This is the peer reviewd version of the followng article:

Continual Activity Recognition with Generative Adversarial Networks / Ye, J., Nakwijit, P., Schiemer, M., Jha, S., Zambonelli, F.. - In: ACM TRANSACTIONS ON THE INTERNET OF THINGS. - ISSN 2691-1914. - 2:2(2021), pp. 1-25. [10.1145/3440036]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

18/11/2026 17:09

(Article begins on next page)

Continual Activity Recognition with Generative Adversarial Networks

JUAN YE, PAKAWAT NAKWIJIT, MARTIN SCHIEMER AND SAURAV JHA, School of Computer Science, University of St Andrews, UK
FRANCO ZAMBONELLI, Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia, Italy

Continual learning is an emerging research challenge in human activity recognition (HAR). As an increasing number of HAR applications are deployed in real-world environments, it is important and essential to extend the activity model to adapt to the change in people's activity routine. Otherwise, HAR applications can become obsolete and fail to deliver activity-aware services. The existing research in HAR has focused on detecting abnormal sensor events or new activities, however, extending the activity model is currently under-explored. To directly tackle this challenge, we build on the recent advance in the area of lifelong machine learning and design a continual activity recognition system, called *HAR-GAN*, to grow the activity model over time. *HAR-GAN* does not require a prior knowledge on what new activity classes might be and it does not require to store historical data by leveraging the use of Generative Adversarial Networks (GAN) to generate sensor data on the previously learned activities. We have evaluated *HAR-GAN* on four third-party, public datasets collected on binary sensors and accelerometers. Our extensive empirical results demonstrate the effectiveness of *HAR-GAN* in continual activity recognition and shed insight on the future challenges.

CCS Concepts: • **Human-centered computing** → **Ambient intelligence**; *Ubiquitous and mobile computing systems and tools*;

Additional Key Words and Phrases: Generative Adversarial Networks, continual learning, human activity recognition, smart home, accelerometer

ACM Reference Format:
Juan Ye, Pakawat Nakwijit, Martin Schiemer and Saurav Jha and Franco Zambonelli. 2018. Continual Activity Recognition with Generative Adversarial Networks. *ACM Trans. Internet Things* 0, 0, Article 0 (2018), 25 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Sensor-based Human Activity Recognition (HAR) is about inferring daily activities such as exercising or cooking from wearable and environmental sensors [55]. It has great potential in a range of applications in personal health, elderly care, and smart homes [13]. Indeed, human activity recognition based systems are moving out of labs and testbeds into real world deployments. This significantly challenges the current approaches to activity recognition, as now they have to account for all sorts of unpredictable changes constantly occurring in the real world. Users can change their activity routine such as wandering at home at night time due to insomnia. Ideally, we would expect

Authors' addresses: Juan Ye, Pakawat Nakwijit, Martin Schiemer and Saurav Jha, School of Computer Science, University of St Andrews, North Haugh, St Andrews, Fife, KY16 9SX, UK, juan.ye@st-andrews.ac.uk; Franco Zambonelli, Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia, Italy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2577-6207/2018/0-ART0 $15.00
https://doi.org/0000001.0000001

the system to quickly learn such new changes so that we can identify health problems at the early stage. All these changes are unavoidable in the real-world environments. Without proper change management, the system will derive incorrect inference and lead to undesirable consequences. All these challenges lead to an emerging, important research topic in HAR – *continual learning* [53].

To achieve continual learning, we will require (1) an activity model to have a flexible, extensible structure, and (2) a mechanism to automatically extend the model so as to learn new types of activities while not compromising the performance on recognising previous activities (which refers to as 'catastrophic forgetting'). Here, an *activity model* is defined as a probabilistic computational model, which can be any classifier that learns the correlation between sensor features and activity classes and thus is able to predict an activity label on the incoming sensor data.

Current attempts in HAR towards continual learning mainly focus on change detection – discovering new activities [1, 12, 15] and active learning – acquiring user annotations on new activities [22]. They often require to re-build and re-train the model from scratch every time when a new activity class is introduced. Few have studied how to automatically evolve an activity model with new types of activities [53]. However, such capability brings the benefit of retaining the knowledge in the activity model that has accumulated over time while reducing the training cost and the manual configuration and engineering effort. We consider this benefit is essential for long-term, sustainable deployment of any sensor-based HAR system, which is the key objective of our proposed *HAR-GAN*.

To meet the requirements of continual learning, we have designed a novel architecture to integrate *Generative Adversarial Network* (GAN) and an evolvable classifier in continual learning for HAR, called *HAR-GAN*. GAN has made significant progress in computer vision and natural language processing in generating synthetic samples mimicking real data distributions [16]. Here we leverage GAN in generating synthetic samples on previously learnt activities to re-train the evolvable classifier. This allows us to automate the training process without explicitly re-feeding the data to the classifier and not to store any historical data.

Even though GAN has achieved promising performance in computer vision [16], its effectiveness in sensor data is still under-explored. This papers aims to adapt and identify appropriate GAN architecture for HAR and mechanisms for ensuring the quality of generated samples for continual activity recognition. To do so, we have selected a collection of promising GAN architectures and experimented different strategies to tackle characteristics of sensor data. Sensor data exhibits the limitations of sensor noise, diversity of patterns within each activity class, and interference or overlapping of patterns between activity classes. All these characteristics make it challenging to converge a GAN model and thus difficult to generate high-quality, discriminative samples. To tackle this challenge, we have experimented different training strategies including gradient penalty and instance noise, and quality control strategies.

Another key novelty of our work is that we apply an evolvable neural network which can extend the output layer with new classes continuously and automatically, which does not require *a priori* knowledge of what and how many new classes are available in the future. To the best of our knowledge, this is the first time that GAN and an evolvable classifier have been applied to continual learning in HAR.

Our main contributions are listed as follows.

- *HAR-GAN* has been extensively evaluated on both binary and accelerometer sensor data, which are the most common sensor data in HAR. The evaluation has demonstrated the effectiveness of *HAR-GAN* in continual activity recognition and as well as uncovered findings in dealing with continual learning in different types of sensor data and activities.

- We have systematically integrated and evaluated with a range of state-of-the-art strategies on tackling catastrophic forgetting. The results have shown that knowledge distillation [21] has consistently improved learning across multiple datasets.
- We have extensively experimented with different GAN architectures and strategies. Our result has shown that the generated samples from GAN can replace real samples and achieve high accuracy in continual activity recognition.

2 RELATED WORK

This section will briefly review the recent work in sensor-based human activity recognition and in continual learning in the field of machine learning.

2.1 Sensor-based Activity Recognition

Activity recognition has been extensively studied in the last decade [8]. A large number of modern data-driven techniques, including Hidden Markov Models, Support Vector Machines, and more recently deep neural networks [49], have achieved promising results in extracting features from raw sensor data [19, 29] and learning complex correlations between sensor data and activities of interest [17]. For example, convolutional neural network and autoencoder have been used to extract spatio-temporal features on EEG signals [58] and an ensemble of deep neural networks has been designed for activity recognition [37]. Inspired by these recent successes, we also take a deep learning approach; that is, leveraging Generative Adversarial Networks (GAN) in continual activity recognition to enable learning the distribution of training data and automatically generating samples.

2.2 Continual Activity Recognition

Continual learning is a natural next step in HAR; that is, what if people start doing new activities, and if so, how a HAR model will be adapted to accommodate the change. Discovering and recognising emerging activities has been attracting increasing attention in recent years. Fang et al. have proposed a von Mises-Fisher-based hierarchical mixture model, which is a probabilistic model to represent an overall population as a mixture of a finite number of different components. Each component corresponds to a type of activity or a pattern of an activity [12]. Based on the Bayesian posterior probability, the model can assess whether a data point is an outlier or belongs to an existing class. Active learning is employed to acquire labels on new activities. Then a new activity class will be modeled as a new component and added to the mixture model. The component weights will be updated in an expectation maximisation (EM) process.

Gjoreski et al. have employed an agglomerative clustering technique to cluster streaming sensor data in real time [15]. If the incoming data does not fall into the existing clusters, they are considered as *anomaly*. To validate anomalous data for new activities, they apply two temporal constraints: (1) a human activity usually lasts for a certain period of time and (2) there should not be frequent transitions between activities. With these constraints, the technique can filter short outliers and be able to more accurately discover clusters for new activities.

Another types of techniques towards continual learning is to evolve activity models. Cheng et al. have adapted a zero-shot learner to recognise a new activity with limited training data [9]. A knowledge-driven model is maintained, which encodes the semantic relationship between high-level activities and low-level sensor attributes generated from accelerometer data. When including a new activity, domain experts and developers will need manually add new attributes and update the activity-attribute matrix with manually specified relationship between attributes and this new activity.

However, none of these techniques enable automatic evolution of an activity model without much re-engineering effort. To directly tackle this challenge, we propose *HAR-GAN* that can dynamically extend with new types of activities over time without the need to re-build or re-train the model from scratch.

2.3 Continual Machine Learning

Beside HAR, continual learning or lifelong learning is an emerging and important topic in the broader area of machine learning. Over recent years, many techniques have been proposed, especially at tackling catastrophic forgetting; that is, how to extend a learning model to accommodate new tasks while maintaining good performance on old tasks. Here we focus our review on a collection of techniques on neural network, as it is the one that has been adopted for our activity recognition model. The techniques can be categorized into three types: regularisation, dynamic architecture, and rehearsal [36].

Regularisation-based techniques in neural networks are inspired by the principle of continual learning in human brain; that is, after learning occurs, a process called *synaptic consolidation* reduces the plasticity of synapses that relate to previously learned tasks making them harder to change. Thus, it reduces the interference from newly received data. Similarly, the regularisation-based techniques control the way that the network updates its weight to balance between learning new tasks and retaining learned knowledge. The importance of weights is estimated in terms of their impact on previous tasks' performance. It prevents changes on the parameters important to the previous tasks and thus slows down changes to the corresponding parts of the network.

Elastic Weight Consolidation (EWC), a classic regularisation-based technique, uses the Bayesian approach to quantify the importance of parameters in terms of the posterior distribution [11]. The assumption is that there are many sets of parameters that lead to low errors for a certain task and there exists a common set of parameters that can produce a low error in all tasks. The goal is to constrain the network's parameters to stay in the low-error region and explore those sets of parameters to find the one that fits all tasks. Synaptic intelligence also uses the similar tactics in controlling the parameter update [57]. The main limitation of these approaches is that they highly depend on the relevance of the tasks. The more irrelevant the problems are, the more difficult it can learn. The hypothesized set of parameters might not exist so this leads to a bias towards maintaining existing knowledge rather than accepting new information.

Learning Without Forgetting (LwF) is not focused on regulating the network's weights, but rather on maintaining the predicted probabilities on all previously learned classes [30]. It penalizes the model when the new prediction distribution is different from previous prediction distributions. This method is drawn from knowledge distillation, a technique to transfer knowledge from one large model to the smaller one [21]. Both EWC and LwF have achieved promising results, so we have adopted and experimented them in *HAR-GAN* with the goal to assess which technique is amenable to sensor data.

Architectural techniques are to dynamically accommodate neural resources for upcoming tasks leaving more room to learn new data; for example, to increase the number of neurons, to update connectivity patterns or to add task-specific components to the network. Progressive network resolves the catastrophic forgetting by introducing a new neural network for each new task [41]. Then previous knowledge will be transferred to the new network through lateral connections. Because there is no interference to the learned networks, there is no catastrophic forgetting.

Context-dependent gating and synaptic stabilisation address the problem with another architectural approach, which is by allocating a different subnetwork for a different task [31]. During the training step, only a subset of nodes in the network will be assigned randomly to participate in each task. As inspired by activities in the human brain, sparse and mostly non-overlapping sets of

dendritic branches are active for any one task. This reduces the interference between tasks and also increases the stability of the learned knowledge by distributing informative data throughout the network. One major drawback of these architectural techniques is scalability. When the tasks accumulate to a certain number, the complexity of these architectures can be difficult to manage.

Differently, we adopt a flexible structure for our classifier so that we can extend the number of output classes on the fly, without the need to fix the number of classes nor to re-build the model from scratch.

Rehearsal techniques are another neuroscience-inspired solution. iCaRL [38] is a classic rehearsal technique, which stores only a small subset of data as 'exemplars' as the representative information of the old classes. These examples will be used to augment the classifier result using the nearest centroid. Generative replay model (GRM) is a more recent rehearsal technique without the need of storing any historic data. GRM uses generated data from a generative adversarial network (GAN) as replay samples [43]. In GRM, for each new task, a new classifier is built and trained with the real samples on the current task and the generated samples on the GANs, each of which is trained on previous tasks. At any point of time, the system maintains a collection of GANs per task and a classifier. There are several drawbacks with this design. Re-building a classifier every time can be computationally expensive. To deal with this problem, *HAR-GAN* extends the classifier every time by adding new classes on the output layer and fine tunes it with generated samples on previous classes and real samples on new classes. Also maintaining a collection of GAN models can be inefficient if there are a large number of classes to learn over time. *HAR-GAN* employs a conditional GAN [34], able to generate samples conditioned on each class. *HAR-GAN* is configured with various strategies to deal with convergence and stability of GAN. These two problems are already observed on MNIST—a simple hand-writing dataset and are getting more deteriorated with the intrinsic complexity in sensor data in terms of noise, diversity in patterns per class, and similarity between classes.

3 *HAR-GAN* CONTINUAL ACTIVITY RECOGNITION

Here *HAR-GAN* presents an iterative and incremental model to be able to extend with new activity classes while not sacrificing the performance on recognising old activity classes without the need of storing the historic data and retraining the model from scratch. Our key novelty in *HAR-GAN* resides in extending this model with different GAN architectures especially with conditional GAN to generate samples per activity type, and in evolving the classifier automatically over time while tackling the catastrophic forgetting challenge.

In the following, we will formally define the problem of continual learning in HAR, present the workflow, and then describe each component in the workflow and their design decisions. Following the definition in [43], we define the task sequence T of N tasks as $T = [t_1, t_2, t_3, ..., t_N]$. In HAR, each task is a session of work in which new activities emerge and they must be learned and integrated with the current system. Formally, a task t_i ($i \in [1, N]$) is coupled with a set of activity classes $C^i = \{c^i_1, ..., c^i_{K_i}\}$ and a collection of training data $\{(\mathbf{x}^i_j, y^i_j) | y^i_j \in C^i\}_{j=1}^{M^i}$. The learning objective on the ith task is to optimize a model to recognise the current activities in C^i and previous activities in $C^{i-1} \cup ... \cup C^1$, where $C^i \cap (C^{i-1} \cup ... \cup C^1) = \emptyset$; *i.e.*, new classes in the current task have not been observed in the previous tasks.

Figure 1 presents the workflow of *HAR-GAN*. The training session starts with the arrival of the first task t_1, which will build a GAN and a classifier. The classifier will learn to recognise activities in C^1 and the GAN learns the latent structure of $\{(\mathbf{x}^1_j)\}_{j=1}^{M^1}$. When the next task t_2 comes in, the GAN will generate samples for the previous activities in C^1. With the generated samples and the training data in t_2, we will update the classifier to recognise activities in both C^1 and C^2. The GAN

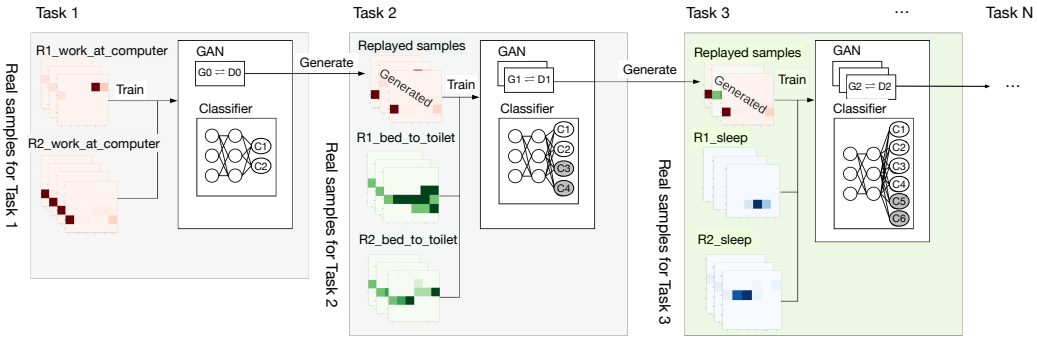


Fig. 1. The workflow of *HAR-GAN*. It starts with the first task of two activities R1_work_at_computer and R2_work_at_computer, which will be used to train a GAN and a classifier. When the next task of two activities R1_bed_to_toilet and R2_bed_to_toilet becomes available, then the GAN will generate samples on the classes in the first task, which will be combined with real samples on the second task to update the GAN and classifier. This process will be repeated with each new task.

will also be updated so as to be able to generate samples for the new activities. The process will be repeated whenever a new task arrives. In the following, we will illustrate how to design (1) a GAN with particular focus on addressing the convergence issue and improving the quality of generated samples, and (2) a classifier with particular focus on how to evolve the classifier to recognise more activities while not compromising the performance on old activities.

3.1 Design of GAN

GAN, Generative Adversarial Network, is a generative model to generate synthetic samples mimicking real data distributions [16]. It is composed of a generator and a discriminator in a minimax two-player game. A generator generates synthetic samples and a discriminator distinguishes whether the samples are synthetic from the generator or real from the original data. With the feedback from the discriminator, the generator will be updated to produce better samples. The learning completes when the discriminator collapses; i.e., failing to detect the generated samples are synthetic or real. In this way, a generator and a discriminator are competing with each other. This process then enforces the generated distribution to become closer to the true distribution or achieve *Nash Equilibrium*.

Formally, the generator is defined by $G(z)$. By giving a noise z, it then converts that noise vector into a synthetic sample. The discriminator is defined by $D(x)$. It estimates the probability to decide whether the input comes from the real data or the generator. To represent two objectives of the zero-sum game, the discriminator D will be trained to maximize the probability of assigning the correct label by maximising $\mathbb{E}_{x\sim p_r(x)}[logD(x)]$ when x is real and $\mathbb{E}_{x\sim p_g(x)}[1-logD(x)]$ when x is synthetic. Meanwhile, the generator G is trained to trick the discriminator; that is, it is expected to minimise the gap between a synthetic and real sample. Given a synthetic sample $G(z) : z \sim p_g(z)$, the generator's target is to minimise $\mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z)))]$. By combining both aspects, the loss function of this minimax game is defined below:

$$min_G max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[logD(x)] + \mathbb{E}_{z\sim p_g(z)}[log(1-D(G(z)))]. \quad (1)$$

There exist different architectures of GAN. Here we mainly consider single-class GAN and multi-class GAN. For single-class GAN, we train a GAN per class with the class' real data and then

stack them together to form a multi-GAN (mGAN) architecture. We also consider a conditional GAN (cGAN) on multiple classes which can generate samples with a given class label [33]. The loss term of cGAN is similar to Equation (1), and the difference is that the loss is conditioned on a class label:

$$min_G max_D L(D,G) = \mathbb{E}_{x\sim p_r}[logD(x|y)] + \mathbb{E}_{z\sim p_g}[log(1-D(G(z|y)))].$$

The motivation for considering mGAN and cGAN is that each of them has advantages and limitations. On the one hand, mGAN is assumed to generate samples with higher quality as it is only learned on one class at a time. However, as most HAR systems run on resource-constrained devices, stacking a large number of GAN models (e.g., for 20, 50 or 100 activity classes) might lead to the scalability problem. On the other hand, cGAN might suffer more on catastrophic forgetting and instability, as it will be trained with a combination of generated and real samples for multiple classes. As we will present in Section 5, sensor data exhibit high similarity between different activity classes, which could cause interference in consecutive task training. In *HAR-GAN*, we experiment both designs and assess their effectiveness on different types of sensor data.

It is well acknowledged that training GAN is non-trivial [18], [2], [32], as it can be slow or even fail to converge. When training a GAN, we often observe low dimensional supports and vanishing gradient [2], which can cause imbalance between its discriminator and generator and thus lead to instability during the training progress. As introduced earlier, the main challenge in HAR is sensor noise, diffusing boundary between activities and complexity in sensor features. On the one hand, this challenge can lead to poor accuracy in the discriminator, and thus the generator does not have accurate feedback, resulting in low accuracy. On the other hand, due to the high-dimensionality of sensor features, the generator cannot converge as fast as the discriminator. That is, even though a discriminator can achieve high accuracy quickly, the gradient of the loss function will decrease to zero, making the learning progress slow to converge. To tackle this problem, we will introduce the state-of-the-art approaches to alleviate the GAN's convergence issue: instance noise, Wasserstein loss function, and gradient penalty, and aim to find out which one is more suitable for sensor data.

3.1.1 Instance Noise. JS (Jensen-Shannon)- and KL (Kullback-Leibler)-divergence are typically used as a GAN's loss function. They become meaningless when the probability mass between actual data and synthetic data do not overlap, which leads to instability of GANs [3], [2]. One way to avoid this situation is to introduce instance noise to the inputs of the discriminator [3], [32]. The instance noise will smooth the actual and generated distributions so they become overlapped. In our current implementation, Gaussian noise is added into input vectors before feeding to the discriminator. The volume of instance noise will be damped down to zero by each epoch to balance the effect of the noise.

3.1.2 Wasserstein GAN. Another way to alleviate instability is to introduce a new GAN loss function called Wasserstein distance [4]. The Wasserstein distance measures the distance between two probability distributions by calculating the minimum energy cost of transforming one probability distribution to the other distribution. It could result in smoother and more meaningful distance than JS- and KL-divergence and thus will stabilise GAN's learning process. However, the original Wasserstein distance formula is intractable as it requires to exhaust all the possible joint distributions. To make it tractable, a transformation based on the Kantorovich-Rubinstein duality [48] is applied and the new loss function is now calculated by the difference of the average critic score on actual and generated data [4]; that is,

$$min_G max_D L(D,G) = \mathbb{E}_{x\sim p_r}[D(x)] - \mathbb{E}_{z\sim p_g}[D(G(z))].$$

3.1.3 Gradient Penalty. Non-convergence might still be observed in Wasserstein GAN due to weight clipping [18]. Weight clipping is a key component to maintain Lipschitz continuity, but it might end up with exploding or vanishing gradients, which again leads to instability of GAN. An alternative approach to avoid this is to add gradient penalty to the loss function [18]:

$$L_{GP}(D,G) = \mathbb{E}_{x\sim p_r}[D(x)] - \mathbb{E}_{z\sim p_g}[D(G(z))] + GP \tag{2}$$

$$GP = \lambda \mathbb{E}_{x'\sim p_{x'}}[(||\nabla_{x'} D(x')||_2 - 1)^2], \tag{3}$$

where x' is a random sample and λ is a controlling weight which is generally set to be 10. This equation is formulated to penalise the model when the gradient norm diverges from one [18].

3.2 Design of Evolving Classifier

A large number of recent research propose to apply deep neural networks (DNNs) to HAR [49]. The recent work has leveraged the learning capability of DNNs in capturing complex correlations between sensor data and activity classes. Build on top of this, we also design our activity classifier as a neural network, and focus on how to extend the capacity of the network over time in order to accommodate new activity classes. As described in Section 2, this capability has not been explored in HAR, and most of the activity models will require re-engineering or re-build the model to include new output classes.

Here we adopt a dynamically extensible activity classifier; that is, output units will be extended when a new task is introduced. To achieve this, when a new output class is introduced, the new weights and bias of the network will be initialised by a truncated normal distribution with standard derivation equals to $sqrt(2/l)$, where l is size of the last hidden layer. This method provides a controlled initialisation, resulting in faster, more efficient gradient descent and also minimal gradient oscillation during the training [20].

Furthermore, apart from the classifier's architecture, its loss function is customised to consider the loss on generated samples from the old classes and real samples from new classes. The loss function is defined as:

$$\begin{aligned} L(CL_t) = {} & r\mathbb{E}_{(x,y)\sim T_t}[L(CL_t(x), y)] \\ & + (1-r)\mathbb{E}_{x'\sim G_{t-1}}[L(CL_t(x'), CL_{t-1}(x'))] \end{aligned} \tag{4}$$

Given a generator G_t and a classifier CL_t at a train session t, the loss function will be calculated from actual loss and replayed loss regulating by a factor r representing how important of the new task. We set r to be 0.5, considering replay loss and actual loss equally. The actual loss is similar to simple classifier loss function by taking the actual input x and a label y and then applying cross entropy as function L to determine the classifier's ability. In contrast, the replayed loss is determined by loss from generated samples x' drawn from the generator(s) in the previous session.

The above designs allows to extend the classifier dynamically to retrain the knowledge of the network (i.e., weights and bias). However, when we train the classifier incrementally with training data on new activities, the classifier will be optimised to maximise the accuracy on the new activities, which can lead to decreased accuracy on the old activities. In the following, we will introduce Elastic Weight Consolidation (EWC) and Learning Without Forgetting (LwF) to control the gradient updates so as to avoid such decrease. Afterwards, we will describe the strategies to deal with the imbalance problem that is a classic problem in most of HAR datasets and to control the quality of generated samples from GAN.

3.2.1 Elastic Weight Consolidation (EWC). EWC is a regularisation technique that penalises the changes that are important to the previous tasks [11]. It adds an additional term to the loss function

as follows.

$$L(\theta) = L_t(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{t-1,i})^2,$$

where $L_t(\theta)$ is the loss for the current training session t, λ is a tuning parameter that indicates the importance of the previous tasks, F is the Fisher information matrix and θ^*_{t-1} are the trainable parameters from the previous tasks. A parameter's importance is estimated by the Fisher Information matrix. It will be evaluated after completing each training session using both generated samples on old tasks and real samples in the current task.

3.2.2 Learning Without Forgetting. Different from EWC, learning without forgetting (LwF) aims to maintain the probabilities predicted by the previous model with the help of knowledge distillation (KD). To do so, before training the tth task, all input will be fed to the most recent classifier (*i.e.*, the model trained at the training session $t-1$) to determine its probability vector y^*_{t-1}.

$$y^c_{t-1} = p_{\theta_{t-1}}(Y = c|x), y^{*c}_{t-1} = \frac{{y^c_{t-1}}^{1/T}}{\sum_{j=1}^{j=C}(y^j_{t-1})^{1/T}},$$

where y^c_{t-1} is the conditional probability distribution on a class label c given an input x, which is obtained from the network configured with the parameters θ_{t-1}. In order to increase the importance of the smaller logit values, before calculating loss, a probability vector y^c_{t-1} will be soften into y^{*c}_{t-1} by temperature T as follows (T=2) [21]. Given the training data $\{(\mathbf{x}^t_j, y^t_j)|y^t_j \in C^t\}_{j=1}^{M^t}$ at the session t, the new loss function is a combination of the loss function on new training data $L^N_{\theta_t}$ and the knowledge distillation loss $L^D_{\theta_{t-1}}$ on data in previous learned classes:

$$L(\theta) = \lambda^N L^N_{\theta_t} + \lambda^D L^D_{\theta_{t-1}} + R$$

$$L^N_{\theta_t} = -\sum_{j=1}^{M^t} \mathbf{y}_j log\hat{\mathbf{y}}_j$$

$$L^D_{\theta_{t-1}} = -\sum_{k=1}^{C^{1..t-1}} (\sum_{i=1}^{M_k} y^*_{i,t-1} log y^*_{i,t-1})$$

where $\mathbf{y}$ is the vectorised ground truth of y_j in the new training data, $\hat{\mathbf{y}}$ is the vectorised prediction, M is the number of generated samples on old activity classes, and R is the normal regularisation on the network, and $C^{1..t-1}$ is the number of classes that have been learned so far. λ^N and λ^D are the ratio to balance loss weight, which are set to be 1 and T^2, as they have achieved well balanced performance on old and new classes [46].

3.2.3 Oversampling. Class imbalance, where only a few activities occur often while most activities occur rather infrequently, is generally observed in HAR. Oversampling is implemented to handle the imbalanced problem. We use Synthetic Minority Over-sampling Technique (SMOTE) [7], which synthesises new data from the minority class.

3.2.4 Self-Verification of Generated Samples. The quality of replayed samples plays an important role in maintaining the accuracy of the previous tasks. The low-quality samples could behave as a noise which will distract and degrade the model's accuracy. A random process is embedded in a vital part of GANs so unexpected noise could be accidentally generated. Therefore, the quality control is needed but as it is impossible to manually verify these generated data and we cannot visually inspect the generated sensor data like the way that we do with the images. To ensure high standard quality, the previously trained classifier is used as a quality control. We assume that it is

trained to capture the key properties of those trained classes. On that account, before training for the current task, we will generate samples for each previous class and test them with the previous classifier. We verify their labels to see if the classifier can predict the same class label. If the samples pass the test, we will use them as replayed samples; otherwise, we will discard them. This process is computationally effective since generating new data takes less time than training. We hypothesise that this self-verification process can prevent accidental noise.

4 EVALUATION METHODOLOGY

In the following we introduce the datasets exploited in the evaluation, present the evaluation metrics and process on continual learning, and describe our implementation details and parameter selection.

4.1 Datasets

To assess the generality of our approach, we selected four state-of-the-art datasets with binary sensors and accelerometers, all of which have been widely used in the area of activity recognition. The selected datasets exhibit different forms of sensor representations and different levels of complexity such as diverse patterns in an activity class and high similarity between activity classes. The evaluation on these datasets will help build a comprehensive performance profile of *HAR-GAN*. As our focus is on continual learning, instead of feature extraction, we employ the basic feature extraction technique on binary sensor data and use the datasets that have already extracted features on accelerometer data. This will assess the generality of our approach independently of feature extraction techniques, and also make it easier to reproduce the results.

The first two datasets record sensor events from binary sensors. The sensors will be activated and produced a reading of '1' or 'ON', when a user is interacting with an object embedded with a sensor or is present in front of a sensor. The first dataset (denoted as *House A*) is collected by the University of Amsterdam [47]. It contains 14 sensors and 7 activities from a single-resident houses including 'sleep', 'shower', 'toilet', 'breakfast', 'dinner', 'drink', and 'leave home'. The second dataset is twor.2009 datasets from CASAS at the Washington State University [6] (denoted as *CASAS*). It records activities of daily living on two residents. This testbed consists of 72 sensors in 6 different types: motion sensors, door sensors, light switch sensors, water flow sensors, burner sensors and item sensors. For both binary sensor datasets, we follow the state-of-the-art techniques to segment sensor data into a 60-second interval and normalise the ratio of senor activation by the total number of events being activated in an interval [54]. We also add one temporal feature, which is the hour of the first sensor in each interval.

The other two datasets contain continuous sensor readings from inertial measurement units that are worn by participants, which are PAMAP2 [39] and DSADS [5]. As we focus on continual learning, we use the datasets that contain the extracted features on these datasets [50], which include mean, standard deviation, and correlations of axes.

To make the results comparable across different datasets, we fix the total number of activities as 10 on the three larger datasets, namely CASAS, PAMAP2 and DSADS, and use all the 6 activities on House A. Figure 2 presents the statistics of the selected datasets including the number of features, the number of activity classes, and the number instances, along with the class distribution of selected activities in our experiments. As we can see in Figure 2, these datasets exhibit imbalanced class distribution, which adds extra complication to continual learning. Intraclass diversity and interclass similarity are also observed in the datasets, as presented in Figure 3.

Dataset Statistics

Datasets	No. Features	No. Activities	No. Instances
House A	15	6	483
CASAS	72	10	15655
PAMAP	243	10	6451
DSADS	405	10	4800

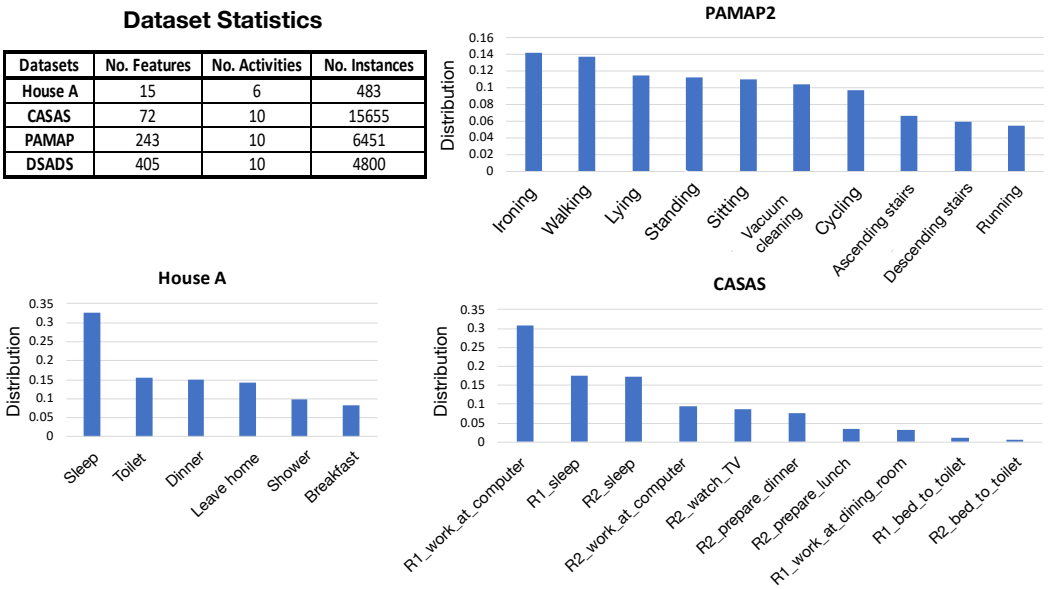


Fig. 2. Description of datasets, including the number of features, activities, and instances, and the class distributions. DSADS has the equal activity distribution.

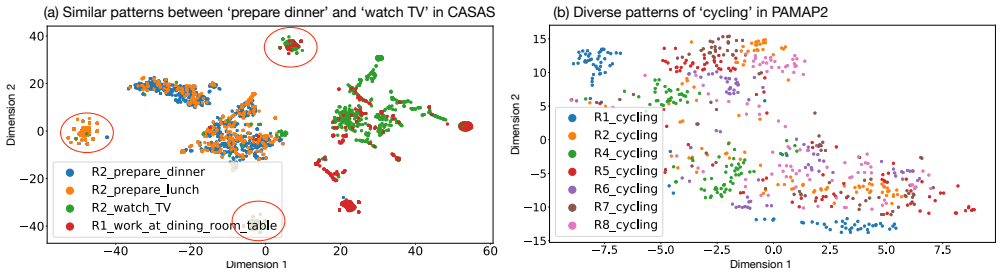


Fig. 3. t-SNE plots to demonstrate interclass similarity and intraclass diversity in sensor data. (a) 'prepare dinner' and 'watch TV' in CASAS exhibit high similarity as they have overlapping and diffusing boundaries of activity classes. (b) there exists diverse patterns on the 'cycling' activity in PAMAP2.

4.2 Evaluation Metrics and Process

To evaluate our model, we randomly generate task sequences and introduce tasks to the model one by one. Each task consists of 2 new activity classes. To train a new activity class, we use 80% of its data for training and 20% for testing. Every time when we train a new task, we will compute three types of accuracy. (1) Base accuracy – the accuracy of recognising the activity classes in the first task after training on each task, which will indicate the stability of the model; (2) *New* accuracy – the accuracy of recognising the new activity classes in the current task, which will indicate the plasticity of the model; and (3) *Overall* accuracy – the accuracy of recognising all the activity classes that have learned so far, which will indicate the overall performance of the model.

In addition, we consider the following three metrics from the machine learning community [27], which average and normalise the above three types of accuracy over time. Given N as a number of learned tasks so far, M_{base} measures how the model retains the knowledge by measuring the averaged ratio between the accuracy of the first task *(base accuracy)* to the expected accuracy that

is the accuracy of the same task from the offline model *(ideal accuracy)*. M_{new} indicates how well the model acquires the new knowledge. The test accuracy of a task that is immediately learned after each training *(new accuracy)* is used and divided by the ideal accuracy of the same task. Lastly, $M_{overall}$ presents the overall result of the model determined by the average ratio between the accuracy of each task after training the last task and its expected accuracy. These three metrics are relative to ideal accuracy on the offline model, which can reveal the effectiveness of continual learning, independent of specificity of learning tasks.

$$M_{base} = \frac{1}{N}\sum_{i=1}^{N}\frac{\textbf{base accuracy} \text{ after training task } i}{\textbf{ideal accuracy} \text{ of task } i},$$

$$M_{new} = \frac{1}{N}\sum_{i=1}^{N}\frac{\textbf{new accuracy} \text{ after training task } i}{\textbf{ideal accuracy} \text{ of task } i},$$

$$M_{overall} = \frac{1}{N}\sum_{i=1}^{N}\frac{\textbf{overall accuracy} \text{of task i at the last task}}{\textbf{ideal accuracy} \text{ of task } i}.$$

4.3 Implementation and Parameter Tuning

The classifier for activity recognition is implemented as a fully-connected neural network with two hidden layers, which is based on the recent studies on similar types of data [42], [56], [35]. We have also experimented on 3-layered neural network, the performance is not better than our 2-layered network. ReLU activation function is used in all hidden layers to avoid vanishing gradient. Adam optimiser is applied, which is a better options for problems with noisy data, sparse gradients and non-stationary objectives [28]. The experimental results from recent studies show that Adam optimiser could yield better results in both generative and classifier models [40], [18] and also in continual models [57], [24]. The learning rate and the batch sizes are set as the default setting (learning rate=0.01, batch sizes=5). We have run grid search on the number of hidden units per layer and based on the results, we select 200, 500, 1000 and 1000 neurons per layer for House A, CASAS, PAMAP2 and DSADS respectively.

We experiment four variant architectures of GANs, including mGAN (a stack of vanilla GANS [16]), cGAN (a conditional GAN on activity classes), mWGAN and cWGAN (with Wasserstain distance as a loss function). On all the GANs, we use the same architecture for discriminator and generator as the above classifier[1].

Another important parameter in *HAR-GAN* is the importance ratio between reply loss and actual loss from the current task (Equation (4)). To understand the effect of this parameter, we run various ratios on the datasets and examine their impact on the accuracy of continual learning. The results show that this ratio factor reflects the stability-plasticity dilemma as there is a trade-off between increasing model's stability and decreasing in model's plasticity. In the end, we decide to set the ratio as 0.5 to treat replay loss and actual loss equally which provides the maximum overall accuracy.

5 RESULTS AND DISCUSSION

This section will present the results on assessing the effectiveness of continual learning and discuss different design decisions of GANs and classifiers. We are seeking the answers to the following questions:

- **Q1** – Can *HAR-GAN* achieve effective continual activity recognition?
- **Q2** – To what extent the generated samples from GAN can replace real data samples?

[1]The implementation and results have been submitted as the supplementary materials.

- **Q3** – What strategies in Section 5.5 play a significant role in improving the accuracy of continual learning?

5.1 Effectiveness of Continual Learning

To assess the effectiveness of continual learning of *HAR-GAN*, we configured with cGAN and evolving classifier with the following components: self-verification, LwF-based regularisation, instant noise, and SMOTE for over-sampling. We follow the evaluation process in Section 4.2, where we extend and train a model every time when a new task is introduced. We compare our technique with five different settings:

- **Batch**: The model is trained with all activities at once. That is, we train a neural network with 80% of each dataset and test it with the remaining 20%. As there is no task sequence, the base, new, and overall accuracy is the same for each dataset.
- **Offline**: The model is trained all at once using data from all tasks so far. It considers being an ideal model for the experiment. All metrics are calculated using this model as the expected model.
- **None**: The model is sequentially trained one task at a time without any data from the previous tasks. It is treated as lower bound for the experiment.
- **Exact Replay**: The model is sequentially trained on a new task and all the training data from previously learned tasks. It represents a model with the best quality of replayed samples. Here we use all of the training data of the old classes for replay.
- **Generative Replay**: The model is sequentially trained using a new set of data and generated data from the generator, which is *HAR-GAN*.

Besides the above settings, we also select the following continual learning techniques for comparison.

- LwF [30] and EWC [11] which are regularisation only techniques. *HAR-GAN* has experimented the regularisation terms in LwF and EWC, and comparing with them will assess the advantage of using generated samples in continual learning.
- iCaRL (Incremental Classifier and Representation Learning) [38] that stores real samples from old tasks in memory and combines them with new training samples to update the classifier. However, the number of in-memory samples is often constrained, as it is impractical to store all the old samples. Here we compare *HAR-GAN* with iCaRL to assess the advantage of generating unbounded samples over storing constrained real samples.
- DGR (Deep Generative Replay) [45] that uses variational autoencoder (VAE) for generating samples. An extended technique is DGR-KD that combines DGR with knowledge distillation loss to further mitigate catastrophic forgetting. RtF (Replay-through-Feedback) [45] that improves upon DGR with generative feedback connections. The comparison with DGR, DGR-KD, and RtF[2] will demonstrate the strength of GAN over VAE in generating high-quality samples.

Figure 4 and 5 present the *base*, *new*, and *overall* accuracy on each dataset over time between *HAR-GAN*, basic settings, and state-of-the-art continual learning techniques. From these results, we can draw the following conclusions.

Firstly, as shown in Figure 4, *HAR-GAN* achieves high accuracy in continual activity recognition, as the overall accuracy is close to the offline accuracy across the four datasets. More specifically, compared to *offline*, the overall accuracy on *HAR-GAN* only drops 0.092, 0.187, 0.09, and 0.166 on the last task on House A, CASAS, PAMAP2, and DSADS respectively. Compared to *none* setting,

[2]We adapt the implementations from https://github.com/GMvandeVen/continual-learning.

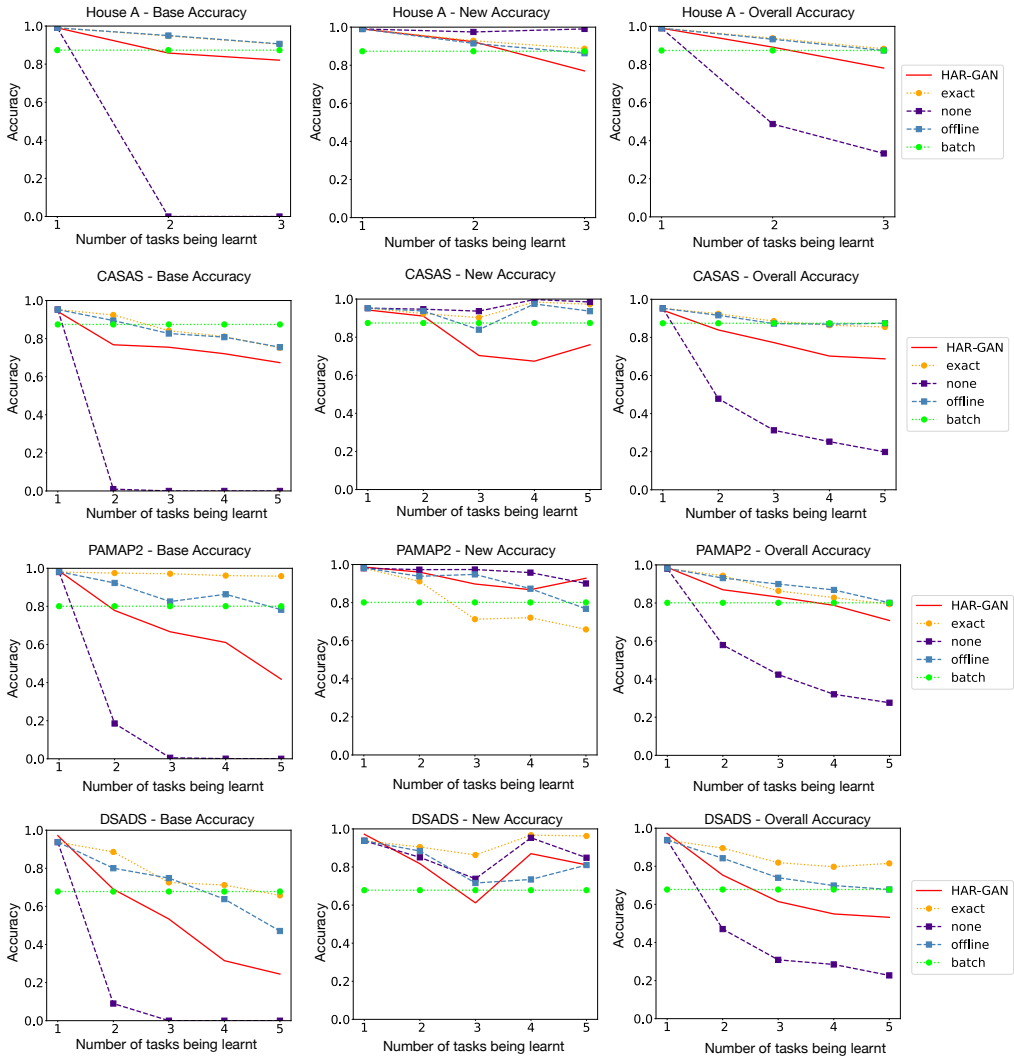


Fig. 4. Comparison of *Base*, *new*, and *overall* accuracy on activities learned over tasks on four different datasets between *HAR-GAN* and basic settings. The upper bound settings are presented in dashed lines with a square marker and the lower bound settings are in dotted lines with a circle marker.

the gain in the overall accuracy is 0.396, 0.496, 0.349 and 0.288 on these four datasets. Therefore, we conclude *HAR-GAN* performs effectively in learning activities in a continuous manner.

Secondly, *HAR-GAN* well balances retaining old knowledge while learning new activities as it achieves high accuracy on new activity while not dropping significantly on base activities on three datasets. We have observed the larger drop in the *base* accuracy on DSADS, which can be due to the poor quality of samples generated from GAN. The reason is that DSADS has the feature dimension of 405, which can be challenging to generate high-quality samples to resemble old classes.

Thirdly, as shown in Figure 5, GAN is a good replacement for real samples, as there is no significant difference in accuracy between *HAR-GAN* and *exact replay*, although the adoption of different GAN architectures may have different impacts as discussed in the following subsection. It

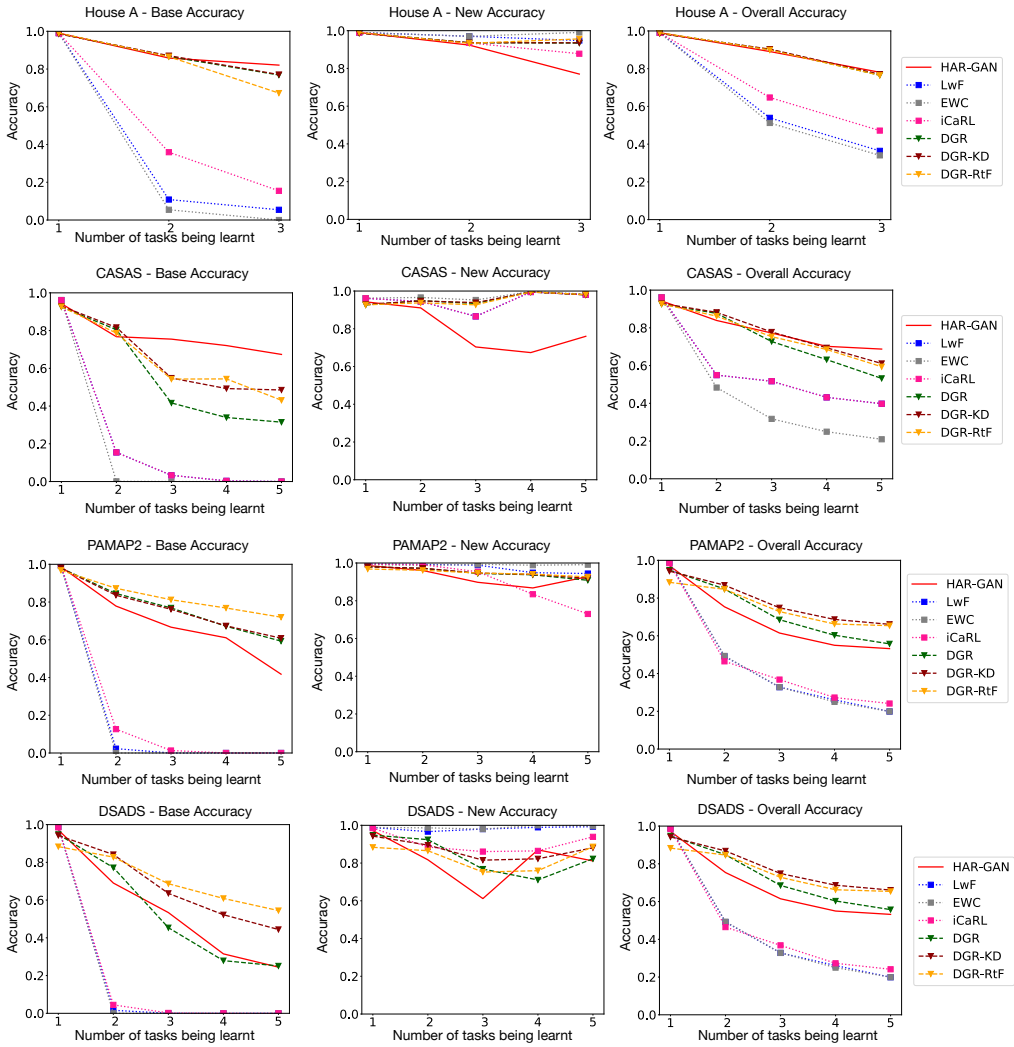


Fig. 5. Comparison of *Base*, *new*, and *overall* accuracy on activities learned over tasks on four different datasets between *HAR-GAN* and state-of-the-art continual learning techniques.The generative techniques are presented in dashed lines with a triangle marker and the other techniques are in dotted lines with a square marker.

significantly outperforms LwF and EWC in *base* and *overall* accuracy, which again demonstrates the strength of generative samples in mitigating catastrophic forgetting. Compared to iCaRL that stores a limited number of real samples from old classes, *HAR-GAN* produces higher accuracy on most of the datasets as it can generate as many samples as needed. However, on DSADS, iCaRL performs better than *HAR-GAN*still has the limitation of generating high-quality samples when the dimension is high.

Fourthly, compared to other generative techniques such as DGR, DGR-KD, and RtF, *HAR-GAN* has achieved similar accuracy to them on the four datasets. On House A and CASAS, *HAR-GAN* has a higher *overall* accuracy to the best of these three techniques by 2% and 10%. However, on PAMAP2 and DSADS, *HAR-GAN* is underperforming by 7% and 12%. This is likely to do with the fact that the

output layer of the classifier in DGR, DGR-KD, and RtF is fixed; that is, they do not need to extend the classifier every time when a new task is added, and thus they do not reshuffle the weights at the last layer. This makes learning more effective, especially on RtF that has a feedback training loop between VAE and classifier and the weights on the last layer are optimised from training the first task. However, this design does not suit the real world application, as they need to know the number of all the future classes in advance, which is very likely to be unrealistic. That is, it is often difficult, if not impossible at all, to predict what new activities a user might perform in the future. In contrast, *HAR-GAN* uses an evolvable classifier, which can automatically extend to any new number of classes without *a priori* knowledge on future activities.

Table 1. Experiments of different task sequence to demonstrate the challenge of continual activity recognition in the face of interclass similarity

Experiment	**Task**	**Activities** in each task	**Accuracy** of recognising 2 activities after training Task 1	recognising 2 activities after training Task 2	**Accuracy** of recognising 4 activities after training Task 1 and 2 sequentially
#1	Task 1	R2_prepare_dinner	0.985		0.015
		R2_watch_TV	0.988		0.16
	Task2	R2_prepare_lunch		1	0.96
		R1_work_at_dining_room_table		1	0.95
#2	Task 1	R2_prepare_dinner	0.41		0.525
		R2_prepare_lunch	0.65		0.29
	Task2	R2_watch_TV		0.872	0.592
		R1_work_at_dining_room_table		0.75	0.67

As shown in Figure 4, on DSADS, all the techniques exhibit a decrease on the 3rd task sequence. After looking into the results, we have found that the classes in Task 3 and 2 are similar, and the discriminability of the model on new classes in Task 3 is decreased so the accuracies drop. And classes in Task 4 are easier to learn, so the accuracy goes back. This phenomenon is caused by the impact of task sequence on catastrophic forgetting, which is already well known in continual learning [14] and is further complicated with high interclass similarity in HAR.

To have a closer look, we present a concrete example in Figure 3 (a) and we design two experiments with different task sequences, as shown in Table 1. We put dissimilar activities in the same task; that is, 'R2_prepare_dinner' and 'R2_watch_TV' in one task and 'R2_prepare_lunch' and 'R1_work_at_dining_room_table' in another. When we train each task separately, the accuracy on all these four classes is very high, between 98.5% and 100%. However, at the end of training these two tasks, when we test these four activities, the accuracy on the 2 activities in the first task (which are 'R2_prepare_dinner' and 'R2_watch_TV') is compromised; reducing to 1.5% and 16% respectively. The GAN models tend to generate samples to their similar counterpart that has been just trained, and the classifiers also forget about the previous learned activities, resulting in 95.5% accuracy on the latter two classes.

In another experiment, we put similar activities in the same task; that is, 'R2_prepare_dinner' and 'R2_prepare_lunch' in one task, and 'R2_watch_TV' and 'R1_work_at_dining_room_table' in another. When we train each task separately, the accuracy on all these four classes is not as high as the previous experiment, dropping to 49% and 85%. However, when we test at the end of training, *HAR-GAN* has achieved more balanced accuracy on these classes: between 29% and 67%. That is, the GAN and classifier suffer less catastrophic forgetting problem. However, in practice, it is difficult to avoid this situation. With more and more new tasks, the interference is more likely to impact. To tackle this problem, in the future, we will consider spinning out a new cGAN and classifier if new activities are too similar from the previous ones. It will be worthy of investigating

how to assess the similarity between activities and how to manage a growing number of cGANs and classifiers in a controlled and systematical manner.

5.2 Assessment of Different of GAN Architectures

Table 2. Comparison of M_{base}, M_{new}, and M_{all} on different GAN architectures with two baseline settings: *None* and *Exact Replay*. The bold text indicates the best performance among GANs.

Dataset	Architecture	M_{base}	M_{new}	$M_{overall}$
House A	None	0.36 ± 0.01	1.13 ± 0.01	0.34 ± 0.01
	Exact Replay	1.03 ± 0.01	1.07 ± 0.01	1.01 ± 0.01
	mGAN	0.95 ± 0.03	1.02 ± 0.02	0.84 ± 0.03
	mWGAN	**1.00 ± 0.02**	1.03 ± 0.02	0.83 ± 0.04
	cGAN	0.96 ± 0.02	1.02 ± 0.02	**0.89 ± 0.03**
	cWGAN	0.82 ± 0.04	**1.06 ± 0.02**	0.84 ± 0.03
CASAS	None	0.29 ± 0.05	1.20 ± 0.05	0.23 ± 0.02
	Exact Replay	1.23 ± 0.16	1.17 ± 0.05	1.04 ± 0.04
	mGAN	0.99 ± 0.16	1.07 ± 0.07	0.80 ± 0.04
	mWGAN	0.90 ± 0.17	1.12 ± 0.06	0.80 ± 0.04
	cGAN	**1.15 ± 0.25**	0.99 ± 0.07	**0.83 ± 0.06**
	cWGAN	0.88 ± 0.10	**1.13 ± 0.07**	0.76 ± 0.04
PAMAP2	None	0.27 ± 0.02	1.18 ± 0.03	0.32 ± 0.02
	Exact Replay	1.12 ± 0.05	0.98 ± 0.06	0.97 ± 0.06
	mGAN	0.61 ± 0.05	1.15 ± 0.03	0.68 ± 0.04
	mWGAN	0.30 ± 0.04	**1.17 ± 0.03**	0.42 ± 0.03
	cGAN	**0.79 ± 0.06**	1.14 ± 0.04	**0.85 ± 0.04**
	cWGAN	0.51 ± 0.06	1.16 ± 0.03	0.49 ± 0.04
DSADS	None	0.42 ± 0.06	1.40 ± 0.14	0.33 ± 0.05
	Exact Replay	1.97 ± 0.31	0.85 ± 0.11	0.83 ± 0.11
	mGAN	0.64 ± 0.08	1.33 ± 0.11	0.66 ± 0.06
	mWGAN	0.45 ± 0.06	**1.46 ± 0.13**	0.39 ± 0.06
	cGAN	**0.97 ± 0.23**	1.34 ± 0.11	**0.77 ± 0.09**
	cWGAN	0.57 ± 0.08	1.45 ± 0.14	0.42 ± 0.06

To assess the impact different GAN architectures on continual activity recognition, the accuracy M_{base}, M_{new}, and M_{all} on each dataset has been measured and reported in Table 2. By examining the results, we have the following observations. (1) cGAN works the best; *i.e.*, achieving the highest base and overall accuracy. mGAN is the second best, which suggests that cGANs conditional on activity classes might help guide the search for a target distribution throughout the search space, thus producing samples that have finer discriminative capability between classes. This finding is consistent with the comparison between normal GAN and conditional GAN in computer vision [34], [26]. (2) The Wasserstein loss function has little positive effect on the generators as both mWGAN (mGAN with Wasserstein loss) and cWGAN (cGAN with Wasserstein loss) have lower accuracy than their counterparts. (3) The small difference in accuracy between *exact replay* and various types of GANs implies that generated samples are good choices to replace real data. The overall performance $M_{overall}$ on all the GANs reach about 0.76 ~ 0.88 of the ideal accuracy, with the difference between cGANs and the *exact replay* setting as 0.12, 0.21, 0.12, and 0.07 on House A, CASAS, PAMAP2, and DSADS.

One important aspect when assessing the quality of GAN generated samples is *mode collapsing*; that is, the generated data are collapsed to a few mode centers and are not diverse enough to capture the whole space of real data. This can be detrimental to HAR, as there often exist multiple patterns for one activity class (See an example in Figure 3). To examine the diversity aspect, we have plotted

the generated samples from cGAN, mGAN, cWGAN, and mWGAN on the 'Vacuum Cleaning' activity on PAMAP2 dataset in Figure 6. None of the GANs can cover the whole distribution of real data. cGAN and mGAN has a different coverage because cGAN is conditioned on a class label, which might focus more on the data that have better discriminative features. In contrast, cWGAN and mWGAN covers much smaller space than cGAN and mGAN, indicating that Wasserstein may constrain the diversity of samples being generated. In the following section, we thoroughly assess the quality of generated samples across a wide range of the state-of-the-art metrics.

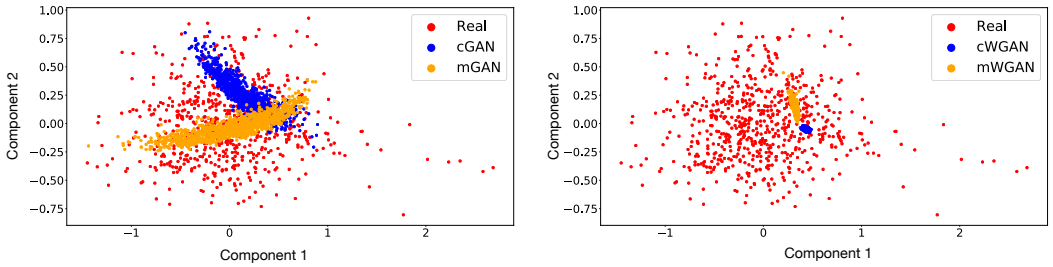


Fig. 6. PCA (Principal component analysis) plots to compare distribution of generated samples compared to real samples on the 'Vacuum Cleaning' activity from PAMAP2

5.3 Quality of GAN Samples

To gain a better understanding of GAN, we will assess the quality of generated samples and how the quality has a significant impact on tackling catastrophic forgetting. It is difficult to quantify the quality of the generative model. Because the majority of research on GAN works with images, the evaluation is dominated by visual inspection and qualitative description, which can be subjective and possibly misleading. In recent years, measurement of dissimilarity between real and generated probability distribution; *e.g.,* KL- and JS-divergence, has become a widely accepted metric in the literature. But both divergence metrics assume that the input distributions need to be known and well-defined [51], which might not be suitable for many HAR applications where the distributions of the sensor data can change over time due to the change in sensor performance and users' daily routine. Therefore, these divergence metrics alone cannot be used as quality measurement for GAN samples.

Following the literature [51], we consider three metrics for sample quality evaluation: Inception score, Kernel MMD (Maximum Mean Discrepancy), and 1-Nearest Neighbor score. Inception score measures certain properties of generated samples with the Inception network, a pre-trained model on the ImageNet [44]. The inception score is defined by the average KL divergence between the probability distribution of a certain label y recognised by the Inception model given an input x – $p(y|x) : x \sim \mathbb{P}_g$ and the marginal distribution obtained from all the samples – $p(y)$. The score is computed by the following equation.

$$IS(\mathbb{P}_g) = e^{\mathbb{E}_{x\sim\mathbb{P}_g}[KL(p(y|x)||p(y))]}, \tag{5}$$

where $p(y|x)$ is the posterior probability on output class given an input x. The higher $p(y|x)$, indicating that the generated samples can be recognised by the offline model, the higher quality of the generated samples. The term $p(y)$ indicates the variety of the output; that is, a higher $p(y)$ suggests an even distribution of classes. We replace the Inception network with our batch offline classifier that is trained with real data on all the activities. As a complementary to IS, we also consider test accuracy from the offline model (Acc_{off}) to indicate the quality of the sample being generated.

Kernel MMD represents the dissimilarity as the distance between samples drawn independently from two distributions $\mathbb{P}_g$ and $\mathbb{P}_r$ for some kernel function k, which is set as Gaussian kernel in our experiment. A lower MMD score suggests a closer distribution between $\mathbb{P}_g$ and $\mathbb{P}_r$. It is computed by the following equation:

$$MMD^2(\mathbb{P}_g, \mathbb{P}_r) = \mathbb{E}_{\mathbf{x}_r, \mathbf{x}'_r \sim \mathbb{P}_r, \mathbf{x}_g, \mathbf{x}'_g \sim \mathbb{P}_g}[k(\mathbf{x}_r, \mathbf{x}'_r)] + k(\mathbf{x}_g, \mathbf{x}'_g)] - 2k(\mathbf{x}_r, \mathbf{x}_g)]. \quad (6)$$

1-Nearest Neighbor (1-NN) Score is computed by the leave-one-out accuracy of a 1-nearest neighbor classifier trained. The model will be trained with positive samples from true distribution $\mathbb{P}_g$ and negative samples from generated distribution $\mathbb{P}_r$. It asserts that the classifier should yield about 0.5 accuracy when GANs achieve the high quality results because the average distance at any given point to both distributions are about the same when two distributions are close and overlapped. It can also indicate the cases that $\mathbb{P}_g$ is overfitting to $\mathbb{P}_r$ (in the extreme, GANs can only re-generate the exact input) by observing when the score is lower than 0.5 and it will go zero when $\mathbb{P}_g = \mathbb{P}_r$.

TPR (true positive rate) and TNR (true negative rate) are used to indicate when mode collapse occurs [51]. A low TPR implies that the majority of real samples are surrounded by generated samples indicating that the generator can capture different modes of real data. In contrast, a high TNR implies that the generated samples are surrounded by samples from the same class with a few mode centers, indicating that mode collapse occurs.

Table 3 reports the results of the above four metrics on different GAN architectures. cGAN and mGAN produce comparable quality of generated samples, as they achieve similar accuracy on the offline model. Compared to cGAN, mGAN achieves higher Acc_{off}: 0.17 on House A and 0.02 on DSADS, but lower Acc_{off}: 12.6 on CASAS and 0.07 on PAMAP2. Also, cGAN and mGAN gain similar IS scores: 6.932 vs. 8.325 on CASAS and 7.684 vs. 8.592 on PAMAP2. cGAN have higher IS scores than mGAN: 4.35 vs. 3.671 on House A and 6.301 vs. 3.787 on DSADS.

Table 3. Sample quality assessment of GAN architectures

Dataset	Architecture	IS	Acc_{off}	Kernel MMD	1-NN		
					Acc	TPR	TNR
House A	mGAN	3.671	**1.000**	0.171	**0.934**	**0.888**	**0.956**
	mWGAN	3.017	0.830	**0.158**	0.966	0.915	0.991
	cGAN	**4.350**	0.833	0.180	0.957	0.903	0.983
	cWGAN	4.098	**1.000**	0.222	0.982	0.944	1.000
CASAS	mGAN	**8.325**	0.791	0.250	0.910	0.891	0.930
	mWGAN	6.309	0.796	**0.201**	**0.902**	**0.884**	**0.921**
	cGAN	6.932	0.917	0.204	0.947	0.893	1.000
	cWGAN	7.811	**0.937**	0.215	0.973	0.945	1.000
PAMAP	mGAN	**8.592**	0.894	**0.156**	**0.927**	**0.854**	1.000
	mWGAN	4.369	0.438	0.593	0.999	0.999	**0.998**
	cGAN	7.684	**0.901**	0.202	0.962	0.925	0.999
	cWGAN	5.362	0.501	0.384	1.000	0.999	1.000
DSADS	mGAN	3.787	**0.617**	0.656	1.000	1.000	1.000
	mWGAN	3.685	0.299	0.796	1.000	1.000	1.000
	cGAN	**6.301**	0.597	**0.347**	**0.988**	**0.976**	1.000
	cWGAN	1.109	0.093	0.895	1.000	1.000	1.000

Consistent with the results in Section **??**, the Wasserstein loss function has weak positive effect on the generators. mGAN and cGAN have achieved smallest the kernel-MMD scores, which indicates that their generated samples are close to the real samples. Compared to cGAN, cWGAN achieves better Acc_{off} with the improvement of 0.17 on House A and 0.12 on CASAS, but much worse Acc_{off} with a drop of 0.4 on PAMAP2 and 0.5 on DSADS. mGAN achieves consistently higher

Acc_{off} than mWGAN, with an improvement of 0.17 on House A, 0.22 on CASAS, 0.46 on PAMAP2, and 0.32 on DSADS. Although Wasserstein loss significantly stabilises GAN's training process, it does not guarantee to get a proper data coverage.

Furthermore, the 1-NN score clearly shows that mode collapse occurs in all models. It indicates that the generators can only produce a certain type of given data. This could reduce the generality of the classifier especially in a situation that has a high variety of activity patterns. However, the results would seem to suggest that the mode collapse does not cause a significant accuracy drop. It is likely that the GANs could capture enough properties that are sufficiently used to solve the tasks.

5.4 Convergence, Training Time, and Memory Size

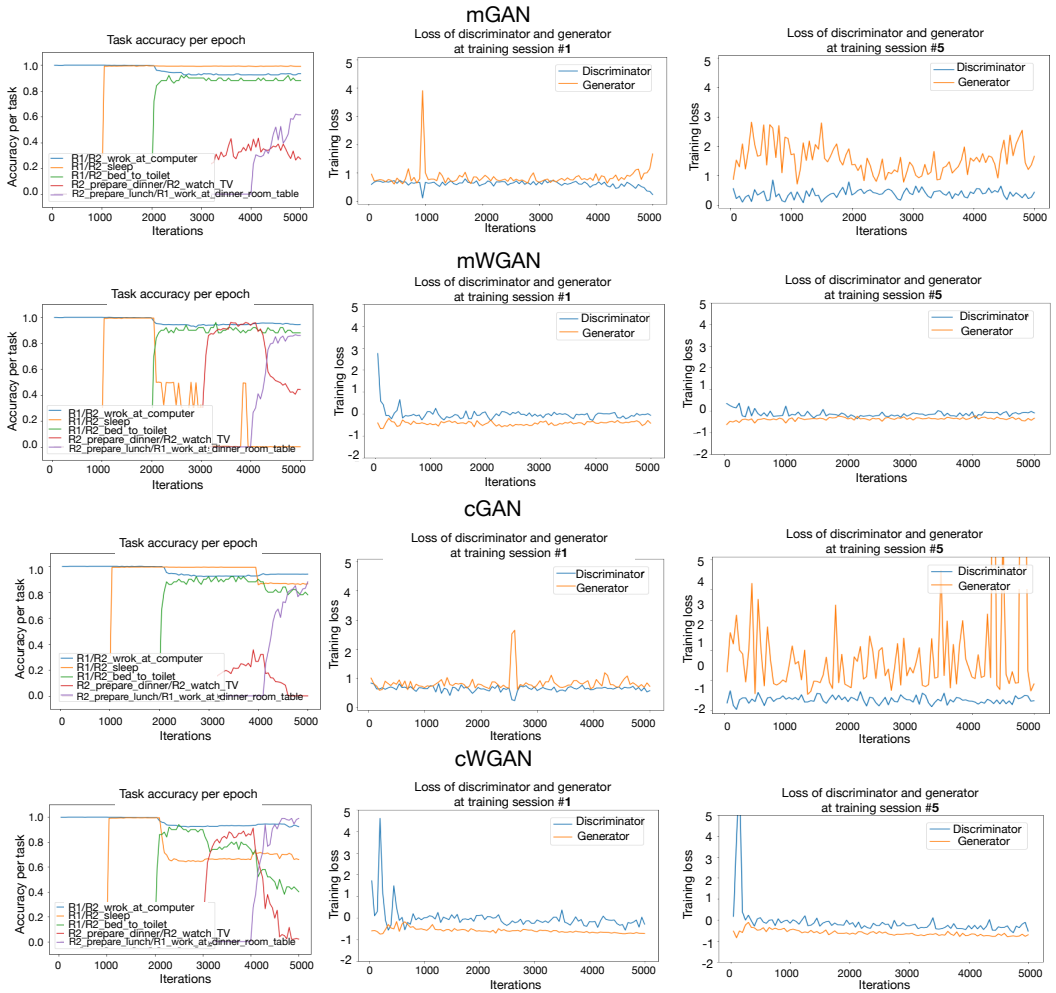


Fig. 7. Comparison of continual activity recognition accuracy and the loss on generator and discriminator on each GAN architecture over time on the CASAS dataset

Instability might be a major drawback of using GANs in the continual learning model. Figure 7 shows that the divergence between discriminator and generator could be observed in several classes on the CASAS dataset. In extreme, the generator might lose all information about the given classes

and produce only noisy and unrelated samples. This will degrade the classifier accuracy on that task or even interfere with all other classes.

Considering the complexity of the input space, both cGAN and cWGAN are likely to be more susceptible to the inference and instability problem. While better stability could also be observed in the WGAN as Wasserstein distance gives meaningful indicator even when the probability distribution of real and synthetic data are not overlapping [4]. Surprisingly, the stability issue does not lead to a major decrease in the performance of the framework. The unstable GANs could still have moderately good accuracy or even result in a better model. It is still unclear how instability could harm the model but arguably, it certainly has a gap between exact replay and generative replay.

The training time is assessed on the same moderate machine: a DELL Inspiron 5370 with a processor i5-85250U CPU @ 1.60GHz, 4 cores and 12G memory. On the larger datasets PAMAP2 and DSADS mGAN and cGAN take about 200 $\sim$ 300 seconds to learn a new task, and mWGAN and cWGAN take longer, which is about 300 $\sim$ 425 seconds. The averaged training time per task on different GANs are presented in Table 4.

Table 4. Training time (per task) and model size of GANs

Dataset	Architecture	Training Times (in seconds)	Model size (in MB)
HouseA	mGAN	218.221 ± 4.457	6.25
	mWGAN	420.430 ± 6.197	6.25
	cGAN	230.270 ± 41.246	1.05
	cWGAN	417.628 ± 84.313	1.05
CASAS	mGAN	113.453 ± 0.959	66.84
	mWGAN	221.088 ± 1.594	66.84
	cGAN	152.586 ± 18.197	6.71
	cWGAN	254.707 ± 26.133	6.71
PAMAP	mGAN	199.390 ± 1.438	287.44
	mWGAN	364.938 ± 4.492	287.44
	cGAN	269.627 ± 28.846	28.79
	cWGAN	397.254 ± 42.922	28.79
DSADS	mGAN	234.003 ± 9.589	324.54
	mWGAN	424.940 ± 11.527	324.54
	cGAN	313.566 ± 33.527	32.50
	cWGAN	394.307 ± 60.784	32.50

The model size of mGANs is between 6.25 MB (on House A) and 324.54 MB (on DSADS), which has linearly increased with the number of tasks. This is a major issue to the mGAN architecture. When there is a large number of tasks, there might take up much memory space. On the contrary, cGAN maintains the same size over time (the smallest size 1.05 MB on House A and the largest size 32.5 MB on DSADS), which can be preferable for resource-constrained devices.

5.5 Design Decisions on Classifiers

We have equipped the classifier in *HAR-GAN* with different components: self-verification, SMOTE, and classifier regularisation (both EWC and LwF). In this section, we evaluate the importance of each component by ablation analysis. Specifically, we fix our GAN architecture as cGAN, and then add only one component a time and compare their accuracy.

Figure 8 presents M_{base}, M_{new}, and $M_{overall}$ of the ablation analysis on four datasets. From the results, we draw the following conclusions. (1) The most significant component is the knowledge distillation loss from LwF, which consistently outperforms the other components. (2) SMOTE does not have a significant impact on the performance. It helps improve accuracy on House A

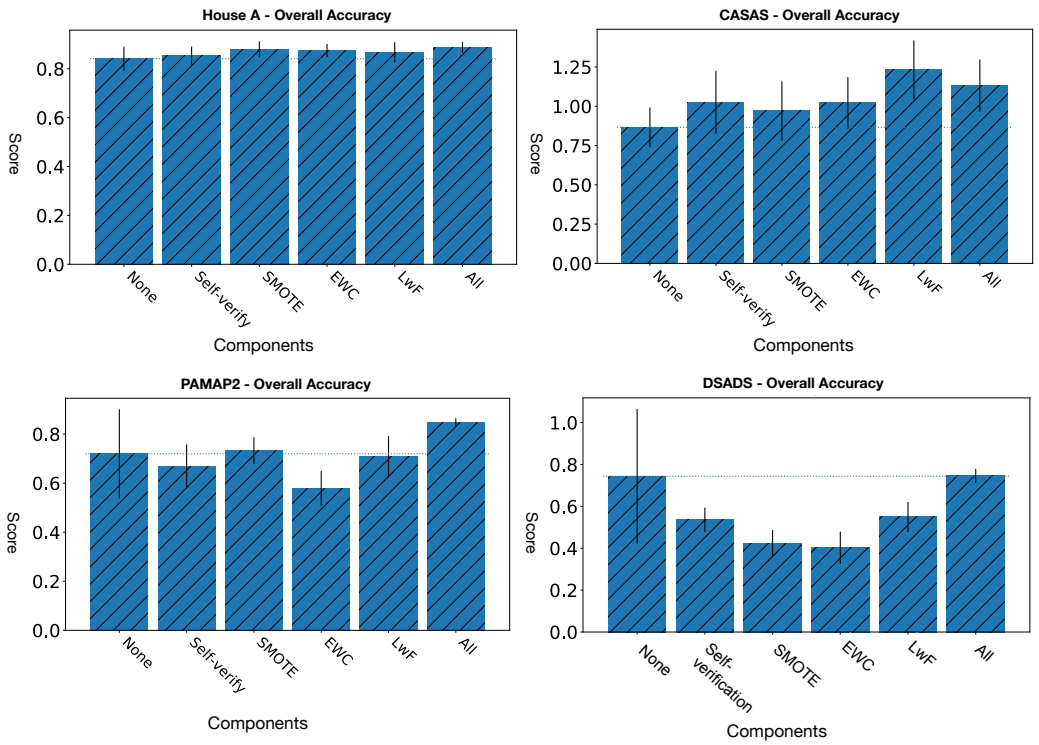


Fig. 8. Ablation analysis of different components

and PAMAP2, but it leads to a large dip in accuracy on DSADS, which is still due to the curse of dimensionality. On CASAS, classes have high similarity, which makes it difficult for SMOTE to generate good-quality samples exactly within the class boundary. (3) The impact of self-verification is negatively correlated with the feature dimension. Self-verification helps improve the overall accuracy on House A and CASAS, but not on high-dimensional feature space like PAMAP2 and DSADS. The reason is that when cGANs fail to generate high-quality samples to pass self-verification, we have less training data on the previous classes to allow the classifier to retain the old knowledge. Our current implementation does not compensate poor samples by repetitively generating new samples. The reason is that we have noticed that when GAN itself is not capable, it can take several rounds to generate the required number of samples. In the future, we will look into how to further improve the quality of samples on a high-dimension feature space and also to design more flexible self-verification strategy to include more generated samples.

6 CONCLUSION

This paper proposes *HAR-GAN* to support continual learning in human activity recognition with generative rehearsal. Continual learning refers to the ability to learn new activity classes over time. This requires to dynamically extend the original model with new output classes, and update the model parameters to learn new classes' distributions, while not compromising the recognition performance on old activity classes.

With extensive experiments on four datasets from both binary sensors and accelerometers, *HAR-GAN* has demonstrated great potential in supporting continual activity recognition. It can be

combined with anomaly detection techniques in Section 2 so as to enable an automatically evolving HAR system. Anomaly detection helps identify new activities, which can be fed to *HAR-GAN* to update the activity model. In this way, the HAR system can be continuously updated and recognise new activities without much need for re-engineering effort; such as re-training and re-deploying the model every time when new activities become available.

As the accuracy for continual learning on HAR still has a room for improvement, in the future, we will look into new techniques to generate high-quality of samples that best represent the real data while also encouraging the diversity. Also we are interested in designing a hierarchy of conditional GANs to mitigate catastrophic forgetting effect and also reduce the interference between similar activity classes. To do so, we will look into more sophisticated regularisation terms such as contrastive loss [10] and anchor-based margin ranking loss [23]. We will also investigate a more advanced GAN architecture that combines autoencoder to mitigate its forgetting [25] or a robust GAN model with Gumbel softmax in its discriminator [52].

REFERENCES

[1] Z. S. Abdallah, M. M. Gaber, B. Srinivasan, and S. Krishnaswamy. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems*, pages 1–21, 2016.

[2] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR 2017*, 2017.

[3] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks, Jan. 2017.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML 2017*, pages 214–223, 2017.

[5] B. Barshan and M. C. Yüksek. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, pages 1649–1667, 2014.

[6] CASAS. WSU CASAS smart home project. http://casas.wsu.edu/datasets/, Aug. 2014. Accessed: 2019-5-30.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu. Sensor-based activity recognition. *IEEE TSMC Part C*, pages 790–808, 2012.

[9] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceedings of MobiSys '13*, pages 361–374. ACM, 2013.

[10] S. Dai, Y. Cheng, Y. Zhang, Z. Gan, J. Liu, and L. Carin. Contrastively smoothed class alignment for unsupervised domain adaptation. *ArXiv*, abs/1909.05288, 2019.

[11] J. K. et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[12] L. Fang, J. Ye, and S. Dobson. Discovery and recognition of emerging human activities using a hierarchical mixture of directional statistical models. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[13] L. Fiorini, F. Cavallo, P. Dario, A. Eavis, and P. Caleb-Solly. Unsupervised machine learning for developing personalised behaviour models using activity data. *Sensors*, 17, 2017.

[14] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, pages 128–135, 1999.

[15] H. Gjoreski and D. Roggen. Unsupervised online activity discovery using temporal behaviour assumption. In *Proceedings of ISWC '17*, pages 42–49, 2017.

[16] I. Goodfellow et al. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Y. Guan and T. Plötz. Ensembles of deep lstm learners for activity recognition using wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2), June 2017.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

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

[20] K. He et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV 2015*, pages 1026–1034, 2015.

[21] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] H. S. Hossain, N. Roy, and M. A. A. H. Khan. Active learning enabled activity recognition. In *Proceedings of PerCom '16*, pages 1–9, 2016.

[23] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR 2019*, 2019.

[24] W. Hu. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR 2019*, 2019.

[25] W. Hu, Z. Lin, B. Liu, C. Tao, Z. Tao, J. Ma, D. Zhao, and R. Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *ICLR*, 2019.

[26] P. Isola and et al. Image-to-image translation with conditional adversarial networks. In *CVPR 2017*, pages 1125–1134, 2017.

[27] R. Kemker and et al. Measuring catastrophic forgetting in neural networks. In *AAAI 2018*, 2018.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, Third 2013.

[30] Z. Li and D. Hoiem. Learning without forgetting. *IEEE TPAMI*, 40(12):2935–2947, 2017.

[31] N. Y. Masse, G. D. Grant, and D. J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

[32] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

[33] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[34] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *JMLR*, pages 2642–2651, 2017.

[35] F. J. Ordonez and D. Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016.

[36] G. I. Parisi et al. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[37] V. Radu, C. Tong, S. Bhattacharya, N. Lane, C. Mascolo, M. Marina, and F. Kawsar. Multimodal deep learning for activity and context recognition. In *Proceedings of Ubicomp '18*, 2018.

[38] S.-A. Rebuffi and et al. icarl: Incremental classifier and representation learning. In *CVPR 2017*, pages 2001–2010, 2017.

[39] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *ISWC 2012*, pages 108–109, 2012.

[40] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[41] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[42] S. S. Saha et al. Position independent activity recognition using shallow neural architecture and empirical modeling. In *UbiComp/ISWC '19 Adjunct*, pages 808–813, 2019.

[43] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[45] G. M. van de Ven and A. Tolias. Generative replay with feedback connections as a general strategy for continual learning. *ArXiv*, abs/1809.10635, 2018.

[46] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[47] T. L. van Kasteren, G. Englebienne, and B. J. Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity recognition in pervasive intelligent environments*, pages 165–186. Springer, 2011.

[48] C. Villani. *Optimal Transport: Old and New (Grundlehren Der Mathematischen Wissenschaften)*. Springer, 2008.

[49] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3 – 11, 2019. Deep Learning for Pattern Recognition.

[50] J. Wang and et al. Stratified transfer learning for cross-domain activity recognition. In *PerCom 2018*, pages 1–10. IEEE, 2018.

[51] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

[52] S. Yao, Y. Zhao, H. Shao, C. Zhang, A. Zhang, S. Hu, D. Liu, S. Liu, L. Su, and T. Abdelzaher. Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3), 2018.

[53] J. Ye, S. Dobson, and F. Zambonelli. Lifelong learning in sensor-based human activity recognition. *IEEE Pervasive Computing*, 18(3):49–58, 2019.

[54] J. Ye, S. Dobson, and F. Zambonelli. Xlearn: learning activity labels across heterogeneous datasets. *ACM TIST*, 2020.

[55] K. Yordanova. Challenges providing ground truth for pervasive healthcare systems. *IEEE Pervasive Computing*, pages 100–104, 2019.
[56] M. Zeng et al. Convolutional neural networks for human activity recognition using mobile sensors. In *MobiCASE 2014*, pages 197–205, 2014.
[57] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *JMLR*, pages 3987–3995, 2017.
[58] X. Zhang and et al. Converting your thoughts to texts: Enabling brain typing via deep feature learning of eeg signals. In *PerCom 2018*, pages 1–10, 2018.

Received February 2020