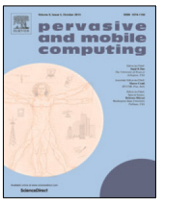
Pervasive and Mobile Computing 93 (2023) 101817

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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



Online continual learning for human activity recognition

Martin Schiemer *, Lei Fang, Simon Dobson, Juan Ye

School of Computer Science, University of St Andrews, UK

ARTICLE INFO

Article history:
Received 15 January 2023
Received in revised form 1 May 2023
Accepted 16 June 2023
Available online 24 June 2023

Dataset link: PAMAP2 Physical Activity Monitoring Data Set (Reference data), Daily and Sports Activities Data Set (Reference data), Human Activity Recognition Using Smartphones Data Set (Reference data), WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set (Reference data)

Keywords:
Human activity recognition
Online continual learning
Deep learning
Pervasive computing

ABSTRACT

Sensor-based human activity recognition (HAR), with the ability to recognise human activities from wearable or embedded sensors, has been playing an important role in many applications including personal health monitoring, smart home, and manufacturing. The real-world, long-term deployment of these HAR systems drives a critical research question: *how to evolve the HAR model automatically over time to accommodate changes in an environment or activity patterns*. This paper presents an online continual learning (OCL) scenario for HAR, where sensor data arrives in a streaming manner which contains unlabelled samples from already learnt activities or new activities. We propose a technique, OCL-HAR, making a real-time prediction on the streaming sensor data while at the same time discovering and learning new activities. We have empirically evaluated OCL-HAR on four third-party, publicly available HAR datasets. Our results have shown that this OCL scenario is challenging to state-of-the-art continual learning techniques that have significantly underperformed. Our technique OCL-HAR has consistently outperformed them in all experiment setups, leading up to 0.17 and 0.23 improvements in micro and macro F1 scores.

1. Introduction

Sensor-based human activity recognition (HAR) is the process of identifying human activities such as walking and cooking from ambient and wearable sensors. HAR is the key enabling technology for applications in health monitoring [1,2], smart home [3], traffic monitoring [4], and personalised training [5]. In recent years, HAR has been extensively investigated [6,7], and numerous machine learning approaches, particularly deep learning [8,9], have been developed to extract features from sensor signals from multiple modalities and discover complicated correlations between features and behaviours [10–12].

The promising performance has potentially driven the increasing deployment of HAR systems in real-world applications. As time passes, individuals alter their activity patterns and engage in new activities, such as experimenting with a new type of exercise due to a change in their health status. This raises an important research question: *how to discover and learn new types of activities over time*. There is a pressing need for a HAR system to incrementally identify and incorporate new types of activities into an existing HAR model. This refers to *continual learning* [13].

More formally, continual learning is the capability of a machine learning model to maintain old knowledge trained on old tasks and incrementally learn and extend the knowledge from new tasks without the need for retraining the model from scratch. In a computer vision application, for instance, a system may be trained with a set of photos to recognise a pre-defined set of objects, and then it could be incrementally updated with new data to classify a new set of objects over

* Corresponding author.
E-mail address: ms400@st-andrews.ac.uk (M. Schiemer).

https://doi.org/10.1016/j.pmcj.2023.101817

time. However, such scenarios might not be feasible for HAR applications that often expect to process streaming sensor data and automatically detect and learn new types of activities with sparsely labelled data. In addition, we often observe distribution shift or concept drift in HAR data due to changes in user behaviour, sensor deterioration, or the addition of new users. These problems are under-explored in the current continual learning fields.

This paper presents an OCL scenario for HAR, which captures the above unique characteristics to HAR. Compared to the existing class-incremental continual learning setting, our scenario does neither assume explicit task boundaries nor abundant labelled training data for new tasks. To tackle this scenario, we propose an end-to-end technique, called *OCL-HAR*, which automatically detects novel activities from streaming sensor data and extends the current HAR model with sparsely labelled data. This technique is specifically designed for HAR, targeting real-world, long-term, deployment challenges.

One of the key challenges in continual learning is *catastrophic forgetting* [14], where the model tends to optimise towards new data but forget old data, leading to degraded performance on predicting old classes. In our technique, we have drawn effective practices from the literature to mitigate the forgetting effect; for example, sample and feature replay [15–19], knowledge distillation [20], and weight normalisation and re-scaling [21].

Another challenge faced in our OCL setting is the scarcity of labels. We assume working on streaming unlabelled data and OCL-HAR will actively detect new activities and select samples for users to annotate. In real-world deployments, it is difficult to acquire high-quality activity labels on sensor data [22], so our technique aims to query users as little as possible. To do so, we have employed a cost-effective, semi-supervised learning technique with the mixup approach [23], which can leverage a very small number of labelled samples and tolerate corrupted labels.

Our main contributions are summarised as follows:

- We design and formalise an online continual learning setting for HAR that takes into account the main challenges including new activities (new class), streaming data (online), and changes in activity patterns and sensor performance (distribution shift).
- We propose a continual learning technique where we integrate new class discovery and learning while only requiring a very constrained number of labels.
- We empirically evaluate our technique with state-of-the-art continual learning methods on a range of HAR datasets. Our results show consistent, superior performance on sparsely labelled data streams with distribution shift.

2. Related work

In the following, we introduce continual learning and its sub-field online continual learning, and compare and contrast our work with relevant studies.

2.1. Continual learning

Continual learning is a popular topic in the field of machine learning and there is an increasing number of techniques proposed every year. In general, these techniques are usually grouped into three categories [24]: dynamic architectures, regularisation, and rehearsal. Dynamic architecture-based approaches increase the capacity of the network with new parameters to learn new tasks, which often are more suitable for large sets of classes; e.g., 100 or 1000 classes and thus more computationally expensive to train. For example, dynamically expandable representation for class incremental learning (DER) [25] freezes the feature representation learnt from previous tasks to preserve the knowledge while at the same time adding additional layers to learn features on each new task. We consider them not suitable for human activity recognition (HAR) systems because of their large computational complexity and memory footprint.

Regularisation-based approaches introduce additional terms to the loss function that constrain the weight updates of the network to prevent compromising the performance of old tasks. For example, *Learning without forgetting (LwF)* [26] is the earliest attempt to employ the knowledge distillation loss [20] to prevent the current model's softmax output from deviating from the previous model's prediction. *Elastic weight consolidation (EWC)* [27] assumes that the posterior probability of the network parameters given the data indicates which parameters are important. Unfortunately, the true posterior is intractable. Thus, it is approximated as a Gaussian with diagonal precision given by the Fisher information matrix (FIM) using Laplace approximation.

Rehearsal-based approaches store a subset of data from previous tasks and mix them with new tasks' data in training. *Incremental classifier and representation learning (iCaRL)* [28] is a class-incremental learning technique that reserves representative samples from each task and combines them with new tasks' data to update the classifier over time. Additionally, it applies the distillation loss similar to LwF. A very similar technique to iCaRL is *incremental learning in online scenario (ILOS)* [29] that uses a combination of the logit output of the old and the current model for the cross entropy loss. The distillation loss is still calculated using the current model output. *Greedy sampler and dumb learner (GDumb)* [30] is developed to question the progress in continual learning in general. It employs memory replay to train a model from scratch for every incoming data batch. Surprisingly, this simple approach outperforms more sophisticated approaches by a large margin. These three approaches have demonstrated strong results in recent continual learning surveys [31,32].

The above rehearsal-based techniques use real samples for replay. Another strand of research is to use generated samples for replay. Cong et al. [33] apply style-transfer generative adversarial neural networks (GAN) where the source

model is frozen to mitigate forgetting while tuning a set of target-specific style parameters for new tasks' data. However, the quality of generated raw data will deteriorate over time and it often requires many samples and a long time to train. deep generative replay-through-feedback (DGR-RtF) [34] uses variational autoencoder (VAE) [35] to generate samples, and adds a classifier to the VAE via feedback connections such that they can train the classifier and VAE simultaneously. This approach has outperformed a GAN-based approach on HAR datasets [36]. Xiang et al. [37] use GANs to generate intermediate CNN features; that is, the output at the intermediate layer of a CNN for replay. The higher layers of CNN and the fully connected layers will be updated to accommodate new knowledge. Our continual learning approach uses both real sample replay and generated feature replay, with the goal to better mitigate the forgetting effect.

2.2. Continual learning in HAR

Continuous learning has recently gained interest in the HAR community. Siirtola et al. [38] propose incremental learning methods to personalise HAR devices for different users. Ye et al. [39] have investigated the driving force behind continual learning for real-world, long-term HAR system deployment and proposed a general framework to address the problem. Jha et al. [31] have conducted empirical evaluation on a wide range of state-of-the-art continual learning techniques on HAR datasets, and identified the best-performing techniques and their computational cost. Ye et al. [36] have applied GANs to generate sensor data on the seen activity classes and mix them with new classes' data to train an incremental classifier. This helps to alleviate the privacy concern of storing real samples. However, GANs suffer from forgetting over time and the quality of generated samples deteriorates. Leite et al. [40] propose a resource-efficient strategy for HAR that utilises expandable networks that grow when new classes have to be learned. They employ the replay of compressed samples selected for maximal variability.

Adaimi et al. [41] argue that HAR based continual learning is still under-explored. They employ prototypical networks and memory replay to develop an online continuous learning system in which prototypes are continuously updated. Further, they enforce inter-class separation through contrastive loss. Zhang et al. [42] examine multimodal learning in a continuous fashion for HAR. They couple EWC and canonical correlation analysis in an effort to exploit the correlations between various activity sensors in order to mitigate catastrophic forgetting.

The main difference between these projects and ours is that in our OCL setting, the data stream does not have any labels, and our technique needs to actively detect the existence of new classes, acquire and propagate labels, and then perform continual learning. Their focus is only on the last part; i.e., continual learning, and we are looking for an end-to-end systematic solution for OCL.

2.3. Online continual learning

OCL is gaining increasing attention as the existing offline setting of continual learning (CL) is considered less realistic for real-world applications [29,43]. Aljundi et al. [44] consider online continual learning as a setting where new samples are arriving in a streaming manner and there is no clear task boundary. Their focus is on optimising the sample selection for replay-based techniques. They leverage gradient-based information to select samples that are likely to form a diverse representation of the seen data.

Mai et al. [45] define OCL where data arrives in small batches a time and the previous batches are inaccessible so that an OCL algorithm is required to have a single pass over the online data stream. They have reviewed a collection of CL techniques and found out that these CL techniques are under-performing under this OCL setting.

Another online continual learning setting is to mix samples from new classes and old classes in a task to create *blurry* data batches [30,46–48]. Koh et al. [48] propose a mixture of *importance based memory management* where they choose samples for their memory buffer by the expected loss decrease per sample, *memory only training* where instead of training the incoming data together with the memory buffer only the updated memory buffer is used for each training step (similar to GDumb) and an adaptive learning rate scheduling.

Our setting is similar to the above blurry setting as new data batches can contain old and new classes, but we do not assume to have access to all the labels immediately and the data within the same class exhibits distribution shift; e.g., this can be caused by different users performing the same activity differently. More specifically, new batches can contain new data from seen users and their activity classes, new emerging users or new classes, or both. Our model has to learn from this stream and make predictions, and as well as seek opportunities to detect new classes and extend the model. This is a more realistic and relevant scenario for a real-world HAR application.

3. Background of continual learning and problem statement

In this section, we will introduce the CL problem and describe the most studied CL scenarios. Then we will define the setting of *online continual learning* in HAR and highlight the difference from CL in classic areas such as computer vision.

Here we focus on the CL problem of a neural network model. Let $\mathcal{T} = [t_1, t_2, \ldots, t_n]$ be a sequence of tasks and each task t_i $(1 \leq i \leq n)$ consist of a finite set of new classes C_i and their training data $\{(x^j, y^j)|y^j \in C_i\}_{j=1}^{m_i}$. Continual learning refers to the process of training a neural network model with the training data of multiple tasks in succession, in order to predict labels for all classes observed to date. The goal is to incorporate new information into an existing model without

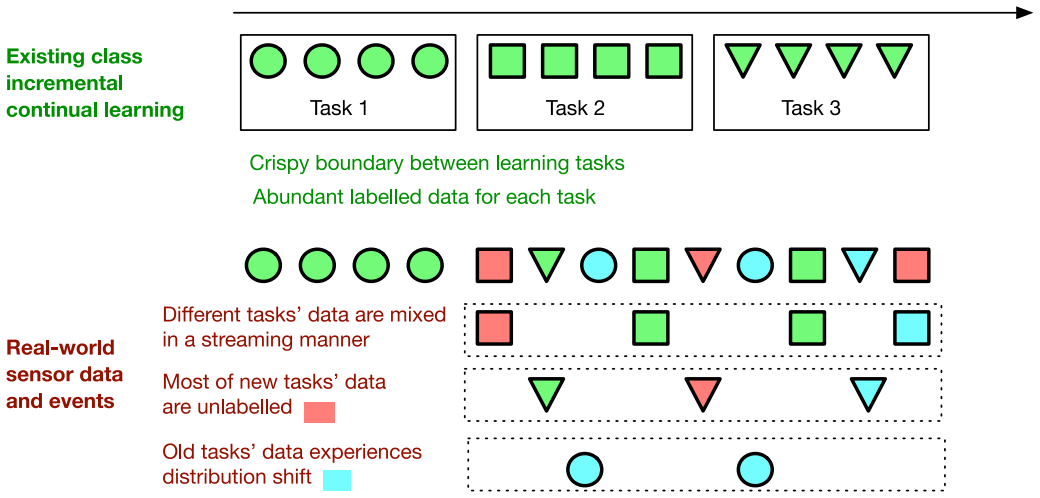


Fig. 1. Class-incremental continual learning vs. online continual learning..

forgetting previous information. Early research efforts concentrated on two continual learning scenarios: task incremental learning (task-IL) and class incremental learning (Class-IL) [49]. In task-IL, a model contains task-specific components; for instance, in a multi-head architecture, each head corresponds to a classifier output head for all the classes in a task. A task ID (e.g., "animals") is always provided during training and prediction to direct the network to a specific component (e.g. "animal classifier"). Class-IL typically lacks task-specific components, and a task ID is not provided during prediction. It is a more difficult scenario and better resembles real-world problems because it is unrealistic to expect a known task ID for an unknown sample.

However, class-IL may not be suitable for HAR systems. Firstly, class-IL still assumes a distinct separation between tasks; however, sensor data frequently arrives in a streaming format, which can mix samples from old and new classes. Secondly, class-IL requires a large number of well-labelled samples for each task, which can be impractical to collect on a deployed HAR system. Thirdly, it often assumes that the distribution of old tasks remains unchanged, whereas in HAR, sensors can degrade and activity patterns can shift. Consequently, the distribution of old classes will typically change over time. To account for these constraints, we present an online scenario for continual learning, as shown in Fig. 1.

Assume that there exists the first task with its training data $\{(x^j, y^j)|y^j \in C_1\}_{j=1}^{m_1}$, which can be used to initialise a model; for example, training a feature extractor and a classifier. After this first task, we will only observe streaming data without task boundaries: $\{x^j\}_{j=1}^{\infty}$, where each sample x_j may belong to a previously seen class $\in C_1$, or a new class. Even when x_j belongs to a seen class, it may be sampled from a different distribution; i.e., from a different user. All these samples do not have a label. An online continual learning algorithm needs to either make a prediction on these samples or identify potential new activities from them and acquire labels in a semi-supervised manner. This setting is different from recent works [50,51] in that the incoming batches are from a fuzzy data stream and not collected from an array of sequential tasks without explicit but implicit task boundaries.

4. Proposed approach

In this section, we illustrate an online continual learning framework for HAR. It operates on a small batch of streaming sensor data and is able to discover new types of activities with which to incrementally evolve the model. Fig. 2 presents the overall pipeline which consists of the following four main steps.

- *A. Training*: Assume we have some initial data that are labelled with a subset of activity classes of a dataset. With the data, we will train a neural network (NN) as a classifier to predict the activity labels provided in the initial data.
- *B. Predicting and discovering*: Working with streaming sensor data, we perform outlier detection and prediction on a small batch.[1]. A batch is a sequence of streaming data that can contain the data from old and new classes, and the data points are not labelled. The batch size is generally smaller than the size of a task, and it depends on the operational device's capacity for incoming data. To preprocess this incoming unlabelled data, outliers and possible members of already seen classes have to be identified. If a sample is an *outlier*, it is stored in an in-memory buffer for step C; otherwise, the NN classifier created in step A is used to provide a label.

[1] To note that batch is different from a *task* in a classic continual learning setting. A task's data often consists of a large number of well-labelled samples on a new set of classes; for example, a few thousand images for *"cat"* or *"dog"* This data will be used for offline training on a continual learning algorithm.

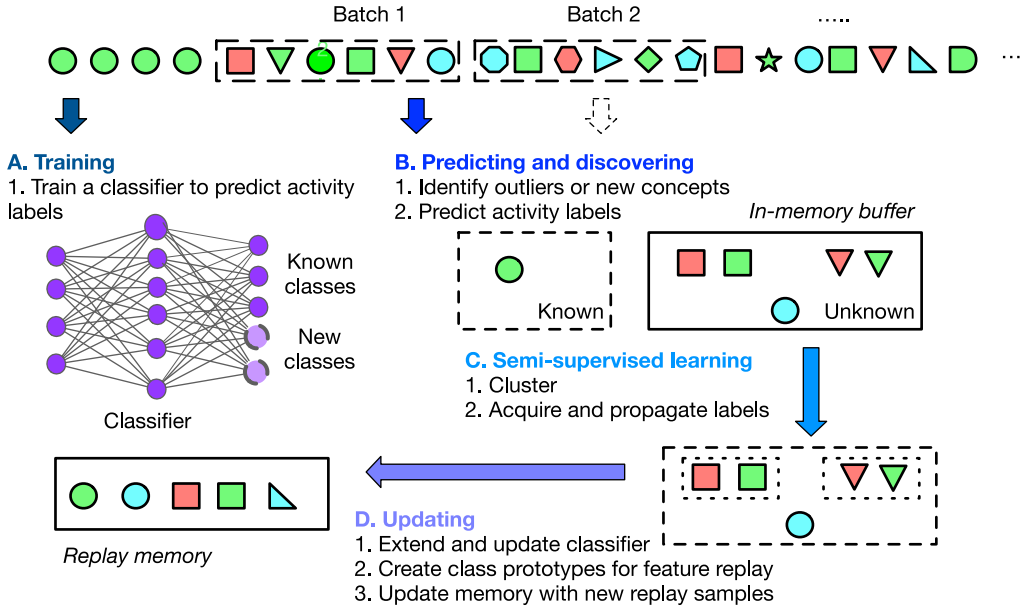


Fig. 2. Workflow of an online continual learning algorithm where newly incoming batches can contain new samples of new and known classes (represented by colour) and/or new and known users (represented by shape).

- *C. Semi-supervised learning*: We process the in-memory buffered samples to discover any new activities. To accomplish this, we will cluster the samples and select a very limited number of representative samples to query end-users for annotation. Then, label propagation is performed on the remaining samples in the buffer. The now-labelled outliers and previously labelled data samples are utilised to update the classifier.
- *D. Updating*: We will extend the architecture of the NN classifier to accommodate new classes and then fine-tune it with replay samples and newly labelled data. In addition, we will update a replay memory with new samples to reflect the changes in data distribution.

Steps B, C, and D are repeated as new streaming data arrive. Each of these steps will be described in detail in the following subsections.

4.1. Prediction and discovery

For each sample in the incoming batch, we first perform outlier detection to filter outlier samples that deviate from the data seen so far. These outlier samples can be drawn from unknown, new classes or from the known classes with similar or different distributions, or they can be noisy samples due to environmental interference or sensor degradation.

A straightforward technique for outlier detection is to assess a models prediction confidence to identify uncertain samples. However, NN classifiers are often over-confident on their predictions [52]. One observation is that uncertain samples might not be the novel classes, but the samples residing at the boundaries of overlapping classes; for example, sitting and standing. Therefore, we are looking for other outlier detection algorithms. Ideally, we would like a low-memory and low-computing outlier detection algorithm that can accurately detect all outliers.

We decide to employ *isolation forests* [53] that generate multiple decision trees that split a dataset iteratively by randomly selecting a feature of the data and then selecting a random value between the maximum and minimum of the feature. Normality can be measured by the average length of the path on the trees required to separate a single sample from the rest. Outliers are identified if their separation requires fewer steps than in-distribution samples [53]. The isolation forests have the following advantages: there is no need for distance or density measurements, the time complexity is linear, it has a small memory footprint, and it is highly scalable to a large number of samples. Most importantly, they do not require the storage of any real data samples which allows us to use them for any constraint environment. We create an isolation forest for each class that has been seen. If an incoming sample is identified to be an outlier by all forests, it will be assigned as such and stored in the in-memory buffer. Otherwise, it will be passed to NN for classification. We have run preliminary test and compared with other techniques such as distance-based method and Dirichlet. Isolation forest can achieve 74.3% on correct outlier detection, outperforming the alternatives.

4.2. Semi-supervised learning

In our setting, streaming data does not have labels. Once the in-memory buffer is full, we will perform semi-supervised learning to acquire labels and update the classifier. Semi-supervised learning has long been applied in scenarios with scarce annotation [54]. The goal is to achieve high accuracy on label propagation and at the same time to require as few real labels as possible. Motivated by this objective, we employ a technique based on clustering. First, we employ *K-means* [55,56] as clustering algorithm on all the outlier samples in the buffer once the buffer is full. The clusters are arranged in descending order by size. Then, we select the top k groups whose sizes are large enough to indicate that they represent recurrent patterns and potentially novel types of activities. For each selected group, a label on the centroid sample is acquired. These k-labels are then spread to the remaining samples in their respective clusters and all newly labelled data are added to the training data. We also experimented with the other off-the-shelf graph-based label propagation algorithms including *Label Propagation* [57] and *Label Spreading* [58], which are more computationally expensive and do not outperform this simple approach in our setting; i.e., 91% of correct label assignment.

4.3. Updating

After preprocessing the incoming new data batch we extend the output layer of the current model by the number of newly detected classes and re-train the model. As we assume no prior knowledge about the total number of classes in the datasets, we need to extend the output layer with the newly discovered classes. To minimise the impact to the model size, we only add neurons to the output layer, and therefore, the increased size of the model is the product of the number of new classes and the number of parameters in the second last layer.

The main challenge of updating the model is *catastrophic forgetting* [13], a well-known problem in continual learning. Neural networks are too plastic to recently learned data. Thus, when the model is optimised to new data, the performance on previously seen data will decrease. To mitigate the forgetting effect, we will adopt regularisation and replay-based techniques that have demonstrated promising performance, as discussed in Section 2.

One of the most commonly applied regularisation algorithms is *knowledge distillation (KD)* [20], which uses a modified cross-entropy loss and aims to retain the knowledge from the previous model by penalising a large update on the model parameters [26]. Let f be a feature extractor and g be its classifier and the KD loss L_{KD} is defined as follows:

$$L_{KD} = -\sum_{i=1}^{c} y'^{(i)}_{old} log y'^{(i)}_{new} \tag{1}$$

$$y'^{(i)} = \frac{(y^{(i)})^{1/T}}{\sum_j (y^{(j)})^{1/T}}, \quad y^{(i)}_{new} = softmax(g^t(f^t(x))), \quad y^{(i)}_{old} = softmax(g^{t-1}(f^{t-1}(x))) \tag{2}$$

where c is the number of known classes. $y'^{(i)}_{old}$ and $y'^{(i)}_{new}$ are the temperature T scaled softmax outputs of the old and new model for class i. g^t, f^t, g^{t-1}, f^{t-1} are the classifiers and feature extractors at the time step t and $t-1$ respectively. The loss L_{KD} aims to minimise the difference of predictions between the current model $g^t(f^t(x_i))$ and the previous model $g^{t-1}(f^{t-1}(x_i))$.

Previous research [30,59] has demonstrated that regularisation alone is insufficient to combat catastrophic forgetting; therefore, it is common to integrate regularisation with replay techniques, i.e., sub-sampling data from seen classes, storing them in memory, and mixing them with data from new classes when re-training the model. To further improve knowledge retention, we introduce a recent approach on feature replay [18,19]. After training at each step, we generate a prototype for each new class. The prototype is based on the final feature representation of the data; that is, the activations in the second last layer of the model. The class prototype is represented by the mean μ_c and variance σ_c:

$$\mu_c = \frac{1}{n^t_c}\sum_{i=1}^{n^t_c} f(x^t_i), \quad \sigma^2_c = \frac{1}{n^t_c}\sum_{i=1}^{n^t_c} (f(x_i) - \mu_c) \odot (f(x_i) - \mu_c) \tag{3}$$

where n^t_c is the number of samples in a class c at the time step t. During the following training steps, these prototypes are used to sample feature vectors for each class from a normal distribution given the μ_c and the average of all class variances.

$$x^{proto}_c \sim \mathcal{N}(\mu_c, \sigma^2_c) \tag{4}$$

These generated features are then replayed to the last layer in the feature relay loss $L_{CEFeatRep}$. It calculates the cross entropy loss between the true class label c and the predicted label on the generated prototype sample x^{proto}_c. The purpose of this loss is to penalise large deviance of the classification layer on the already learnt features.

$$L_{CEFeatRep} = \sum_{c=1}^{C} CE(g(x^{proto}_c), c) \tag{5}$$

The advantage to other generative methods such as GANs [60] and VAEs [35] is that the process is more resource-effective, which is desirable on HAR devices. To maintain the usefulness of the class prototypes even after updating with new data, feature level distillation [15,17,18] on the last layer before the classifier is also used to prevent the model from deviating from its former representations:

$$L_{FeatureKD} = -\frac{1}{n^t}\sum_{i=1}^{n} \|f(x_i) - f^{t-1}(x_i)\|_2 \tag{6}$$

In our OCL setting, we only query a small number of labels and spread labels on the other data via label propagation, which can lead to corrupted labels. To enhance the robustness, we employ mixup [23], a commonly adopted technique in dealing with noisy labels. It works by generating linear combinations of samples and their labels:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \tilde{y} = \lambda y_i + (1-\lambda)y_j \tag{7}$$

where (x_i, y_i) and (x_j, y_j) refer to any two random samples in the current batch, y_i (y_j) is the one-hot encoding of the target labels, and λ is sampled from a beta distribution for each sample. These linear combined samples are then used for training with the cross-entropy loss $L_{CEMixup}$. Putting all the loss components together, we reach the final loss function:

$$L = L_{CEMixup} + L_{KD} + L_{CEFeatRep} + L_{FeatureKD} \tag{8}$$

ALGORITHM 1: General Algorithm for Online Continual Learning

```
input : Batch 1 data b_1 = {(x^j, y^j)|y^j ∈ C_1}_{j=1}^{m_1}
input : Streaming data S = {x^j}_{j=1}^{∞}
build a model with b_1
initialise an in-memory buffer B to host outlier samples
initialise a replay memory M to host replay samples
for x_j ∈ S do
    if x_j is abnormal then
        add it to B
    end
    else predict a class label on x_j ;
    if |B| > threshold then
        run clustering on B
        identify significant clusters
        perform semi-supervised learning to label each significant cluster
        update the model with newly labelled data and replay samples in M
        update M with samples subsampled from new data
    end
end
```

The last preventative measure for catastrophic forgetting is weight alignment which is composed of vector normalisation and re-scaling [21]. The objective is to eliminate the output weight bias caused by the imbalanced training batches. It has been empirically found that imbalanced training batches lead to larger weight vector norms for the majority classes [21,61,62]. A larger norm indicates that a group is regarded as more significant and its classification is more likely. The normalisation is performed in the following way:

$$w_c^{norm} = \frac{w_c}{\|w_c\|} \tag{9}$$

where w_c is the output weight vector of class c. The output-layer weights' vectors are normalised after calculating gradients and before the weight update. For the re-scaling part, after learning a batch, the weights w_i of class i are re-scaled based on the sample ratio of old and new classes:

$$w\prime_i = \frac{{n_{max}}^{\gamma}}{n_i} * w_i \tag{10}$$

where n_{max} is the number of samples of the class with the most samples so far and n_i is the number of samples in class i. The scaling factor γ is a hyperparameter, indicating the scaling strength [63]. The larger γ the stronger the influence of infrequent classes. In the online setting, as we deal with streaming sensor data and we do not assume prior knowledge of the number of training data on all the classes, then n_i varies on each batch. Therefore, we will have to re-scale the weights on each batch update.

At the end of each update, we update a replay memory with a small number of samples for each class for memory replay. The size of the memory buffer is fixed and each of the seen classes is allocated with the same size. The sample selection is performed randomly as more complex strategies like herding [64] need access to many class samples to select a representative set. Algorithm 1 summarises the overall algorithm. Compared to retraining a model from scratch, this way of continually and incrementally updating the model is computationally efficient.

Table 1
Characteristics of the used Datasets.

Dataset	Nr. samples	Nr. features	Nr. classes	Nr. users	Removals	Class balanced	User balanced
PAMAP2	5624	243	11	6	1 class, 3 users	No	No
DSADS	9120	405	19	8	None	Yes	Yes
HAPT	10184	561	11	28	1 class, 2 users	Yes	No
WISDM	23074	91	18	44	None	No	No

5. Experiment setup

In the following we present our experimental setup for OCL. Section 5.1 introduces the used datasets, Section 5.2 explains the evaluation metrics, Section 5.3 defines our protocol and experiment design, Section 5.4 describes the compared techniques and our reasoning for choosing them and Section 5.5 illustrates the hyperparameter settings.

5.1. Datasets

To evaluate our online continual learning scenario, we are looking for datasets that satisfy the following criteria: (1) they have a large set of classes so that we can introduce new classes over time; (2) they exhibit diverse patterns in each class so that we can simulate distribution shifts; for example, a dataset has multiple users; and (3) they use different sensing technologies so that we can assess the generality of our approach. According to these criteria, we have reviewed the existing, publicly available HAR datasets and selected the following four datasets.

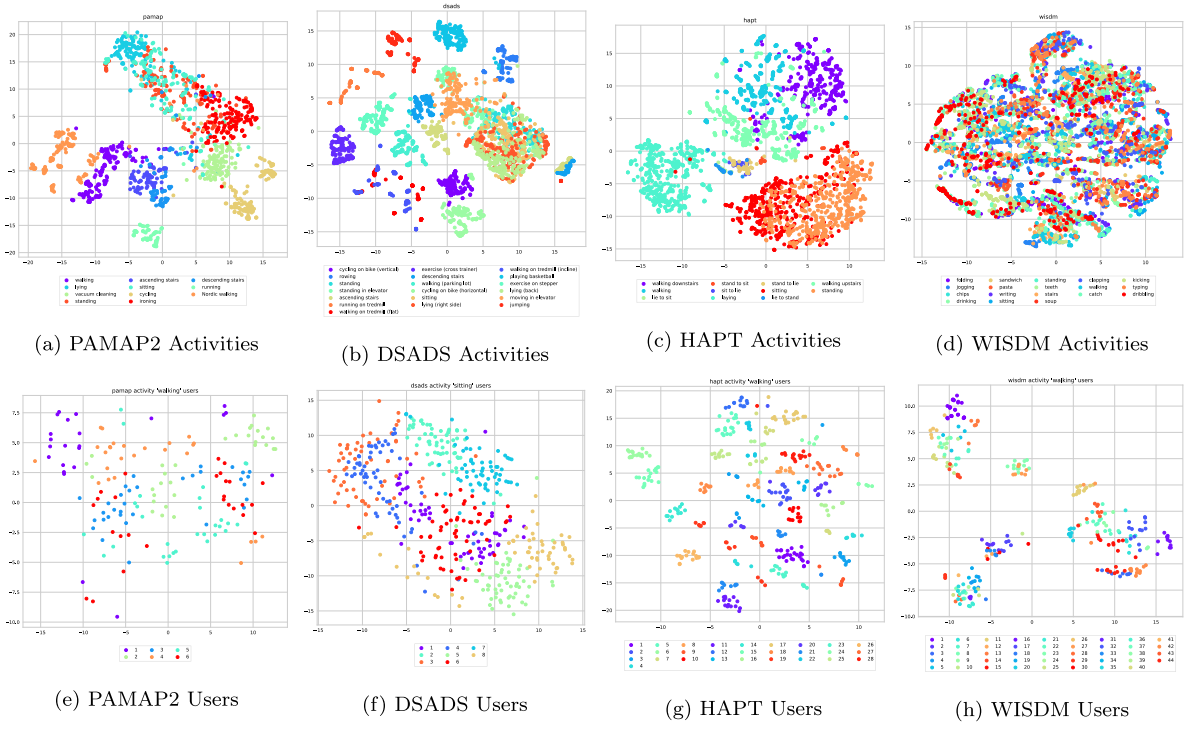
- *Physical Activity Monitoring (PAMAP2)* [65] was collected for activity recognition and intensity estimation purposes. It contains 18 physical activities, including a wide range of everyday, household and sport activities, from 9 users. The sensor data is collected on accelerometers worn on the chest, dominant arm and side ankle, totalling 10 h of data with a sampling frequency of 100 Hz. We use features generated by [66] which contain 12 classes and 6 users. Their feature generation protocol is that 27 features are extracted per sensor on each body part which results in 81 per body part, including mean, standard deviation, and spectrum peak position.
- *Daily and Sports Activities Dataset (DSADS)* [67] contains 19 activity classes such as running, rowing, and sitting. Each user performs each of these activities for 5 min at a 25 Hz sampling frequency. The data samples are collected on 8 users with 5 accelerometer units on each user's torso and extremities. The subjects were not instructed on how the activities should be performed. Here we use the same set of feature generation protocol as PAMAP2 [66].
- *Human Activity Recognition Dataset (HAPT)* [68] contains 12 activities such as standing or walking and postural transitions such as stand-to-sit. It is collected from 30 subjects wearing a smartphone (Samsung Galaxy S II) on their waist with a 50 Hz sampling frequency. The data is split with a fixed-width sliding window of 2.56 s and 50% overlapping. After filters and transformations, features like mean, absolute difference, skewness and more are generated. In the end, Reyes-Ortiz et al. [68] extracted 561 features from accelerometer and gyroscope sensor readings.
- The Smartphone and Smartwatch Activity and Biometrics dataset (WISDM) [69] dataset was collected from 51 subjects performing 18 daily activities including walking, jogging, brushing teeth, eating and drinking, each performed for 3 min. During the data collection each subject either wore a smartwatch; *i.e.*, a LG G Watch on their dominant wrist, or had a smartphone *i.e.*, Samsung Galaxy or Google Nexus 5/5X in their pocket. The accelerometer and gyroscopic data from both watches and phones were collected at a rate of 20 Hz. We use the generated features of [69], where 10-second non-overlapping time windows were used to split the data. The derived features include average, peak, absolute difference, and variances.

We present the characteristics of each dataset in Table 1. Except for DSADS, all the other datasets are imbalanced to various degrees. WISDM is relatively balanced in terms of activity classes but data on each user is imbalanced. For PAMAP2 and HAPT, data is balanced on users but not on the classes. Fig. 3 shows the class distribution for datasets. We can observe user diversity on each class, which allows us to simulate data distribution shifts effectively by iteratively adding new users to the learn-able pool. To allow for consistent evaluation on all users and classes, we filter each dataset such that users that do not have all classes and classes that are severely underrepresented are removed (see Fig. 4).

5.2. Evaluation metrics

We consider F1-scores as a balanced measure of both precision and recall. Because our datasets are imbalanced, we use both micro-F1 and macro-F1 scores.

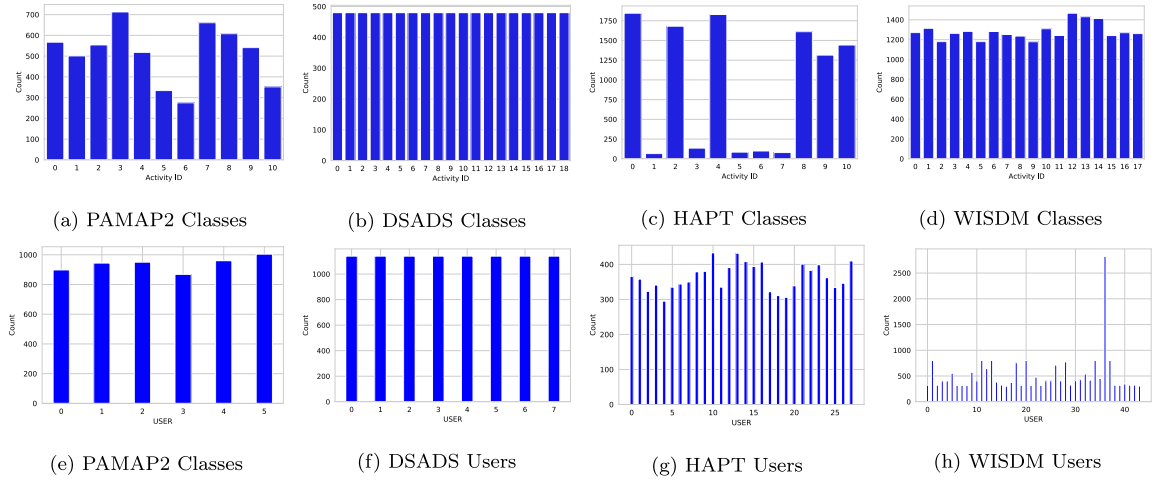
In continual learning, a commonly-considered metric is *forgetting* [70]. It indicates how much knowledge can be retained. The higher the forgetting score, the less knowledge is retained. The forgetting score is measured on a class



(a) PAMAP2 Activities (b) DSADS Activities (c) HAPT Activities (d) WISDM Activities

(e) PAMAP2 Users (f) DSADS Users (g) HAPT Users (h) WISDM Users

Fig. 3. t-SNE plots for users and activity ids on all the datasets.



(a) PAMAP2 Classes (b) DSADS Classes (c) HAPT Classes (d) WISDM Classes

(e) PAMAP2 Classes (f) DSADS Users (g) HAPT Users (h) WISDM Users

Fig. 4. Histograms of user and activity distribution on all the datasets.

c at each updating step t as the difference between the current accuracy and the highest accuracy achieved so far. We average the score on all the seen classes at each step t as below:

$$F^t = \frac{1}{|C|}\sum_{c=1}^{C} \max_{l\in 1,\ldots,t-1}(a_c^l) - a_c^t.$$

5.3. Evaluation protocol

Our goal is to investigate online continual learning with distribution shift. Here distribution shift is simulated as addition of new users' behaviour patterns on known classes. We have designed a systematic evaluation methodology where we can blend new class samples with new users' samples on the seen classes in the incoming batches.

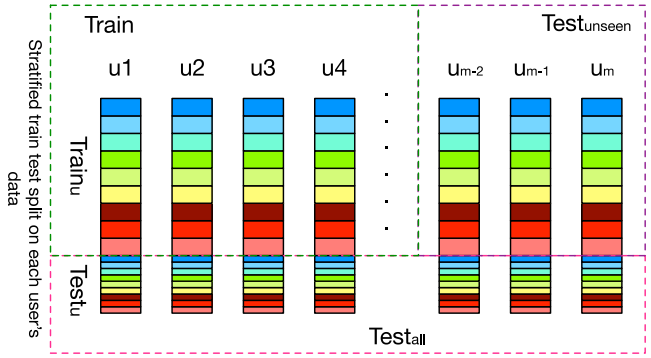


Fig. 5. Train-test splits for our OCL scenario.

For each experiment, we first apply stratified train and test splitting on each user's data, as shown in Fig. 5. Then we shuffle the user list and split the users into train and test users. The training data *Train* consists of the training data on the users in the train set. We have two test data: $Test_{all}$ consisting of the test data on all users and $Test_{unseen}$ consisting of data from the users in the test set. $Test_{unseen}$ and $Test_{all}$ is not used in any training process, and only for testing different purposes: the model's ability to perform on unseen users with $Test_{unseen}$ and the forgetting effect on seen users with $Test_{all}$. That is, with distribution shift, whether the OCL algorithm will still be able to recognise activities on all the seen users or only the recently encountered. We perform the following three different types of experiments:

- *Within-subject*: We take each user's training data $Train_u$ for simulating OCL: streaming their data to our technique for batch processing. That is, the initial batch b_0 contains a subset of the data of a subset of all classes. Incoming batches b_i contain samples of the remaining known classes or new classes. We evaluate the final model on the test data on each user $Test_u$. This is the simplest scenario for learning individual users' activities over time.
- *Between-subject*: We mix all users' training data for simulating OCL in the same way as above. This is a more common setting for a HAR system that infers activities collectively from a large set of users.
- *Between-subject-with-distribution-shift*: This is similar to the above but the main difference is that we do not expose all the training users in one go and we introduce them incrementally in order to model distribution shifts of activity classes; that is, the activity patterns can change over time. Algorithm 2 in Appendix presents how to simulate distribution shifts in OCL.

For all these evaluations, we only use labels in the initial batch, and for all the incoming batches, we perform semi-supervised learning to acquire labels.

5.4. Comparison techniques

As mentioned in Section 2, we will focus on memory replay-based techniques and select the ones that have shown promising results in a recent survey [45] and are commonly used as baselines for similar works [32,36]: iCaRL [28], ILOS [29], GDumb [30], and DGR-RtF [34]. In addition, we include the fine-tuning *Baseline* in which we only naively update the classifier with newly labelled data, without the use of regularisation nor memory replay. We also add a *Replay* technique that simply adds all samples from the memory buffer to the current batch during training and only employs cross-entropy loss without any other regularisation. As all of these comparison techniques are not originally designed for online continual learning, especially unable to deal with unlabelled data, we will update them with the same output from our semi-supervised learning step.

5.5. Hyperparameter settings and implementation details

For all experiments we are using fully connected neural networks with 2 hidden linear layers and the number of neurons for all layers is the same as the number of input features for each dataset (e.g. 243 neurons for each layer for PAMAP2). The output-layer has as many neurons as there are currently found classes. ReLU is used as as activation function. The initial training batch is trained for 500 epochs and the incoming streaming batches for 100 epochs to accommodate the smaller streaming data batches. We use the ADAM optimiser with an initial learning rate of 0.0001, beta values of 0.9 and 0.999, epsilon 1e-08 and no weight decay. We reset the optimiser for each incoming batch, because we do not want to consider data to be of different importance. The λ value for all the distillation methods is set to 1. The α and β values for the beta distribution to sample the mix ratio λ in Eq (7) are set to 1.

Table 2
The mean and standard deviation of micro and macro F1 scores of OCL-HAR and other comparison techniques on the *within-subject* experiment. The highest scores are highlighted in bold.

	PAMAP2 Micro/Macro F1	DSADS Micro/Macro F1	HAPT Micro/Macro F1	WISDM Micro/Macro F1
Baseline	0.21(0.06)/0.11(0.05)	0.11(0.03)/0.06(0.02)	0.23(0.08)/0.09(0.05)	0.12(0.04)/0.06(0.04)
Replay	0.68(0.10)/0.65(0.11)	0.73(0.07)/0.67(0.08)	0.81(0.11)/0.76(0.16)	0.68(0.15)/0.62(0.14)
iCaRL	0.71(0.13)/0.67(0.14)	0.72(0.05)/0.67(0.06)	**0.90**(0.07)/**0.85**(0.1)	0.68(0.15)/0.63(0.17)
ILOS	0.56(0.15)/0.47(0.16)	0.62(0.08)/0.54(0.1)	0.72(0.19)/0.68(0.19)	0.58(0.19)/0.50(0.22)
GDumb	0.42(0.2)/0.34(0.23)	0.58(0.13)/0.51(0.14)	0.79(0.26)/0.74(0.29)	0.47(0.14)/0.37(0.11)
DGR-RtF	0.39(0.14)/0.23(0.10)	0.21(0.08)/0.11(0.06)	0.53(0.27)/0.37(0.28)	0.13(0.07)/0.05(0.05)
OCL-HAR	**0.85**(0.04)/**0.84**(0.05)	**0.88**(0.05)/**0.87**(0.05)	0.89(0.1)/0.79(0.17)	**0.74(0.13)**/**0.69**(0.15)

Table 3
The mean and standard deviation of micro and macro F1 scores of OCL-HAR and other comparison techniques on the *between-subject* experiment; evaluating on the test data $Test_{all}$ of all the subjects. The highest scores are highlighted in bold.

	PAMAP2 Micro/Macro F1	DSADS Micro/Macro F1	HAPT Micro/Macro F1	WISDM Micro/Macro F1
Baseline	0.23(0.06)/0.14(0.05)	0.15(0.05)/0.09(0.03)	0.29(0.13)/0.14(0.12)	0.20(0.03)/0.12(0.03)
Replay	0.54(0.07)/0.46(0.07)	0.50(0.06)/0.44(0.07)	0.53(0.08)/0.42(0.10)	**0.22**(0.03)/**0.15**(0.03)
iCaRL	0.52(0.06)/0.44(0.07)	0.50(0.06)/0.43(0.05)	0.56(0.08)/0.45(0.10)	0.22(0.03)/0.15(0.03)
ILOS	0.47(0.11)/0.38(0.11)	0.37(0.06)/0.29(0.04)	0.47(0.09)/0.35(0.08)	0.20(0.02)/0.12(0.02)
GDumb	0.26(0.14)/0.18(0.14)	0.14(0.06)/0.06(0.05)	0.41(0.12)/0.33(0.16)	0.06(0.01)/0.01(0.01)
DGR-RtF	0.52(0.06)/0.44(0.09)	0.20(0.04)/0.11(0.04)	0.56(0.04)/0.29(0.05)	0.15(0.03)/0.08(0.02)
OCL-HAR	**0.67**(0.06)/**0.64**(0.06)	**0.68**(0.04)/**0.67**(0.06)	**0.60**(0.10)/**0.52**(0.11)	0.19(0.03)/0.11(0.02)

We adopt the implementations of the comparison techniques [31], and for iCaRL, we use a neural network as a classifier, rather than nearest-of-mean classifier, because the results with nearest-of-mean classifiers are very poor on our sensor data. For DGR-RtF [34] the z-layer size is set to 1/10 of the input features and 300 samples are generated for each step. All the implementations are in PyTorch [71] and will be made available.

6. Results and discussion

In this section, we present and discuss the results of our experiments.

6.1. Overall performance of OCL on between-subject-with distribution

This subsection will report the results of the three experiments listed in Section 5.3. For this part, we set up 20 training steps after the initial training, 1 labelling query per training step, and a total of 20 samples for memory replay of the seen classes. Table 2 presents the micro and macro F1 scores of the *within-subject* experiment. OCL-HAR outperforms the other comparison techniques; specifically, the improvement over the second best-performing techniques is 0.14 and 0.17 on PAMAP2, 0.15 and 0.2 on DSADS, and 0.06 and 0.06 on WISDM in micro and macro F1 respectively. It scores lower on HAPT than iCaRL with 0.01 and 0.06 in micro and macro F1.

Table 3 presents the performance on the *between-subject* experiment. OCL-HAR outperforms the other techniques on most of the datasets; more specifically, 0.13 and 0.18 on PAMAP2, 0.18 and 0.23 on DSADS, and 0.04 and 0.07 on HAPT in micro and macro F1. On WISDM, all techniques struggle to learn the diverse user patterns. OCL-HAR is slightly lower than Replay and iCaRL within 0.03 and 0.04 of micro and macro F1.

Tables 4 and 5 present the performance of the *between-subject-with-distribution-shift* experiment, where we systematically increase the subset of available users and classes to simulate distribution shifts. Table 4 refers to the results on $Test_{all}$ and Table 5 on $Test_{unseen}$. Similar to the above, OCL-HAR outperforms the other techniques with large margins on both: for example, 0.17 and 0.23 on PAMAP2 in micro and macro F1 for $Test_{unseen}$.

Cross these three experiments, the *within-subject* experiment is about continual learning on individual users' activities, while the other two experiments are more challenging. *Between-subject* introduces more users with diverse patterns and *between-subject-with-distribution-shift* requires to take into account generalisation on unseen users and distribution shift of the activity classes. We can see that the overall results for the techniques are affected by the increasing harshness in our scenarios. This highlights the advantage of studying HAR data for online continual learning scenarios: the simulated distribution shift compromises the training for all methods noticeably; for example, all the methods achieve lower

Table 4
The mean and standard deviation of micro and macro F1 scores of OCL-HAR and other comparison techniques on the *between-subject-with-distribution-shift* experiment; evaluating on the test data $Test_{all}$ of all the subjects. The highest scores are highlighted in bold.

	PAMAP2 Micro/Macro F1	DSADS Micro/Macro F1	HAPT Micro/Macro F1	WISDM Micro/Macro F1
Baseline	0.21(0.06)/0.14(0.06)	0.18(0.1)/0.12(0.09)	0.24(0.09)/0.1(0.04)	0.17(0.02)/0.09(0.02)
Replay	0.53(0.08)/0.46(0.09)	0.46(0.04)/0.39(0.05)	0.52(0.08)/0.39(0.09)	**0.21**(0.02)/**0.13**(0.02)
iCaRL	0.54(0.07)/0.45(0.08)	0.49(0.02)/0.41(0.02)	0.50(0.10)/0.42(0.11)	0.19(0.02)/0.10(0.02)
ILOS	0.42(0.09)/0.35(0.09)	0.35(0.06)/0.25(0.06)	0.44(0.09)/0.32(0.1)	0.17(0.02)/0.09(0.02)
GDumb	0.19(0.09)/0.11(0.09)	0.17(0.10)/0.09(0.09)	0.39(0.17)/0.28(0.16)	0.07(0.01)/0.02(0.01)
DGR-RtF	0.30(0.08)/0.25(0.1)	0.17(0.05)/0.08(0.04)	0.41(0.07)/0.22(0.05)	0.15(0.02)/0.07(0.02)
OCL-HAR	**0.66**(0.06)/**0.64**(0.06)	**0.66**(0.03)/**0.63**(0.04)	**0.59**(0.06)/**0.47**(0.10)	0.17(0.02)/0.10(0.02)

Table 5
The mean and standard deviation of micro and macro F1 scores of OCL-HAR and other comparison techniques on the *between-subject-with-distribution-shift* experiment and the results are evaluated on the test data $Test_{unseen}$ of the test subjects. The highest scores are highlighted in bold.

	PAMAP2 Micro/Macro F1	DSADS Micro/Macro F1	HAPT Micro/Macro F1	WISDM Micro/Macro F1
Baseline	0.25(0.06)/0.16(0.06)	0.16(0.11)/0.10(0.10)	0.36(0.06)/0.15(0.04)	0.13(0.03)/0.07(0.02)
Replay	0.49(0.09)/0.4(0.09)	0.43(0.07)/0.36(0.06)	0.53(0.08)/0.42(0.10)	**0.17**(0.03)/**0.10**(0.02)
iCaRL	0.49(0.05)/0.4(0.06)	0.48(0.04)/0.39(0.06)	0.50(0.11)/0.44(0.10)	**0.17**(0.02)/**0.10**(0.03)
ILOS	0.42(0.09)/0.35(0.09)	0.33(0.06)/0.24(0.06)	0.44(0.13)/0.30(0.11)	0.16(0.02)/0.09(0.02)
GDumb	0.21(0.10)/0.11(0.11)	0.16(0.09)/0.08(0.07)	0.41(0.16)/0.32(0.17)	0.07(0.01)/0.01(0.01)
DGR-RtF	0.29(0.06)/0.16(0.05)	0.20(0.05)/0.11(0.04)	0.50(0.11)/0.28(0.11)	0.13(0.01)/0.05(0.01)
OCL-HAR	**0.66**(0.08)/**0.63**(0.1)	**0.61**(0.04)/**0.56**(0.04)	**0.67**(0.05)/**0.59**(0.05)	**0.17**(0.03)/0.09(0.03)

accuracies on the *between-subject-with-distribution-shift* experiment. Thus, exemplifying that the standard data split prevalent in previous research may not be able to reflect the domain shift challenge in continual learning.

Between these comparison techniques, Replay boosts the performance even only with a small memory budget, while the impact of knowledge distillation is not significant in that the scores of iCaRL and ILOS are not much better than Replay. Even though demonstrating good performance on computer vision datasets, GDumb does not work well on the HAR datasets, which might be caused by the largely overlapping class patterns between activities such as sitting and standing. Also, it is challenged by fewer training data overall and in the memory buffer than in image scenarios. DGR-RtF, as a generative technique, works well on the majority classes in an imbalanced dataset, but struggles to generate representations that can separate overlapping activities. This can be seen by comparing the performance of the imbalanced PAMAP2 and HAPT with the one on DSADS.

On WISDM, the results from all the techniques on the *within-subject* experiment reported in Table 2 are significantly higher than the results on the other two experiments. The reason might be that WISDM has a much more diverse set of users (e.g. 44) as shown in Fig. 3(h) so that the models that we learnt over time cannot be generalised across them in the harsher scenarios. Besides, WISDM has a large number of activity classes and the subtle difference between some of these classes can lead to poorer performance.

As our goal is online continual learning with distribution shift, the following results will be only reported in the *between-subjects-with-distribution-shift* experiment and evaluated on the test data $Test_{unseen}$ of the unseen users.

6.2. Incremental update

Following the same setup, here we look at the degradation of performance over each training step. Fig. 6 compares micro F1, macro F1, and forgetting scores of our technique and the other comparison techniques on four datasets. On PAMAP2 and DSADS, our technique achieves the highest micro and macro F1 scores (0.66/0.63 and 0.61/0.56), and has the lowest forgetting scores (0.24 and 0.28). It demonstrates that our technique can quickly adapt to newly discovered classes while not suffering much from forgetting. On HAPT, our technique achieves the highest final scores (0.67/0.59). We can see that all methods struggle with HAPT's class imbalance as the macro F1 score is significantly weaker than the micro F1 score. For example, DGR-RtF learns to only generate samples for the majority classes; therefore, it produces relatively strong micro scores (0.54) but much lower macro F1 (0.31) scores. GDumb performs very poorly, because there is very little data to replay, mostly not even better than the baseline. OCL-HAR's curve fluctuates only slightly. Hence, it demonstrates good performance at each inference step.

On WISDM, all models struggle in this harsher setting and only achieve performance slightly better than the baseline.

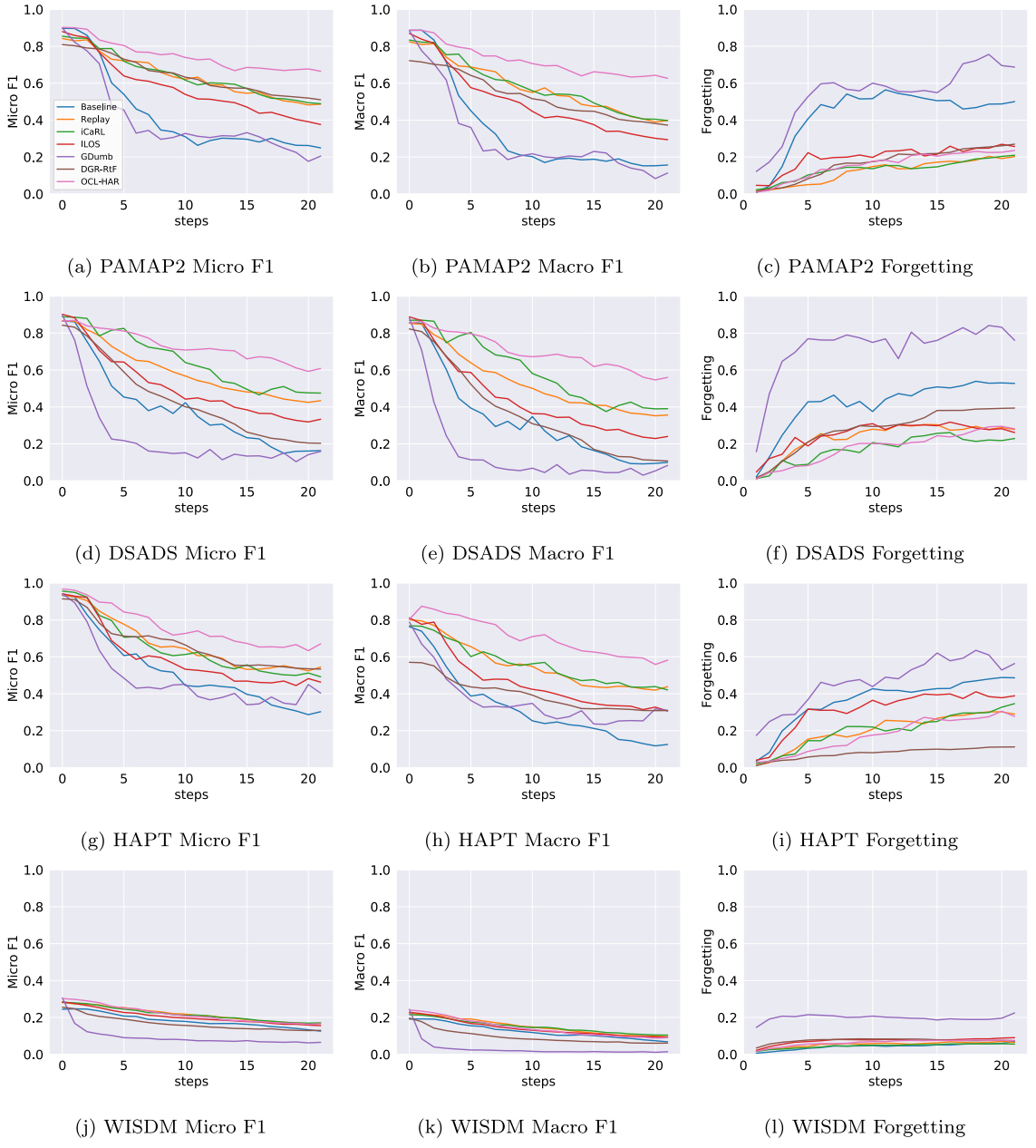


Fig. 6. Comparison of micro F1, macro F1, and forgetting scores between our technique and the comparison techniques on four datasets. The scores are obtained on the test data $Test_{unseen}$ of the unseen subjects. For F1 scores the higher and for forgetting the lower the better.

6.3. Computation cost

Table 6 displays the average training time for each model across all datasets. Not surprisingly, the least expensive models are Baseline and Replay, as they require no additional computation other than a larger batch size due to replay. The second fastest group is iCaRL and ILOS, where only knowledge distillation has been used for regularisation. DGR-RtF learns a generator and a classifier, which has incurred much longer training time. Our technique, OCL-HAR, has also a longer training time, which is due to learning prototypes and generating low-level features for feature replay. Overall, the total training time after 20 steps to learn up to 19 classes is between 2 and 4.25 min (3.3 and 6.7 s per step after the initial step). The inference time is similar between OCL-HAR and the other techniques, as they share the same network architecture, which is around 0.0002 s per sample. Both training and inference are considered acceptable for HAR applications.

Table 6
Mean and standard deviation of training times (in seconds) on all datasets.

	PAMAP2	DSADS	HAPT	WISDM
Baseline	58.59(1.7)	92.98(7.9)	73.73(2.17)	113.73(3.72)
Replay	60.19(0.66)	99.41(9.15)	75.56(1.87)	117.08(6.06)
iCaRL	67.62(1.45)	114.51(12.34)	81.57(2.37)	122.48(4.36)
ILOS	68.99(1.73)	105.88(7.61)	83.45(2.19)	126.52(4.75)
GDumb	60.3(1.01)	103.94(5.97)	74.74(2.49)	125.14(5.81)
DGR-RtF	144.32(3.66)	165.01(2.24)	147.11(4.84)	206.62(4.36)
OCL-HAR	125.81(1.43)	151.91(4.87)	136.76(4.88)	255.69(8.39)

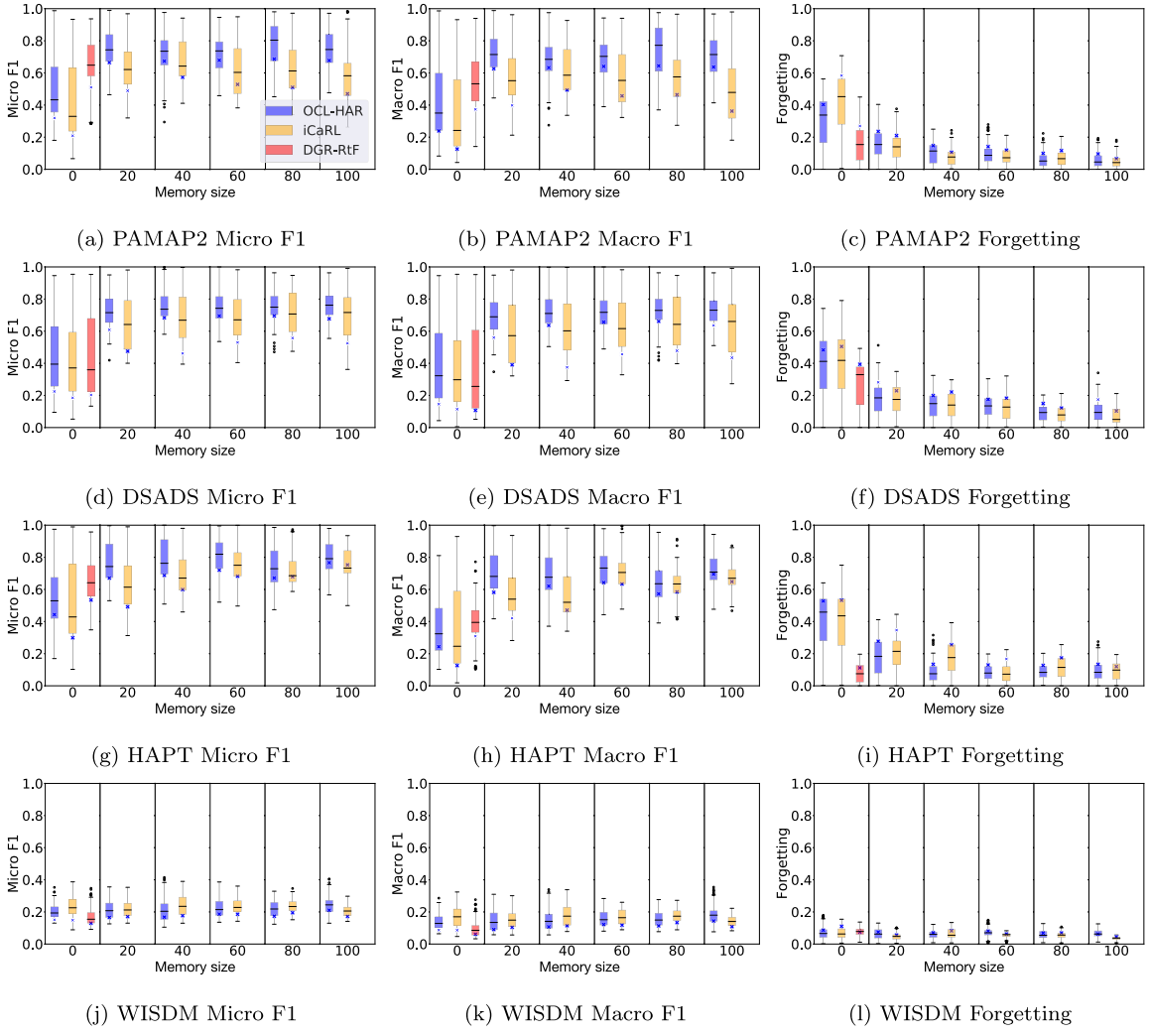


Fig. 7. The comparison of micro F1, macro F1, and forgetting scores between OCL-HAR and two best performing baselines on different memory sizes. Average final values marked with X.

6.4. Impact of memory sizes

The number of replay samples stored in memory can have a significant impact on the performance of continual learning, as the greater the number of available samples on the seen classes, the less forgetting occurs. Fig. 7 depicts the box-plots[2] of micro F1, macro F1, and forgetting scores for OCL-HAR and two representative techniques. DGR-RtF does not

[2] The data for all the box-plots are aggregated over each time step for all the runs.

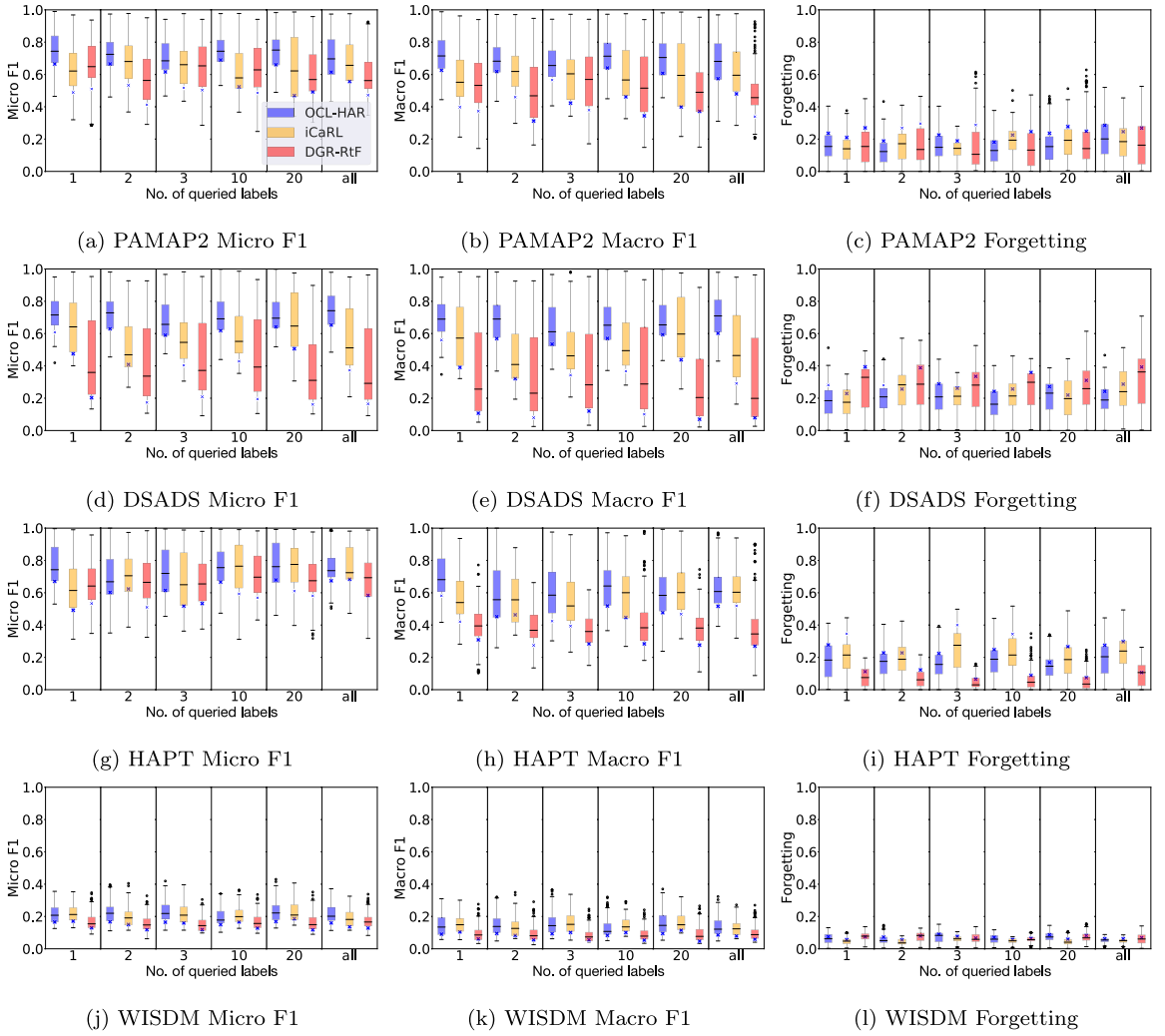


Fig. 8. Micro F1, macro F1, and forgetting scores of OCL-HAR, iCaRL, DGR-RtF on different numbers of labels being acquired. Average final values marked with X.

use any real samples for replay, so its results are only present in the first box where the memory size is 0. When there are no replay samples, DGR-RtF performs the best for PAMAP2 and HAPT. OCL-HAR consistently outperforms iCaRL due to its usage of feature replay. When we increase the memory size to 20, which corresponds to one or two samples per class, our method consistently achieves the highest performance. OCL-HAR has a larger improvement than iCaRL; from (0.32/0.24) to (0.66/0.63) on PAMAP2, from (0.22/0.15) to (0.61/0.56) on DSADS, and from (0.44/0.24) to (0.67/0.59) on HAPT in micro and macro F1. The forgetting scores of OCL-HAR drop largely, too. This result is consistent with the theoretical conclusions [59], indicating that memory replay plays an important role in achieving optimal continuous learning.

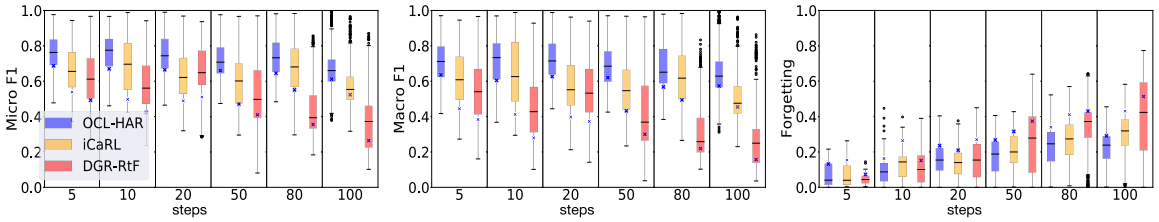
In addition, OCL-HAR does not fluctuate significantly as memory size increases, indicating that it does not require a large number of replay samples. Similarly, iCaRL converges when the memory increases to 20 or 40.

6.5. Impact of labels being queried

To obtain labels for new classes, our method uses a semi-supervised learning strategy. Fewer queries are preferable; therefore, we evaluate how performance is impacted by the number of labels queried. Fig. 8 presents the performance for various numbers of queries, where *all* denotes that labels are acquired for all samples in the buffer. The results demonstrate that our semi-supervised method does not require a large number of queries and that a single query per training step is sufficient to achieve high performance.

Table 7
The mean and standard deviation of micro and macro F1 scores, when all the labels at each training step are available. OCL-HAR-A assumes that all labels are available, while OCL-HAR-1 only acquires 1 label per step.

	PAMAP2 Micro/Macro F1	DSADS Micro/Macro F1	HAPT Micro/Macro F1	WISDM Micro/Macro F1
Baseline	0.24(0.13)/0.13(0.11)	0.21(0.06)/0.11(0.06)	0.71(0.21)/0.45(0.16)	0.10(0.06)/0.05(0.05)
iCaRL	0.59(0.10)/0.55(0.11)	0.60(0.07)/0.53(0.08)	**0.84**(0.05)/**0.67**(0.06)	0.13(0.03)/0.08(0.02)
DGR-RtF	0.24(0.08)/0.14(0.05)	0.19(0.04)/0.08(0.04)	0.66(0.17)/0.36(0.11)	0.09(0.03)/0.04(0.02)
OCL-HAR-A	**0.71**(0.08)/**0.67**(0.09)	**0.71**(0.05)/**0.68**(0.07)	0.83(0.10)/**0.67**(0.08)	**0.17**(0.03)/**0.13**(0.03)
OCL-HAR-1	0.66(0.08)/0.63(0.10)	0.61(0.04)/0.56(0.04)	0.67(0.05)/0.59(0.05)	0.17(0.03)/0.09(0.03)



(a) PAMAP2 Micro F1 (b) PAMAP2 Macro F1 (c) PAMAP2 Forgetting

Fig. 9. Micro F1, macro F1, and forgetting scores of OCL-HAR, iCaRL, and DGR-RtF on different numbers of training steps. Average final values marked with X.

Additionally, we conduct a sanity check on a scenario in which all labels for each training step are available. This situation resembles class-incremental learning or previous online learning scenarios. As shown in Table 7, OCL-HAR outperforms the other techniques in three out of four datasets. When only querying one label per training step, it achieves comparable but worse scores for PAMAP2, DSADS and HAPT compared to having all the labels available; 0.71/0.67 versus 0.66/0.63, 0.71/0.68 versus 0.61/0.56 and 0.83/0.67 versus 0.67/0.59 for micro and macro F1, respectively. Compared to the results in Table 3, the availability of all the labels improves iCaRL's performance; for example, 0.10, 0.12, and 0.34 in micro F1 on PAMAP2, DSADS, and HAPT respectively. This demonstrates that OCL-HAR can tackle insufficient, noisy labels better than the comparison techniques.

6.6. Impact of the number of training steps

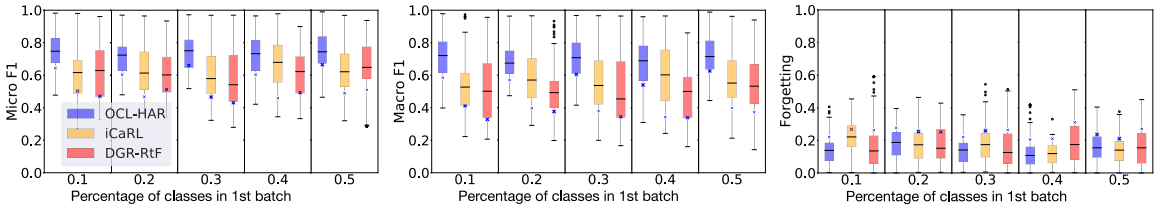
In most cases, neural networks are highly plastic, and their performance degrades after a series of incremental updates. We now assess the impact of the number of training steps on performance. As shown in Fig. 9, our technique performs consistently well over training steps ranging from 5 to 100. The forgetting scores increase slightly when the training step increases, suggesting that after many iterations of update the model is more likely to forget the seen classes; especially, the classes learnt at the beginning. DGR-RtF suffers more than OCL-HAR and iCaRL as it needs to update the generator and model, which both may have the forgetting effect.

6.7. Impact of initial classes

Here, we examine the effect of the number of initial classes on performance, assuming that the more classes fed to the network in the first task, the better it can learn. Fig. 10 shows that OCL-HAR can still perform well when only 10% of classes are included in the first batch. This shows that the size of the to information to be integrated over time only affects the performance marginally.

6.8. Ablation study

As OCL-HAR consists of multiple components to mitigate the forgetting effect, an ablation study is conducted to determine which components contribute the most. As shown in Table 8, when we do not store any replay samples (i.e., the memory size is 0), feature replay and distillation will improve performance by 0.06 and 0.03 in micro and macro F1. Mixup and vector normalisation will nearly double the performance, from 0.21 to 0.41 in micro F1 and 0.14 to 0.30 in macro F1. When we begin to use the replay samples during the model update, the performance improves by a factor of three; therefore, we conclude that real sample replay techniques result in significantly better performance than prototype-based feature replay techniques. Mixup and vector normalisation also contribute significantly. Lastly, adding vector re-scaling gives an additional performance boost.



(a) PAMAP2 Micro F1 (b) PAMAP2 Macro F1 (c) PAMAP2 Forgetting

Fig. 10. Micro, macro F1, and forgetting scores of OCL-HAR on PAMAP2 on different percentages of classes in the first batch. Average final values marked with X.

Table 8
Ablation Study of components in OCL-HAR. F.R refers to feature replay, F.D to feature distillation, M.R to memory replay, K.D to knowledge distillation, V.N to vector normalisation, and V.S to vector re-scaling.

	Micro F1	Macro F1	Training time (in seconds)
Baseline	0.21	0.14	58.59
F.R. + F.D (memory = 0)	0.27	0.17	73.84
F.R. + F.D + mixup + V.N + V.S (memory = 0)	0.41	0.30	125.5
M.R + K.D (memory = 20)	0.49	0.4	67.62
M.R + K.D + mixup + V.N (memory = 20)	0.60	0.54	72.53
F.R. + F.D + M.R + K.D + mixup + V.N (memory = 20)	0.62	0.58	102.49
F.R. + F.D + M.R + K.D + mixup + V.N + V.S (memory = 20)	0.66	0.63	125.81

7. Conclusion and future work

This paper presents an online continual learning setting that is currently under-investigated in the field of continual learning. It neither assumes explicit task boundaries nor has abundant labelled samples, which makes it challenging for the existing continual learning techniques. Even though motivated from HAR systems, this OCL setting can be applicable to other real-world applications that process streaming data and experience constant change in class distribution.

To tackle the problem, we have proposed an OCL-HAR technique that combines semi-supervised learning with effective practices from continual learning including real sample and feature replay, knowledge distillation, and weight vector normalisation and re-scaling. This leads to an efficient HAR model that only needs to acquire few labels on selected samples and can be extended over time while mitigating the forgetting effect. To accommodate sparse, potentially corrupted labels, we have employed the mixup technique that has further improved the robustness of the technique. The evaluation results have shown that our technique has outperformed the state-of-the-art continual learning techniques on the OCL setting and is less sensitive to the initial class set and training steps. The training time is slightly higher than some of the baselines but still acceptable for HAR applications; that is, up to 7 s per update. Therefore, we consider it suitable for frequent, incremental updates over long-term deployment.

In the future, we will look into few-shot learning techniques [72] to improve the prediction accuracy on scarcely labelled data. More advanced sample selection techniques to reserve more representative samples for replay will be explored. We will also deploy the technique on devices for online and practical evaluation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

PAMAP2 Physical Activity Monitoring Data Set (Reference data) (UCI Machine Learning Repository)
Daily and Sports Activities Data Set (Reference data) (UCI Machine Learning Repository)
Human Activity Recognition Using Smartphones Data Set (Reference data) (UCI Machine Learning Repository)
WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set (Reference data) (UCI Machine Learning Repository)

Acknowledgment

This work is partly funded by Leverhulme Research Project Grant RPG-2021-355.

Appendix

Sampling Algorithm

ALGORITHM 2: Sampling Online Continual Learning Tasks

input : Training Data $D = \{(x^j, y^j, u^j)|y^j \in C, u^j \in U\}_{j=1}^{m}$, where C is the whole class set and U is the training users
input : p_1 is the percentage of classes in the first task
input : p is the percentage of training data on each user to be sampled in the first task
input : τ is the probability threshold to decide distribution shift and/or class addition

// sample the first task's data
sample p_1% classes C_a from the whole class set C ($C_a \subset C$)
sample initial available users U_a ($U_a \subset U$)
sample p% labelled instances for each available user in U_a for each class in C_{N_1} and add them to D^a
remove D_{t_1} from D
// simulate incoming streaming data
while *True* **do**
    // use a random probability *Prob* to decide whether to add a new user or a new class.
    if $Prob_U > \tau$ **then**
        add a new user $u \in U - U_a$ to U_a
    end
    if $Prob_C > \tau$ **then**
        add a new class $c \in C - C_a$ to C_a
    end
    sample a batch with size b from any available class on any available user $D^t = \{(x^j, u^j)|u^j \in U_a\}_{j=1}^{|b|}$
    remove D^t from D
end

References

[1] Abdulhamit Subasi, Kholoud Khateeb, Tayeb Brahimi, Akila Sarirete, Human activity recognition using machine learning methods in a smart healthcare environment, in: Innovation in Health Informatics, Elsevier, 2020, pp. 123–144.
[2] Sreenivasan Ramasamy Ramamurthy, Indrajeet Ghosh, Aryya Gangopadhyay, Elizabeth Galik, Nirmalya Roy, STAR-Lite: A light-weight scalable self-taught learning framework for older adults' activity recognition, Pervasive Mob. Comput. 87 (2022) 101698.
[3] Valentina Bianchi, Marco Bassoli, Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, Ilaria De Munari, IoT wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment, IEEE Internet Things J. 6 (5) (2019) 8553–8562.
[4] Biswajit Maity, Abdul Alim, Sanghita Bhattacharjee, Subrata Nandi, A deep learning based system to characterize vehicular honks in presence of ambient noise, Pervasive Mob. Comput. 88 (2022) 101727.
[5] Matthias Kranz, Andreas Möller, Nils Hammerla, Stefan Diewald, Thomas Plötz, Patrick Olivier, Luis Roalter, The mobile fitness coach: Towards individualized skill assessment using personalized mobile devices, Pervasive Mob. Comput. 9 (2) (2013) 203–215.
[6] Sreenivasan Ramasamy Ramamurthy, Nirmalya Roy, Recent trends in machine learning for human activity recognition—A survey, WIREs Data Min. Knowl. Discov. 8 (4) (2018).
[7] Charmi Jobanputra, Jatna Bavishi, Nishant Doshi, Human activity recognition: A survey, Procedia Comput. Sci. 155 (2019) 698–703.
[8] Kun Wang, Jun He, Lei Zhang, Attention-based convolutional neural network for weakly labeled human activities' recognition with wearable sensors, IEEE Sens. J. 19 (17) (2019) 7598–7604.
[9] Kun Wang, Jun He, L. Zhang, Sequential weakly labeled multiactivity localization and recognition on wearable sensors using recurrent attention networks, IEEE Trans. Hum.-Mach. Syst. 51 (2020) 355–364.
[10] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, Lisha Hu, Deep learning for sensor-based activity recognition: A survey, Pattern Recognit. Lett. 119 (2019) 3–11.
[11] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, Yunhao Liu, Deep learning for sensor-based human activity recognition, ACM Comput. Surv. 54 (4) (2022) 1–40.
[12] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, Xue Liu, A survey on deep learning for human activity recognition, ACM Comput. Surv. 54 (8) (2022) 1–34.
[13] Michael McCloskey, Neal J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, Psychol. Learn. Motiv. - Adv. Res. Theory (1989).
[14] Robi Polikar, Lalita Udpa, Satish S. Udpa, Vasant Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Trans. SMC Part C Appl. Rev. (2001).
[15] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, Yoshua Bengio, FitNets: Hints for thin deep nets, in: Yoshua Bengio, Yann LeCun (Eds.), ICLR 2015, 2015.
[16] Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, Larry Heck, RILOD: near real-time incremental learning for object detection at the edge, in: SEC 2019, ACM, New York, NY, USA, 2019, pp. 113–126.
[17] Li Chen, Chunyan Yu, Lvcai Chen, A new knowledge distillation for incremental object detection, in: IJCNN 2019, IEEE, 2019, pp. 1–7.
[18] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, Davide Maltoni, Latent replay for real-time continual learning, in: IROS 2020, IEEE, 2020, pp. 10203–10209.
[19] Subhankar Roy, Mingxuan Liu, Zhun Zhong, Nicu Sebe, Elisa Ricci, Class-incremental novel class discovery, in: ECCV 2022, 2022, pp. 317–333.
[20] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, Distilling the knowledge in a neural network, in: NIPS 2014 Workshop, 2015.
[21] Byungju Kim, Junmo Kim, Adjusting decision boundary for class imbalanced learning, IEEE Access 8 (2020) 81674–81685.
[22] Max Schröder, Kristina Yordanova, Sebastian Bader, Thomas Kirste, Tool support for the online annotation of sensor data, in: Proceedings of the 3rd International Workshop on Sensor-Based Activity Recognition and Interaction, ACM, New York, NY, USA, 2016, pp. 1–7.

[23] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz, MixUp: Beyond empirical risk minimization, in: ICLR 2018, 2018.
[24] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, Stefan Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71.
[25] Shipeng Yan, Jiangwei Xie, Xuming He, DER: Dynamically expandable representation for class incremental learning, in: CVPR 2021, 2021.
[26] Zhizhong Li, Derek Hoiem, Learning without forgetting, in: ECCV 2016, 2016, pp. 614–629.
[27] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, Raia Hadsell, Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. USA (2017).
[28] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, Christoph H. Lampert, iCaRL: Incremental classifier and representation learning, in: CVPR 2017, 2017.
[29] Jiangpeng He, Runyu Mao, Zeman Shao, Fengqing Zhu, Incremental learning in online scenario, in: CVPR 2020, IEEE, 2020, pp. 13923–13932.
[30] Ameya Prabhu, Philip H. S. Torr, Puneet K. Dokania, GDumb: A simple approach that questions our progress in continual learning, in: ECCV 2020, 2020, pp. 524–540.
[31] Saurav Jha, Martin Schiemer, Franco Zambonelli, Juan Ye, Continual learning in sensor-based human activity recognition: an empirical benchmark analysis, Inform. Sci. In Press (2021) 1–35.
[32] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, Tinne Tuytelaars, A continual learning survey: Defying forgetting in classification tasks, IEEE Trans. Pattern Anal. Mach. Intell. 44 (7) (2022) 3366–3385.
[33] Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, Lawrence Carin, GAN memory with no forgetting, in: NeurIPS 2020, Vol. 33, 2020.
[34] Gido M. van de Ven, Hava T. Siegelmann, Andreas S. Tolias, Brain-inspired replay for continual learning with artificial neural networks, Nature Commun. 11 (1) (2020) 4069.
[35] Diederik P. Kingma, Max Welling, Auto-encoding variational Bayes, 2013.
[36] Juan Ye, Pakawat Nakwijit, Martin Schiemer, Saurav Jha, Franco Zambonelli, Continual activity recognition with generative adversarial networks, ACM Trans. Internet Things 2 (2) (2021) 1–25.
[37] Ye Xiang, Ying Fu, Pan Ji, Hua Huang, Incremental learning using conditional adversarial networks, in: ICCV 2019, 2019, pp. 6618–6627.
[38] Pekka Siirtola, Juha Röning, Incremental learning to personalize human activity recognition models: The importance of human AI collaboration, Sensors 19 (23) (2019) 5151.
[39] Juan Ye, Simon Dobson, Franco Zambonelli, Lifelong learning in sensor-based human activity recognition, IEEE Pervasive Comput. 18 (3) (2019) 49–58.
[40] Clayton Frederick Souza Leite, Yu Xiao, Resource-efficient continual learning for sensor-based human activity recognition, ACM Trans. Embed. Comput. Syst. 21 (6) (2022) 1–25.
[41] Rebecca Adaimi, Edison Thomaz, Lifelong adaptive machine learning for sensor-based human activity recognition using prototypical networks, Sensors 22 (18) (2022) 6881.
[42] Xiao Zhang, Hongzheng Yu, Yang Yang, Jingjing Gu, Yujun Li, Fuzhen Zhuang, Dongxiao Yu, Zhaochun Ren, HarMI: Human activity recognition via multi-modality incremental learning, IEEE J. Biomed. Health Inf. 26 (3) (2022) 939–951.
[43] Viktor Losing, Barbara Hammer, Heiko Wersing, Incremental on-line learning: A review and comparison of state of the art algorithms, Neurocomputing 275 (2018) 1261–1274.
[44] Rahaf Aljundi, Min Lin, Baptiste Goujaud, Yoshua Bengio, Gradient based sample selection for online continual learning, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.
[45] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, Scott Sanner, Online continual learning in image classification: An empirical survey, Neurocomputing 469 (2022) 28–51.
[46] Rahaf Aljundi, Klaas Kelchtermans, Tinne Tuytelaars, Task-free continual learning, in: CVPR 2019, 2019.
[47] Jihwan Bang, Heesu Kim, Young Joon Yoo, Jung Woo Ha, Jonghyun Choi, Rainbow memory: Continual learning with a memory of diverse samples, in: CVPR 2021, 2021.
[48] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, Jonghyun Choi, Online continual learning on class incremental blurry task configuration with anytime inference, 2021.
[49] Gido M. van de Ven, Tinne Tuytelaars, Andreas S. Tolias, Three types of incremental learning, Nat. Mach. Intell. 4 (12) (2022) 1185–1197.
[50] Rahaf Aljundi, Punarjay Chakravarty, Tinne Tuytelaars, Expert gate: Lifelong learning with a network of experts, in: CVPR 2017, 2017.
[51] German I. Parisi, Vincenzo Lomonaco, Online continual learning on sequences, in: Recent Trends in Learning from Data, 2020, pp. 197–221.
[52] Chuan Guo, Geoff Pleiss, Yu Sun, Kilian Q. Weinberger, On calibration of modern neural networks, in: ICML 2017, 2017.
[53] Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou, Isolation forest, in: ICDM 2008, IEEE, 2008, pp. 413–422.
[54] Riccardo Presotto, Gabriele Civitarese, Claudio Bettini, Federated clustering and semi-supervised learning: A new partnership for personalized hman activity recognition, Pervasive Mob. Comput. 88 (2022) 101726.
[55] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (2) (1982) 129–137.
[56] David Arthur, Sergei Vassilvitskii, K-means++: The advantages of careful seeding, in: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, 07-09-Janu, 2007, pp. 1027–1035.
[57] Xiaojin Zhu, Z. Ghahramani, Learning from Labeled and Unlabeled Data with Label Propagation, School Comput Sci Carnegie Mellon Univ Tech Rep CMUCALD02107, 2002, p. 2865.
[58] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, Bernhard Schölkopf, Learning with local and global consistency, in: Advances in Neural Information Processing Systems, 2004.
[59] Jeremias Knoblauch, Hisham Husain, Tom Diethe, Optimal continual learning has perfect memory and is NP-HARD, in: ICML 2020, Vol. PartF16814, 2020, pp. 5283–5293.
[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. (2014) 2672–2680.
[61] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, Yun Fu, Large scale incremental learning, in: CVPR 2019, IEEE, 2019, pp. 374–382.
[62] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, Shu-Tao Xia, Maintaining discrimination and fairness in class incremental learning, in: CVPR 2020, IEEE, 2020, pp. 13205–13214.
[63] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, Serge Belongie, Class-balanced loss based on effective number of samples, in: CVPR 2019, IEEE, 2019, pp. 9260–9269.
[64] Max Welling, Herding dynamical weights to learn, in: ICML 2009, ACM Press, New York, New York, USA, 2009, pp. 1–8.
[65] Attila Reiss, Didier Stricker, Introducing a new benchmarked dataset for activity monitoring, in: 2012 16th International Symposium on Wearable Computers, IEEE, 2012, pp. 108–109.
[66] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, Philip S. Yu, Stratified transfer learning for cross-domain activity recognition, in: PerCom 2018, IEEE, 2018, pp. 1–10.

[67] Kerem Altun, Billur Barshan, Orkun Tunçel, Comparative study on classifying human activities with miniature inertial and magnetic sensors, Pattern Recognit. 43 (10) (2010) 3605–3620.
[68] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, Davide Anguita, Transition-aware human activity recognition using smartphones, Neurocomputing 171 (2016) 754–767.
[69] Gary M. Weiss, Kenichi Yoneda, Thaier Hayajneh, Smartphone and smartwatch-based biometrics using activities of daily living, IEEE Access 7 (2019) 133190–133202.
[70] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, Philip H.S. Torr, Riemannian walk for incremental learning: Understanding forgetting and intransigence, in: ECCV 2018, 2018.
[71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[72] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, Yihong Gong, Few-shot class-incremental learning, in: CVPR 2020, 2020.