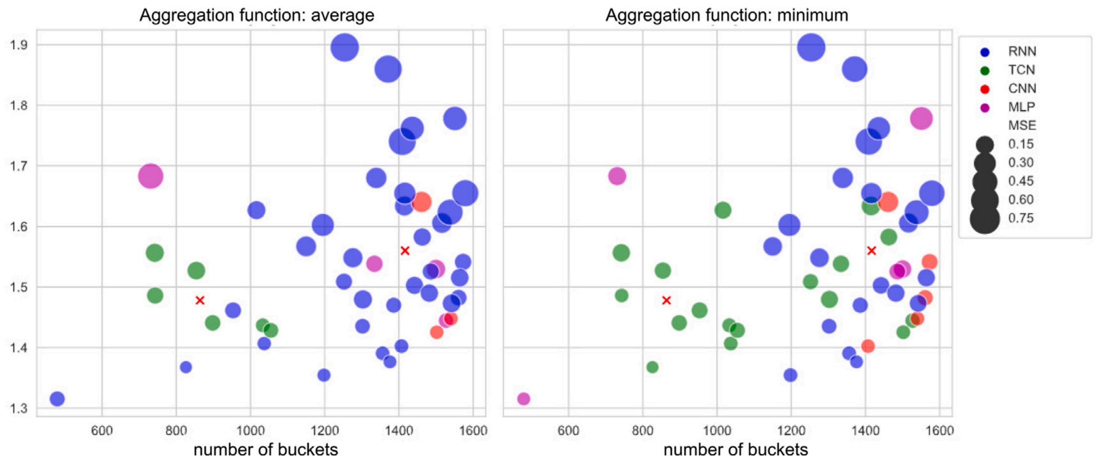

Fig. 15. Best performing neural network architectures for personal models.

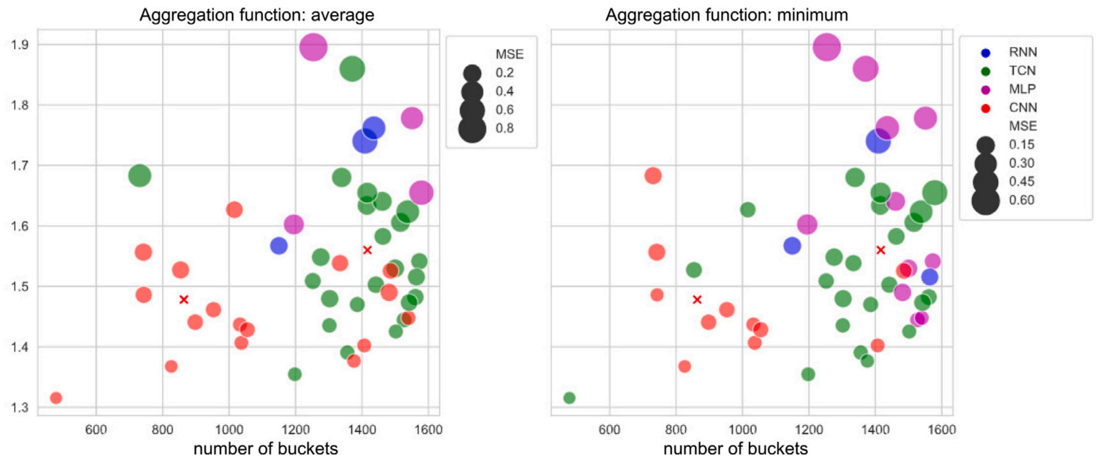

Fig. 16. Best performing neural network architectures for impersonal models.

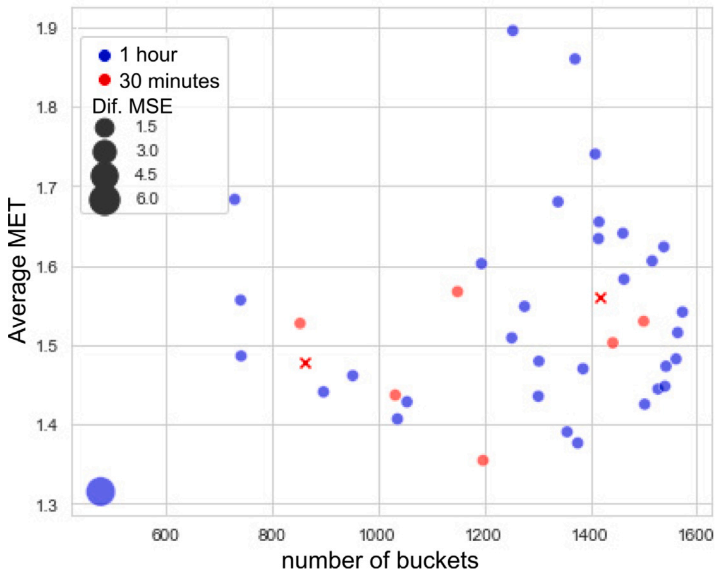

Fig. 17. Better models with respect to granularity.

Table 18
MSE for deciles in personal models.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RNN	0.030	0.101	0.138	0.181	0.225	0.268	0.319	0.374	0.494	0.691	1.922
CNN	0.027	0.120	0.176	0.229	0.286	0.371	0.504	0.925	9.948	156.851	1.11×10^{10}
TCN	0.024	0.137	0.199	0.255	0.346	0.499	0.796	3.255	38.039	784.318	2.39×10^{10}
MLP	0.041	0.145	0.201	0.254	0.319	0.431	0.707	2.287	19.646	358.225	2.16×10^{10}

Table 19
MSE for deciles in impersonal models.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RNN	0.035	0.091	0.128	0.157	0.191	0.233	0.275	0.322	0.420	0.609	1.403
CNN	0.013	0.084	0.118	0.149	0.185	0.228	0.275	0.319	0.419	0.638	705.582
TCN	0.012	0.086	0.118	0.151	0.183	0.225	0.269	0.315	0.406	0.640	149.907
MLP	0.028	0.089	0.123	0.153	0.186	0.228	0.273	0.318	0.416	0.642	339.085

5.5. Number of lags

To carry out the experiments, different numbers of lags were tested (namely 1, 2, 4 and 8). It is possible to deduce that the more information from the past feeds the training model, the more information it will have to be trained to perform better. This may not be the case for two reasons. The first is that the MET that a user will have at time $t+1$ may not be related to the features that we have of the same user at time $t-n$, since n is an arbitrary integer. Having few training cases may generate noise in the model. The second reason is that, since in many cases we do not have data on all the time that the experiment lasted, the more lags we have, the higher the probability that a training case will be discarded. For example, if the number of lags is n and we have data for user x from time t to time $t-n+1$ but not from time $t-n$, the training case is discarded.

These reasons explain the results obtained, in which it was found that models with fewer lags performed better. A statistical difference ($p-value < 0.05$) was found only when comparing pairs of 1, 2 and 4 number of lags with respect to 8 lags. Analysing the sets of experiments more closely, it is possible to observe that the main reason that explains this fact is that experiments with a higher number of lags suffered to a greater extent from the exploding gradient problem. This is verified by the fact that when the test is carried out without taking into account the last decile of each set of experiments, no statistical difference is found.

Regarding the time required to train the models according to the number of lags used, we did not find a statistical difference for any of the of lags used. This fact indicates that the number of columns used to train the models does not increase the time required to train them. This result is possibly due to the high parallelization capacity of the GPU used to train the models.

5.6. Exploding gradient

As we mentioned briefly in other sections of this article, many of the models suffered from the problem known as *exploding gradient*. This problem occurs when the values of the neural network weights, typically known as the neural network parameters, grow in such a way that their values are greater than one. In this way, it happens that when forward propagation is carried out, the result of the intermediate layers grows exponentially, and therefore also the output of the model. This problem results in the model not being able to learn from the training set. All networks architectures, except the RNNs, suffered from the exploding gradient problem to a greater or lesser extent. This was because the LSTM units that were used in the RNNs are specially designed to avoid this problem. In the case of the other neural networks used, no measures were taken to avoid this problem because it was not detected at the time of the tuning process. If the exploding gradient had been detected during the tuning process, a simple technique known as gradient clipping could have been used. Gradient clipping consists in ensuring that the value of the weights of the model is not greater than a certain value, for example, 1.

The exploding gradient problem occurred, except for a few cases, only in personal models. This can be explained by the small number of available training cases. In order to demonstrate the exploding gradient problem, Tables 18 and 19 show the deciles corresponding to the set of experiments for personal and impersonal models, respectively.

As can be seen in Table 18, deciles 7, 8, 9 and 10 contain between 30% and 40% of the experiments for personal models that suffered from the problem discussed in this section for all neural networks except for RNNs. Specifically, 35.7% of this set of experiments have an MSE greater than 1. It is interesting to note that this percentage is lower if each of the iterations of the evaluation process are taken individually (26%, 19%, 12%, 9% and 5%, respectively), which indicates that the exploding gradient problem occurred throughout all the iterations and not in one in particular, although it is true that it is more important in the first iterations. This shows that this problem is caused mainly by the small amount of data available for the experiment in the first iterations.

In the case of Table 19, we only find experiments that suffered from the exploding gradient problem in the last decile and in a radically smaller magnitude. We see that only 2% of this set of experiments have an MSE greater than 1. In this case, we understand that the problem did not occur because of the availability of enough data during the first iterations.

5.7. Comparison with related work

As we discuss in Section 2.1 to the best of our knowledge, the use of MET to predict Future Sedentary Behaviour is novel and it is not straightforward to compare the results obtained by related work with respect to our work. The research that is most closely related to ours is that performed by He and Agu (2016c), He and Agu (2016b), He and Agu (2016a), He and Agu (2017). These authors built several interesting models in their research, but they measured sedentary behaviour in terms of the average of sedentary records per time period, that is, the percentage of all activities in a given time period labelled as *stationary*. Instead, we computed a sedentary level by using the MET levels of each physical activity encountered in each time period according the Compendium of Physical Activities, which we believe that gives a better approximation to the real energy expenditure of the subject.

Another important difference of our work with respect to the work of He and Agu is that they only considered personal models, while we compare personal and impersonal models. The results described in this work revealed that impersonal models performed better than personal models for most subjects. This result is particularly relevant for facing the cold start problem, that is when subjects start using any device or software for detecting sedentary behaviour and no enough information about her/his activities is available for making predictions. We also observed that personal models increase their accuracy compared to impersonal models in the last iterations of the learning algorithm, as more data is available for training. Then, an impersonal model can be very useful until enough information about the subject is collected to make accurate predictions.

In He and Agu (2016a, 2016c), authors only used the activity log available in the Student's life dataset, which consists of a timestamp and an activity type (*stationary, walking, running, and unknown*) recognized only from the smartphone's accelerometer. The methodology followed in this approach segments a user's timeline into buckets with different time granularity to find recurrent patterns of sedentary behaviour. Within each time bucket, the authors computed the percentage of all activities labelled as *stationary* (as an indicator of sedentary behaviour) and considered the resulting series of numbers as a signal whose cycle depends on how often the user exhibit a sedentary behaviour. Then in He and Agu (2016a) they applied a discrete Fourier transform to convert that signal from the time domain to the frequency domain and in He and Agu (2016c) they explored an autoregressive model. In He and Agu (2016b) more features were considered (namely, 22 user context variables, such as location, time and app usage) and logistic regression was used to predict whether the user would be "very sedentary" in the next hour or not. In this work, the authors considered only 1-h buckets and used the same approach than in He and Agu (2016a) to compute the sedentary level of each bucket as the percentage of all activities in the bucket labelled as *stationary*. Then, they discretized the resulting sedentary level into three classes, namely *very sedentary, sedentary, and less sedentary* under the assumption that coarse ranges of sedentary levels will likely be more useful than exact sedentary level values.

In our work, we used 31 features computed from the GPS, Wi-Fi and audio sensors, physical activity, time, among others. Many of these features are the same than those considered in He and Agu (2016b), for example *isLocked*, *isInDark*, *isInConversation* and *isCharging*, but we also added other related features such as *lockedRatio*, *darkRatio*, *conversationRatio* and *chargingRatio*. Some features used in He and Agu (2016b) were not considered in our research since they were very specific for the population of subjects in the dataset (students), for example those related to the time remaining for a scheduled exam, whether the subject is attending to a class or the number of seconds before the next homework is due. Some features used in our work where defined following the idea presented in He and Agu (2017), in which authors claimed that rhythmical sedentary patterns do exist. These features are *secondSine*, *secondCosine*, *weekDaySine*, and *weekDayCosine* and intend to model the cyclical nature of time-related variables. We also defined new features such as the location variance and mean, the speed variance and men and the total distance travelled.

Finally, one of the most important difference with respect to related work is that we face sedentary behaviour prediction as a regression problem instead of a classification problem. This is an important advantage of our approach since we do not need to impose any self reported cutting line between what is considered sedentary and not sedentary. By using the MET value as a measure of energy expenditure, we can consider sedentary behaviour as any behaviour characterized by an energy expenditure lower or equal than 1.5 METs, as defined by the Sedentary Behaviour Research Network, or we can establish different cutting lines for different applications or populations without affecting the learning and predicting algorithms.

6. Conclusions

In this article, we analysed the capability of different deep learning architectures to learn and predict future sedentary behaviour of a subject at two different time intervals: the following half hour, and the following hour. We compared training personal and impersonal models to this aim, that is, learning the model only with the historical information available for the target subject, and using all the information available for all subjects. We performed experiments to evaluate different aspects involved. We summarize next some lessons learned from the experiments performed to predict future sedentary behaviour from lifelogging data using deep learning:

- We did not observed evidence to confirm that, as happens in other domains such as Natural Language Processing, the greater the amount of available data, the greater the complexity of the network. For example, the best DNN architecture for a personal model for one subject is composed with of 16 layers with 256 neurons each while the best DNN architecture for the impersonal model for the same subject was composed by only 2 layers with 32 neurons each, despite the fact that we have data from all other subjects available for training.
- Temporal Convolutional Networks (TCN) obtained better performance than other architectures for most users with impersonal models. For personal models, where fewer data is available, Recurrent Neural Network (RNN) obtained better performance than TCN. Furthermore, RNN resulted to be the most stable architecture when few available data was available.

- For subjects with low energy expenditure and low number of available buckets a TCN architecture using eight lags and a period of four obtained the best performance. For these subjects, since there were few records with information of non-sedentary behaviour, the model is unable to correctly identify those moments in which the user had a high level of MET. On the other hand, for users with high energy expenditure and high number of available buckets, a TCN architecture using eight lags and a period of 1, that is, considering eight consecutive previous buckets for prediction, obtained the best performance.
- We did not find significant difference between using a granularity of 1-h and 30-min for prediction. It is worth noticing that by using a 30-min granularity we approximately duplicate the training cases with respect to 1-h granularity. Lower granularity allows for better temporal accuracy in predicting FSB, while the features used to train the model have smaller amount of data to perform the computations.
- Impersonal models performed better than personal models for most users. This is mainly due to the fact that more data is available for training the models. This observation is supported by the fact that if we analyse different iterations separately, we observe that personal models increase their accuracy compared to impersonal models in the last iterations, as more data is available for training.
- If we inspect users with particular patterns of behaviours, such as subject 46 in the dataset, a personal model adapted better to his/her behaviour. This conclusion is quite straightforward since the model was trained only with the information of that subject and it was able to capture the particular patterns of energy expenditure that he/she revealed, which are particularly different from other subjects in the dataset.
- From the last two insights, we conclude that a real application installed in the subjects wearable or mobile devices might face the cold start problem by using an impersonal model, using a TCN architecture until enough information about the user is collected. Then a RNN model using only his/her information should be trained to replace the impersonal model.

7. Limitations and future work

One of the limitations found lies in the dataset used for model training. The StudentLife Dataset was the only one compatible with the requirements of our problem. Although this dataset could be used for our purposes, it presented certain limitations that resulted in different difficulties when processing and using it to train Deep Learning models for PCSF.

One of the difficulties related to the dataset was that a large number of buckets had to be discarded due to the lack of available information. However, an analysis was carried out in order to maximize the amount of information available: where possible, we used the lack of data as information. For example, for the feature calculated from the GPS sensor *totalDistance* in the buckets for which no data is available was assigned a 0, thus assuming that the user remained in the same geographic location.

The granularity of physical activity reported for users in the dataset is, perhaps, too low for the wide variety of physical activities possible in terms of their MET level. For example, activities classified as “walking” were given a value of 5 METs, whereas there are more than 50 activities within the spectrum of “walking” considered in the Compendium. Each of these activities has a different MET value, ranging from 2 to 12 METs. Therefore, better results could have been obtained if detailed activities with greater granularity were available.

Another limitation was that raw accelerometer data was not available, which could have opened other research opportunities when designing Deep Learning models. Counting with raw accelerometer data would allowed for designing more extensive and complex architectures, which could capture more information about the user’s physical activity, thus being able to enrich the low granularity available.

This article explored the Prediction of Future Sedentary behaviour in the following time interval given the current time. However, prediction for other time intervals remains unexplored. Even if the model is sufficiently accurate, multiple consecutive time intervals could be predicted, for example, 2 h, the whole afternoon or a whole day, which would enrich the value of the suggestions to prevent sedentary behaviour that could be made to the user.

Many issues related to the neural networks used were not explored in order to limit the universe of experiments. Each of the neural networks, has hyperparameters and architectural variations that have not been explored. In addition, novel techniques used in neural networks could be tested for the task addressed in this article, such as attention (Vaswani, et al., 2017).

CRediT authorship contribution statement

Martín Santillán Cooper: Methodology, Software, Investigation, Data curation, Writing – original draft. **Marcelo G. Armentano:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision.

Data availability

We have shared the link to our code at the Attach Files step. Data are publicly available at https://studentlife.cs.dartmouth.edu/

Sedentery behavior prediction (Original data) (Github)

Acknowledgement

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Agnihotri, A., & Batra, N. (2020). Exploring Bayesian optimization. *Distill*, http://dx.doi.org/10.23915/distill.00026, https://distill.pub/2020/bayesian-optimization.

Ainsworth, B. E., Haskell, W. L., Herrmann, S. D., Meckes, N., Bassett, D. R., Tudor-Locke, C., et al. (2011). 2011 Compendium of physical activities: a second update of codes and MET values. *Medicine and Science in Sports and Exercise*, *43*(8), 1575–1581.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211, Theories of Cognitive Self-Regulation.

Atkin, A. J., Gorely, T., Clemes, S. A., Yates, T., Edwardson, C., Brage, S., et al. (2012). Methods of measurement in epidemiology: Sedentary behaviour. *International Journal of Epidemiology*, *41*(5), 1460–1471.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271.

Benatti, F. B., & Ried-Larsen, M. (2015). The effects of breaking up prolonged sitting time: A review of experimental studies. *Medicine and Science in Sports and Exercise*, *47*(10), 2053–2061.

Carter, S., Hartman, Y., Holder, S., Thijssen, D. H., & Hopkins, N. D. (2017). Sedentary behavior and cardiovascular disease risk: Mediating mechanisms. *Exercise and Sport Sciences Reviews*, *45*(2), 80–86.

Chen, F., Wang, R., Zhou, X., & Campbell, A. T. (2014). My smartphone knows i am hungry. In *Proceedings of the 2014 workshop on physical analytics* (pp. 9–14).

Dutta, M. J., Kaur-Gill, S., Tan, N., & Lam, C. (2018). *mHealth, health, and mobility: a culture-centered interrogation* (pp. 91–107). Dordrecht: Springer Netherlands,

Fahim, M., Baker, T., Khattak, A. M., & Alfandi, O. (2017). Alert me: Enhancing active lifestyle via observing sedentary behavior using mobile sensing systems. In *2017 IEEE 19th international conference on e-health networking, applications and services* (pp. 1–4).

Falck, R. S., Davis, J. C., & Liu-Ambrose, T. (2017). What is the association between sedentary behaviour and cognitive function? A systematic review. *British Journal of Sports Medicine*, *51*(10), 800–811.

Felez-Nobrega, M., Hillman, C. H., Dowd, K. P., Cirera, E., & Puig-Ribera, A. (2018). $ActivPAL^{TM}$ Determined sedentary behaviour, physical activity and academic achievement in college students. *Journal of Sports Sciences*, *36*(20), 2311–2316.

Gong, J., Huang, Y., Chow, P. I., Fua, K., Gerber, M. S., Teachman, B. A., et al. (2019). Understanding behavioral dynamics of social anxiety among college students through smartphone sensors. *Information Fusion*, *49*, 57–68.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Grundgeiger, T., Pichen, J., Häfner, J., Wallmann-Sperlich, B., Löffler, D., & Huber, S. (2017). Combating sedentary behavior. In *Proceedings of the 2017 CHI conference extended abstracts on human factors in computing systems* (pp. 1632—1639).

Harari, G. M., Gosling, S. D., Wang, R., Chen, F., Chen, Z., & Campbell, A. T. (2017). Patterns of behavior change in students over an academic term: A preliminary study of activity and sociability behaviors using smartphone sensing methods. *Computers and Human Behaviour*, *67*, 129–138.

He, Q., & Agu, E. (2014). On11: an activity recommendation application to mitigate sedentary lifestyle. In *Proceedings of the 2014 workshop on physical analytics* (pp. 3–8).

He, Q., & Agu, E. O. (2016a). A frequency domain algorithm to identify recurrent sedentary behaviors from activity time-series data. In *2016 IEEE-EMBS international conference on biomedical and health informatics* (pp. 45–48).

He, Q., & Agu, E. O. (2016b). Smartphone usage contexts and sensable patterns as predictors of future sedentary behaviors. In *2016 IEEE healthcare innovation point-of-care technologies conference* (pp. 54–57).

He, Q., & Agu, E. O. (2016c). Towards sedentary lifestyle prevention: An autoregressive model for predicting sedentary behaviors. In *2016 10th International symposium on medical information and communication technology* (pp. 1–5).

He, Q., & Agu, E. O. (2017). A rhythm analysis-based model to predict sedentary behaviors. In *2017 IEEE/ACM international conference on connected health: applications, systems and engineering technologies* (pp. 383–391).

Kanjo, E., Younis, E. M., & Ang, C. S. (2019). Deep learning analysis of mobile physiological, environmental and location sensor data for emotion detection. *Information Fusion*, *49*, 46–56.

Koster, A., Caserotti, P., Patel, K. V., Matthews, C. E., Berrigan, D., Van Domelen, D. R., et al. (2012). Association of sedentary time with mortality independent of moderate to vigorous physical activity. *PLoS One*, *7*(6), Article e37696.

Lockhart, J. W., & Weiss, G. M. (2014). The benefits of personalized smartphone-based activity recognition models. In *Proceedings of the 2014 SIAM international conference on data mining* (pp. 614–622).

Magnon, V., Vallet, G. T., & Auxiette, C. (2018). Sedentary behavior at work and cognitive functioning: A systematic review. *Frontiers in Public Health*, *6*, 239.

Paing, A. C., McMillan, K. A., Kirk, A. F., Collier, A., Hewitt, A., & Chastin, S. F. M. (2018). The associations of sedentary time and breaks in sedentary time with 24-h glycaemic control in type 2 diabetes. *Preventive Medicine Reports*, *12*, 94–100.

Saeb, S., Lattie, E. G., Schueller, S. M., Kording, K. P., & Mohr, D. C. (2016). The relationship between mobile phone location sensor data and depressive symptom severity. *PeerJ*, *4*, Article e2537.

Saeb, S., Zhang, M., Karr, C. J., Schueller, S. M., Corden, M. E., Kording, K. P., et al. (2015). Mobile phone sensor correlates of depressive symptom severity in daily-life behavior: An exploratory study. *Journal of Medical Internet Research*, *17*(7), Article e175.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 253–260).

Sedentary Behaviour Research Network (2012). Letter to the editor: standardized use of the terms "sedentary" and "sedentary behaviours". *Applied Physiology, Nutrition, and Metabolism=Physiologie Appliquee, Nutrition Et Metabolisme*, *37*(3), 540—542. http://dx.doi.org/10.1139/h2012-024.

Steinhubl, S. R., Muse, E. D., & Topol, E. J. (2015). The emerging field of mobile health. *Science Translational Medicine*, *7*(283), 283rv3.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. arXiv:1706.03762.

Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., et al. (2014). StudentLife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing* (pp. 3—14).

Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., et al. (2017). StudentLife: Using smartphones to assess mental health and academic performance of college students. In *Mobile health* (pp. 7–33).

Wilmot, E., Edwardson, C., Achana, F., Davies, M., Gorely, T., Gray, L., et al. (2012). Sedentary time in adults and the association with diabetes, cardiovascular disease and death: systematic review and meta-analysis. *Diabetologia*, *55*(11), 2895–2905.

Wu, C., Boukhechba, M., Cai, L., Barnes, L. E., & Gerber, M. S. (2018). Improving momentary stress measurement and prediction with bluetooth encounter networks. *Smart Health*.

Yerrakalva, D., Yerrakalva, D., Hajna, S., & Griffin, S. (2019). Effects of mobile health app interventions on sedentary time, physical activity, and fitness in older adults: Systematic review and meta-analysis. *Journal of Medical Internet Research*, *21*(11), Article e14343.

Zia, J., Tadayon, A., McDaniel, T., & Panchanathan, S. (2016). Utilizing neural networks to predict freezing of gait in parkinson's patients. In *Proceedings of the 18th international ACM SIGACCESS conference on computers and accessibility* (pp. 333–334).