Continual Learning in Sensor-based Human Activity Recognition: an Empirical Benchmark Analysis

Saurav Jha[a], Martin Schiemer[a], Franco Zambonelli[b], Juan Ye[a]

[a] *School of Computer Science, University of St Andrews, UK*
[b] *Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia*

Abstract

Sensor-based human activity recognition (HAR), with the ability of discovering human daily activity patterns from wearable or embedded sensors, is a key enabler for many real-world applications in smart home, personal healthcare, and urban planning. As we are witnessing an increasing number of activity-aware applications are deployed in real-world environments, emerges an important question: how can a HAR system autonomously learn new activities over a long period of time without being re-engineered from scratch? This is referred to as *continual learning*, which has been particularly popular in computer vision. With a high volume of techniques being developed, this paper aims to assess to what extent the existing continual learning techniques can be applied to HAR. We design a general framework to evaluate their performance on various types of commonly used datasets, perform comprehensive empirical analysis on their computational cost and effectiveness of tackling HAR challenges such as sensor noise and lack of sufficient labels. The results have uncovered insights and future research directions for HAR systems.

Keywords: Human activity recognition, continual learning, lifelong learning, incremental learning

1. Introduction

Sensor-based human activity recognition (HAR) is to infer human daily activities from data collected on various types of sensors [46]. It can be regarded as a *classification* problem; that is, given raw sensor data, extracting features and classifying them into a class label; *i.e.*, an activity. For example, by monitoring users' interaction with everyday objects via ambient binary sensors we can infer whether the user is watching TV or preparing a meal. By tracking users' movement via accelerometers or gyroscopes on wearables we can recognise their physical activities such as jogging or climbing stairs. These daily activities can have a significant impact on a wide range of

real-world applications, ranging from smart home [46] and adaptive environments [41] to personal healthcare [31] and disease diagnosis.

We have witnessed an increasing number of HAR applications being deployed and running over longer spans [46]. This drives an important research question: *how does a HAR system continuously discover and learn new types of activities*? We cannot assume that once a HAR system is trained with an initial set of activities, then the users of the system will only perform the same set of activities all the time. People tend to change their behaviour patterns for internal or external factors. Such change requires a HAR system to adapt their learning and include new types of activities in order to provide desired services. For example, the COVID outbreak has impacted the routine of people across the world, including practising different exercises, cooking new cuisines, and switching work patterns. For a personal healthcare monitoring application, failure to recognise new exercise routines may lead to misdiagnosis and inappropriate medication prescription. Therefore, continual learning is the key enabler for a long-term, sustainable HAR system.

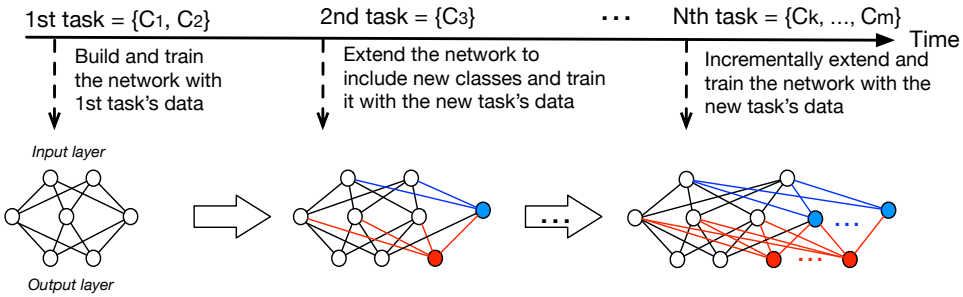


Figure 1: An example of task-incremental continual learning with neural network

Continual learning, or lifelong learning, is evoking increasing attention in the field of machine learning, aiming to retaining old knowledge and accumulating new knowledge by continually learning new tasks over time [25]. In HAR, we are concerned with *task-incremental* continual learning, where a system learns one task at a time and each task contains training data on a new set of classes. Take a neural network as an example in Figure 1, where for each new task, the network needs to be extended to include new classes and/or add more parameters, and then be trained with the new task's data. Faced with this continual learning setting is often the *catastrophic forgetting* (CF) challenge; that is, the network will be optimised towards the new task's data while forgetting the old knowledge and thus degrade the performance on inferring the old classes. As presented in

the example of Figure 2, if no CF mitigation method is employed, the network can only achieve high accuracy on the current classes after training on each new task, while obtaining low accuracy on all the old classes.

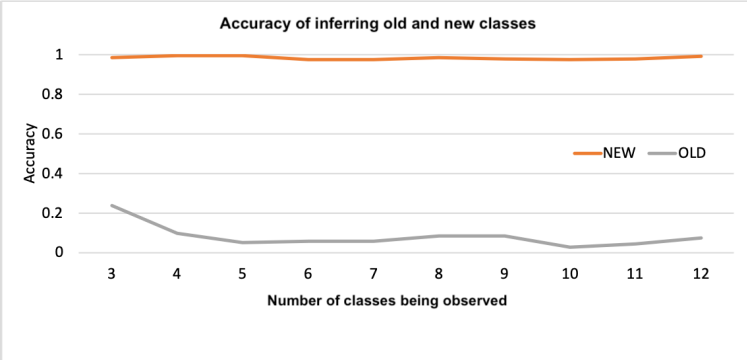


Figure 2: An example of catastrophic forgetting effect [47] where the accuracy on new classes is high and the accuracy on old classes remains low as the network keeps forgetting the old knowledge

A large number of continual learning techniques with neural networks have devoted to tackling the CF challenge [25, 32]: (1) regularisation – constraining the network parameter updates such that old knowledge is retained; (2) rehearsal – storing a small number of old classes' data to replay with new classes' data when training a network; and (3) dynamic architectures – extending the network architecture for new knowledge while keeping the parameters important for the old classes.

The purpose of this paper is to benchmark the state-of-the-art continual learning techniques on sensor-based human activity recognition datasets via extensive empirical evaluation[1]. We exclusively focus on neural network based techniques as they have been widely adopted in HAR tasks. Continual learning techniques have achieved promising performance in computer vision [25] and robotics [26], and it would benefit to assess to what degree they can tackle the continual learning problem in HAR. Unlike image data, sensor data exhibits its limitations.

- *Scarce labels* – In HAR, it is very time- and effort-consuming to label sensor data from real-world deployment. As continual learning is concerned with new, unobserved, unpredicted activities, it relies on users' self-annotation; e.g., a user provides a free-text label for the new activity that he/she is performing. Such labels can be sparse and noisy. Due to the labelling challenge, there is a smaller amount of labelled training data in HAR compared to the image datasets.

[1]The code and all the experiments are available at `https://github.com/srvCodes/continual-learning-benchmark`.

- *Imbalanced class distribution* – Some activity classes can have a high occurrence frequency and dominate the dataset while others being rare. For example, hiking can be relatively infrequent compared to sleeping.
- *Sensor noise* – Sensors can produce noisy readings due to unintended interactions with the environment or degradation over time. For example, a visitor or a pet can trigger unexpected sensor traces in a single-resident dwelling.

Also, sensor data in HAR shares the characteristics with image data: (1) *Intra-class diversity* – one activity class can have multiple patterns due to different ways of performing an activity; *e.g.*, different ways of preparing a meal. (2) *Inter-class similarity* – some activity classes have overlapping decision boundaries, which makes them difficult to separate. For example, reading and writing for a sedentary user can have very similar distributions in sensor feature space.

The above limitations of sensor data will bring extra complication in continual learning. This paper seeks to answer to the following questions: *how accurately are the existing continual learning techniques in recognising new and old activities? how computationally expensive are they in terms of memory and training time?*, *is it affordable to run them on resource-constrained devices?*, or *is their performance sensitive to the amount of training data?* To answer these questions, we adapt the existing continual learning evaluation methodology to HAR and design a general evaluation framework to assess 10 recent techniques published between 2016 and 2020 on 8 commonly used HAR datasets including both ambient binary sensors and accelerometers. We focus on two types of continual learning techniques: regularisation and rehearsal based methods, which are most appropriate for HAR tasks. Our evaluation has uncovered the following findings.

- The rehearsal techniques significantly outperform regularisation techniques on our selected datasets, and the regularisation terms alone are not able to retain the old knowledge. The regularisation terms that tackle class imbalance and inter-class separation are most effective for HAR.
- Most of the rehearsal techniques do not need to store many samples in memory (e.g., 4 or 6 samples per class), and often random sampling can outperform the other sophisticated, computation-expensive techniques.
- The selected continual learning techniques are not sensitive to training data size, and training with 30% of each dataset can achieve good accuracy.
- The computation cost for most of the selected techniques is relatively low; *i.e.*, around 33 sec-

onds for training each incremental task. Therefore, these techniques are affordable on resource-constrained devices.

The paper is outlined as follows. Section 2 defines the continual learning problem and briefly reviews the state of the art techniques in three categories: regularisation, rehearsal, and dynamic architecture. Section 3 presents a detailed description of the selected techniques. Section 4 introduces the experimental setup including datasets, evaluation process, metrics and baseline. Section 5 describes the results and Section 6 discusses our findings to shed light on the future direction of continual learning in sensor-based HAR. Finally, Section 7 concludes the paper.

2. Problem Statement

In this section, we define the setup for our continual learning problem in a task-incremental setting and briefly introduce the mainstream continual learning techniques.

The task-incremental continual learning setting assumes tasks arriving in a sequential manner, where each task comprises one or more classes. Formally, let a task sequence T of N tasks be $T = [t_1, t_2, t_3, .., t_N]$, and a task t_i ($i \in [1, .., N]$) is coupled with a set of classes $C^i = \{c_1^i, ..., c_{K_i}^i\}$ and a collection of training data $\{(\mathbf{x}_j^i, y_j^i) | y_j^i \in C^i\}_{j=1}^{M^i}$, where K_i is the number of classes and M_i is the number of training data in a task t_i. The learning objective on the ith task is to optimise a model to classify the current classes in C^i and all the previous classes in $C^{i-1} \cup ... \cup C^1$, where $C^i \cap (C^{i-1} \cup ... \cup C^1) = \emptyset$; i.e., each task has a mutually exclusive set of classes and new classes in the current task have not been observed in the previous tasks.

The major problem in task-incremental setting resides in *catastrophic forgetting* (CF), where the new knowledge learnt by the model interferes with the old knowledge so that the performance on classifying old classes degrades over time. CF is caused by the stability-plasticity dilemma [32]. Plasticity refers to the ability of a model to accommodate new knowledge while stability refers to its ability to retain old knowledge. High plasticity often causes drastic changes in the model's parameters, thus interfering with the previous knowledge captured by them. Most of the existing continual learning techniques are trying to balance plasticity and stability, and these techniques are often grouped in three categories [32]: (1) regularisation, (2) rehearsal, and (3) dynamic architectures.

2.1. Regularisation Techniques

Regularisation techniques often introduce additional terms to the loss function that constrains the weight updates of the network so as to prevent compromising the performance on old tasks. Knowledge Distillation (KD) [17], a way to transfer knowledge between different networks, has been widely adopted to retain the knowledge of old tasks when learning new ones. It prevents the current model's output deviating from the previous model's prediction that is recorded as logit output as soft labels for old classes. Learning without forgetting (LwF) [27] is the earliest attempt to employ the distillation loss in continual learning, followed by many others such as cross-distilled loss [4] and attention distillation loss [12].

Another category of regularisation techniques is to identify important parameters for old tasks and penalise updates on them. For example, Elastic weight consolidation (EWC) applies Fisher Information matrix to measure the importance of the network parameters [23]. Memory Aware Synapse (MAS) estimates the importance of a parameter based on the magnitude of the gradient; that is, how much change the output of the network is caused by a small perturbation to the parameter [1].

Group sparsity regularisation has been applied to allow for selective training of a subset of neurons. Adaptive Group Sparsity based Continual learning (AGS-CL) [20] introduces regularisation terms using two node-wise group sparsity based penalties. The first term assigns and learns new important nodes via the ordinary group Lasso penalty when learning a new task, while the second term applies the group-sparsity based deviation penalty to prevent the drift on important node parameters.

Other regularisation terms have been designed to tackle specific problems. Weight alignment has been employed to balance the distribution of new tasks' training samples and old tasks' in-memory samples [22, 50]. Customised regularisation terms have been designed to prevent task interference; for example, forcing a large margin between the old and new classes [18].

2.2. Rehearsal Techniques

Rehearsal techniques mitigate catastrophic forgetting by mixing data from previous tasks with the current task. The previous task data can be used as inputs for re-training the network or as a constraint to the network updates for penalising the interference with the previous tasks. iCaRL, a class-incremental learning technique, stores a small set of representative samples for each class in

memory, and combines these samples with new task data to update the network every time [34]. This type of techniques often requires extra memory space for old task samples and also can be prone to overfitting the stored data, instead of generalising to old tasks [25]. REMIND [15] utilises data compression and augmentation to enable more effective replay.

In contrast, gradient episodic memory (GEM) [29] uses these old task samples to impose a constraint to allow positive backward transfer; that is, preventing the gradient update from increasing the loss on previous tasks. Similarly, orthogonal gradient descent (OGD) [14] maintains a space consisting of the gradient directions of the network predictions on previous tasks and projects the loss gradients of new samples orthogonal to this gradient before backpropagation. In this way, OGD minimises the interference on old tasks and thus preserves old knowledge.

In addition to sampling from training data, in-memory samples on previous tasks can also be generated via a generative model that learns the distribution of old tasks. Generative replay model (GRM) [38] employs a Generative Adversarial Network (GAN) for generating samples on previous tasks for *pseudo rehearsal*. This type of techniques relies on the quality of generated samples; for example, GANs might suffer mode collapse in that the generated samples are clustered in one specific space, rather than diverse across the whole space. Also training GANs can add extra computation cost to training and GANs' performance can degrade over time with more and more classes being learnt [48].

2.3. Dynamic Architectures and Ensemble Techniques

Dynamic architectures tackle catastrophic forgetting by retaining the model on the old tasks while extending it with new parameters to learn new tasks. They are closely tied with ensemble methods which train multiple models for different tasks. *ExpertGate* [2] consists of a network of experts where each *expert* is a model trained on a specific task. A gating mechanism decides which expert is required for activation. This bypasses the need for loading all models, which is memory efficient as each model can be computationally intensive. Net2Net [7] is another type of dynamically evolving network which can be widened (adding more neurons) and deepened (adding more layers). Knowledge from the previous network is preserved in the newly constructed network.

Progressive network [37] keeps a pool of models that are pre-trained with previous knowledge and adds lateral connections to them for a new task. To mitigate forgetting, the parameters for previous tasks are never modified while new parameters are learned for the new task. Therefore, it

does not deteriorate the performance of previous tasks. One drawback of using this technique is that the network can become complex with an increasing number of tasks learned. Since a new network is learned for each task and it needs to be connected to the previous network, the complexity of the network structure and parameters increases quickly.

Dynamically Expandable Networks (DEN) [49] allows layer expansion and employs group sparsity regularisation to identify the neurons that are relevant to new tasks and allow selective retraining on these neurons. Compacting, Picking, and Growing (CPG) continual learning [19] adds new neurons for accommodating new tasks and then constantly applies network compression by deleting unnecessary weights.

Dynamic architectures often have been employed in computer vision applications, where there exist hundreds or thousands of classes. As our selected HAR datasets do not have such a high number of activities to learn and a decent size of a neural network often works well, we exclude this category of techniques in our study.

3. Comparison of Techniques

After presenting the overview of continual learning techniques in the previous section, now we will focus on a small set of techniques. We set three criteria for selecting state-of-the-art continual learning techniques: (1) the techniques are built on neural networks and target classification tasks, as neural networks have been widely adopted in HAR tasks for their effectiveness at feature extraction and recognition [42], (2) the techniques should be the most representative ones, which have been included in several recent continual learning surveys [25, 32], and (3) the techniques target HAR limitations including class imbalance and inter-class separation. With these criteria, we select 10 techniques from regularisation and rehearsal categories and in the following, we will provide a brief description of each and illustrate their key characteristics.

3.1. Regularisation Techniques

Learning without Forgetting [27] aims to keep the output of the old tasks from the new network close to the output from the original model. This is achieved by using the knowledge distillation (KD) loss as a regularisation term. KD is a way to transfer knowledge from a complex *teacher* model to a simpler *student* model by minimising the loss on the output class probabilities from the

teacher model [17]. KD has been widely applied to various continual learning techniques [12, 27] to distil knowledge learnt from old tasks to the model for new tasks, which is defined as follows:

$$\mathcal{L}_{KD}(y_o, \hat{y}_o) = -\sum_{i=1}^{l} y_o^{'(i)} \log \hat{y}_o^{'(i)}, \tag{1}$$

where l is the number of class labels, and $y_o^{'(i)}$ and $\hat{y}_o^{'(i)}$ are temperature-scaled *recorded* and *predicted* probabilities of the current sample for an old class label i. The temperature is used to tackle the over-confident probability that the teacher model produces on their prediction. That is,

$$y_o'^{(i)} = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}} \text{ and } \hat{y}_o'^{(i)} = \frac{(\hat{y}_o^{(i)})^{1/T}}{\sum_j (\hat{y}_o^{(j)})^{1/T}}.$$

The loss $\mathcal{L}_{KD}$ is combined with the cross-entropy loss on a new task's samples $\mathcal{L}_{CE}^n$; that is,

$$\mathcal{L}(y_n, \hat{y}_n, y_o, \hat{y}_o) = \lambda_o \mathcal{L}_{KD}(y_o, \hat{y}_o) + \mathcal{L}_{CE}(y_n, \hat{y}_n), \tag{2}$$

$$\mathcal{L}_{CE}^n(y_n, \hat{y}_n) = -y_n \log \hat{y}_n, \tag{3}$$

where y_n is the one-hot encoded vector of the ground-truth label, $\hat{y}_n$ is the predicted logit (i.e., the softmax output of the network) on new class labels, and λ_o is a loss balance weight. A larger λ_o will favour the old task performance over the new task. During training, a new batch gets first fed through the old network to record its outputs, which then are used for $\mathcal{L}_{KD}$ so that the network updates will not deviate from the old network.

Elastic Weight Consolidation (EWC) [23] draws inspiration from research on mammalian brains, which suggests that catastrophic forgetting can be avoided through the protection of synapses. EWC tries to mimic this by inhibiting changes on parameters that are deemed important for previous tasks. The importance of parameters is modelled as the posterior distribution $p(\theta|D)$; that is, optimising the parameters is to find their most probable values with respect to some data D. In the context of continual learning, if the data D is assumed to consist of two independent tasks A and B, then

$$\log p(\theta|D) = \log p(D_B|\theta) + \log p(\theta|D_A) - \log p(D_B).$$

The posterior probability $p(\theta|D_A)$ contains information about which parameters are important to task A. However, the true probability $p(\theta|D_A)$ is intractable, and thus it is estimated via Laplace

approximation as a Gaussian with diagonal precision determined by the Fisher Information Matrix (FIM). The loss function of EWC is defined as:

$$\mathcal{L}(\theta) = \mathcal{L}_{CE}^{B}(\theta) + \sum_{i} \frac{\lambda}{2} F_i(\theta_i - \theta_{A,i}^{*})^2, \tag{4}$$

where $\mathcal{L}_{CE}^{B}$ is the loss on the new task B only, λ indicates the importance of the old task A with respect to B, and i is the parameter index. Originally, one FIM is required for each task, and later it can be resolved by propagating them into a quadratic penalty term. However, this formulation assumes the FIM to be diagonal, which is not always the case. Rotating EWC (R-EWC) [28] improves upon EWC by reparameterising θ through rotation in a way that it does not change outputs of the forward pass but the computed FIM is approximately diagonal.

Memory Aware Synapse (MAS) [1], inspired by neuroplasticity, also considers the importance of network parameters, which is measured in an online, unsupervised manner. Here, the importance is approximated by the sensitivity of the learned function to a parameter change. Given a data point x_k, the network output is defined as $F(x_k; \theta)$. A change in the network output caused by a small perturbation $\delta = \{\delta_{ij}\}$ in the parameters $\theta = \{\theta_{ij}\}$ can be approximated as:

$$F(x_k; \theta + \delta) - F(x_k; \theta) \approx \sum_{i,j} g_{ij}(x_k)\delta_{ij}$$

$$g_{ij}(x_k) = \frac{\partial(F(x_k, \theta)}{\partial \theta_{ij}}.$$

where g is the gradient with respect to the parameter θ. By accumulating gradients over all the data points, the importance weight on a parameter θ_{ij} is computed as:

$$\Omega_{ij} = \frac{1}{N} \sum_{k=1}^{N} ||g_{ij}(x_k)||.$$

When learning a new task, the loss function is defined as:

$$\mathcal{L}(\theta) = \mathcal{L}_{CE}^{n}(\theta) + \frac{\lambda}{2} \sum_{i,j} \Omega_{ij}(\theta_{ij} - \theta_{ij}^{*})^2, \tag{5}$$

where $\mathcal{L}_{CE}^{n}(\theta)$ is the CE loss on the new task, θ_{ij} and θ_{ij}^{*} are the new and old network parameters, and λ is a hyperparameter that indicates the regularisation strength. The idea is to retain the old knowledge by penalising large updates on the sensitive network parameters.

3.2. Rehearsal Technique

Incremental Classifier and Representation Learning (iCaRL) [34] is an early attempt of rehearsal approaches for class-incremental learning. It leverages memory replay and regularisation. After training on each task, it employs the herding sampling technique [43] to select a small number of exemplars (i.e., representative samples) from the current task's training data and stores them in memory. Herding works by choosing samples that are closest to the centroid of each old class. When the next task becomes available, the in-memory exemplars will be combined with the new task's training data to update the network. The loss function of iCaRL is the same as Equation 1 in LwF, which is a combination of CE loss on new classes and the KD loss on old classes to allow knowledge transfer.

Incremental Learning In Online Scenario (ILOS) [16], similar to iCaRL, also employs memory replay with KD loss for regularisation. The key difference is that ILOS uses an updated version of the CE loss. It introduces an accommodation ratio $0 \leq \beta \leq 1$ to adjust the proportion of logits between the current and previous model:

$$\tilde{y}'^{(i)} = \begin{cases} \beta y^{(i)} + (1-\beta)\hat{y}'^{(i)}, & 1 \leq i \leq n \\ y^{(i)}, & n+1 \leq i \leq n+m \end{cases} \tag{6}$$

where the indices $[1, n]$ refer to old classes, $[n+1, n+m]$ refer to new classes, $y^{(i)}$ are the one-hot encoded output logits of the current model and $\hat{y}'^{(i)}$ are the recorded old classes' output logits from the previous model. In this way, the output from the previous model will be retained in the current network and the retaining degree is regulated via the parameter β. The larger the β, the more retained the output from the previous model. While the KD loss is still calculated using $y^{(i)}$, the CE loss is now based on the adjusted output $\tilde{y}'^{(i)}$ instead of the recorded output $\hat{y}^{(i)}$:

$$\tilde{\mathcal{L}}_{CE} = \sum_{i=1}^{n+m} -y^{(i)} \log[\tilde{y}'^{(i)}]. \tag{7}$$

The final loss is the combination of the above CE loss and KD loss:

$$\mathcal{L}_{ILOS} = \alpha L_{KD} + (1-\alpha)\tilde{\mathcal{L}}_{CE}. \tag{8}$$

Following the guideline in the original paper, both α and β can be set to 0.5.

Gradient Episodic Memory (GEM) [29] alleviates forgetting by controlling the gradient updates to balance the performance on old and new classes. GEM uses a memory space to host examples

from previous classes $\cup_{k\in[1,t-1]}M_k$, where M_k is the samples stored on a previous task k ($k \leq t$) and t is the current task's index. When training a new task t, M_k is used in an inequality constraint to avoid an increase in loss. Hence, the loss of a current task $\mathcal{L}$ must be smaller than or equal to the loss of the previous model:

$$minimize_\theta \quad \mathcal{L}(f_\theta, D_t) \quad s.t. \quad \mathcal{L}(f_\theta, M_k) \leq \mathcal{L}(f_\theta^{t-1}, M_k) \qquad \forall k < t \tag{9}$$

where f_θ represents the model with parameters θ. To achieve this, GEM restricts the angle between gradient vectors g_k of previous and current task g_t to be no greater than 90°. If the angle is greater than 90°, the new gradient vector gets projected to the euclidean distance closest vector $\tilde{g}$ that is inside the allowed range. Thus, the optimisation problem can be defined as:

$$minimize_{\tilde{g}} \quad \frac{1}{2}||g - \tilde{g}||_2^2 \quad s.t. \quad \langle \tilde{g}, g_k \rangle \geq 0 \qquad \forall k < t$$

Solving this quadratic program problem for all in-memory samples is very computationally expensive. To address the computational cost, A-GEM [6] is proposed to ensure that there is no increase in the average loss over the episodic memory. That is, A-GEM only uses a smaller set of in-memory samples to calculate an average gradient instead of all gradients for a task.

Learning a Unified Classifier Incrementally via Rebalancing (LUCIR) [18] was proposed to tackle the imbalance between a small number of in-memory samples from old tasks and a larger number of samples from a new task. The imbalance results in the training being biased towards new tasks. LUCIR proposes multiple (loss-) components to mitigate this adverse effect. Firstly, cosine normalisation is used in the last layer to level the difference of the embeddings and biases between all classes since those are significantly higher for new tasks. Secondly, less-forget constraint is introduced to prevent forgetting old classes' geometric configurations by encouraging the extracted features of new classes similarly rotated to those of old ones; that is:

$$\mathcal{L}_{dis}^G(x) = 1 - \langle \tilde{f}^*(x), \tilde{f}(x) \rangle, \tag{10}$$

where $\tilde{f}(x)$ and $\tilde{f}^*(x)$ are normalised features generated by the new and old model, and $\langle , \rangle$ measures the cosine similarity between the two normalised vectors. Thirdly, a margin ranking loss is used to enhance inter-class separation. For each in-memory sample x, it aims to separate old classes from all the new classes by a margin. Using x as an anchor, LUCIR finds positive and negative embeddings and aims to maximise their distances. The positive embeddings are from the ground-truth class of

x (represented as $\tilde{\theta}(x)$), while the negative embeddings are from the top-K new classes that produce the highest response to x (represented as $\tilde{\theta}^k$), indicating the classes that x is mostly confused with. Then the margin ranking loss is defined as:

$$\mathcal{L}_{mr}(x) = \sum_{k=1}^{K} \max(m - \langle \tilde{\theta}(x), \tilde{f}(x) \rangle + \langle \tilde{\theta}^k . \tilde{f}(x) \rangle, 0). \tag{11}$$

This leads to the combined loss function as follows:

$$L = \frac{1}{|N|} \sum_{x \in N} (L_{CE}(x) + \lambda L_{DIS}^{G}(x)) + \frac{1}{|N_o|} \sum_{x \in N_o} L_{MR}(x), \tag{12}$$

where N refers to all the training samples and N_o ($\subset N$) refers to the reserved old samples. λ is a dynamic weight to adjust how much knowledge of the previous model needs to be preserved depending on how many new classes are introduced. It is calculated by multiplying the base λ with the squared root of the ratio between new and old classes; that is, $\lambda = \lambda_{base} \sqrt{|C_N|/|C_o|}$. It regulates the degree of preserving the old knowledge by taking into account the number of new classes being added. For example, when there are many new classes are introduced, the model would preserve less old knowledge to allow the model to adapt to the new knowledge, and vice versa. In general, λ_{base} is set as 5.0 and the margin value m in Eq 11 as 0.5 [18].

Weight Alignment for Maintaining Discrimination and Fairness (WA-MDF) is another approach that tackles the mentioned imbalance problem by correcting the biased weights in the last fully connected (FC) layer after training each task. It aligns the norms of the weight vectors of new classes to those of old classes. The FC layer output of a model can be expressed in a general form: $o(x) = W^T \phi(x)$, where $\phi(x)$ is the feature extraction function of an input x and W is the weight vector of the FC layer. W can be separated into $W = (W_o, W_n)$, where W_o and W_n refers to the weights corresponding to the old and new classes; that is,

$$W_o = (w_1, w_2, ..., w_{C_o^b})$$
$$W_n = (w_{C_o^b+1}, ..., w_{C_o^b+C^b}),$$

where C^b is the total number of classes and C_o^b is the number of old classes. Their norms are defined as:

$$Norm_o = (||w_1||, ||w_2||, ..., ||w_{C_o^b}||)$$
$$Norm_n = (||w_{C_o^b+1}||, ..., ||w_{C_o^b+C^b}||).$$

The normalised weights on new classes are $\widehat{W}_n = \gamma W_n$, where $\gamma = \frac{Mean(Norm_o)}{Mean(Norm_n)}$. In this way, the average norm of the weights on the new classes will be the same as that on the old classes. The corrected output logits from the FC layer will be written as:

$$O_{corrected}(x) = (W_o^T \phi(x), \gamma W_n^T \phi(x))^T. \tag{13}$$

As a norm-control method; i.e., keeping a check on the norms of class embeddings following each gradient update, WA-MDF clamps the parameters of the FC layer at zero. Assuming that a network usually employs variants of ReLU activation units, the clipping facilitates the projection of weights. This helps eliminate large negative elements of weight vectors thus making its norm more consistent with the non-negative output logits of the ReLU function. While such a restriction seems to interfere with the convergence of the model, several studies have shown that training neural networks with weights projected via such distortions makes them robust to other types of distortions [30]. Here, distortion refers to different ways of weight projection operations; for example, adding Gaussian noise or performing additive, multiplicative, or power operations on weights [30]. Such distortion allows weights to accumulate small gradient updates and thus leads to improved performance.

Weight Alignment for Adjusting Decision Boundary (WA-ADB) is also proposed to re-scale the weight vectors of majority and minority classes, but in a more general class-imbalance scenario [22]. Different from the above WA-MDF, WA-ADB re-scales the weights based on the sample ratio of old and new classes instead of their weight norms. For a dataset D with K classes, given that n_i is the number of samples of class i ($n_1 \geq ...n_i... \geq n_K$), the weight vectors of each class are re-scaled by an exponent of the re-scaling factor n_1/n_i; i.e., $w_i = (\frac{n_1}{n_i})^\gamma w_i$. During training, the weight vectors will be normalised after each gradient update. While a larger γ scales up the volume of feature space of a model allocated to infrequent classes, weight vector normalisation forces the class conditional probability to have the same variance independent of the sample size.

To adapt WA-ADB to incremental learning scenario, we make the following adjustment. For an incremental training step i, the dataset D is the combination of in-memory samples on old tasks and new training samples on a new task. We then re-scale the weights of all the classes of old tasks by the same factor, i.e., n_1/n_i, where n_1 is the number of samples on the largest class that has been seen so far and n_i is the holdout size. Because n_i is fixed due to the memory size, we only compute this number once. While the bias term is dropped in the original work [22], we take into account that the biases for classes of new tasks also have norms larger in average than that of classes from

old tasks. As a result, we re-scale the biases of each class by the corresponding weight re-scaling factor computed in the above step.

Bias Correction (BiC) [45] introduces a BiC correction layer after the last FC layer to adjust the weights in order to tackle the imbalance problem. There are two stages of training. In the first stage, the network will be trained with new task data and in-memory samples of old tasks using the CE and KD loss similar to iCaRL. In the second stage, the layers of the network are frozen and a linear BiC layer is added to the end of the network and trained with a small validation set consisting of samples from both old and new tasks. The linear model of the BiC layer corrects the bias on the output logits for the new classes:

$$\tilde{y}_k = \begin{cases} \hat{y}_k & 1 \leq i \leq n \\ \alpha \hat{y}_k + \beta, & n+1 \leq i \leq n+m \end{cases}$$

where $\hat{y}_k$ is the output logits on the kth class, the old classes are $[1, ..., n]$ and the new classes are $[n+1, ..., n+m]$. α and β are the bias parameters of the linear model, which are optimised in the following loss function:

$$L_{BiC} = -\sum_{k=1}^{n+m} \delta_{y=k} \log \texttt{softmax}(\tilde{y}_k),$$

where $\delta_{y=k}$ is the indicator function to check if the ground-truth label y is the same as a class label k. The intuition is that a balanced small validation set for all seen classes can counter the accumulated bias during the training of the new task. As the non-linearities in the data are already captured by the model's parameters, the linear function proves to be quite effective in estimating the bias in the output logits of the model arising.

So far we have described all the 10 selected continual learning techniques and in the next section, we will introduce the evaluation framework and methodology for assessing and comparing them on the HAR datasets.

4. Experimental Setup and Evaluation Methodology

The main objective of this paper is to assess *to what degree* the state-of-the-art continual learning techniques can enable continual activity recognition. More specifically, we will seek to answer the following questions:

1. What is the performance of these continual learning techniques on HAR datasets?
2. Which techniques best balance plasticity and stability when incrementally learning new activities?
3. Which regularisation term works best for what type of data?
4. What is the impact of in-memory sample sizes on the accuracy of these techniques?
5. Are these techniques sensitive to the size of training data?
6. What is the computation cost of these techniques in terms of memory and training time?

To answer these questions, we select 8 HAR datasets (in Section 4.1), design the evaluation process (in Section 4.2), select the evaluation metrics (in Section 4.3) that are appropriate for HAR, describe the baseline approach for comparison (in Section 4.4), and illustrate the architecture configuration and hyperparameter tuning for all the techniques (in Section 4.5).

4.1. Datasets

To present a comprehensive profile of selected continual learning techniques, we will assess on a wide variety of most representative, state-of-the-art HAR datasets, ranging from simple datasets (with a single user, a small number of features and activity classes) to more complex datasets (with multiple users, a large number of features and activity classes). Driven by these criteria, we select the following set of publicly available, third-party datasets that are collected on accelerometer and event-driven binary sensor data, as these two are the most common sensor types in HAR. Table 1 and Figure 3 summarise the main characteristics of these datasets.

Table 1: Main characteristics of selected datasets

Sensor Types	Binary Sensors					Accelerometers		
Datasets	HA	WS	MILAN	ARUBA	TWOR	PAMAP2	HAPT	DSADS
No. of samples	488	909	20525	71278	30128	7312	10929	9120
No. of features	16	32	28	31	43	243	561	405
No. of classes	7	9	15	11	23	12	12	19
Balanced (Y/N)	N	N	N	N	N	N	N	Y

Binary sensor data We use House A (denoted as *HA*) from the University of Amsterdam [40] and 4 CASAS smart home datasets (denoted as *WS*, *Milan*, *Twor*, and *Aruba*) from Washington State University [9]. They are collected on a wide range of event-driven binary sensors, including passive infra-red sensors, state-change sensors, switch sensors, pressure sensors, and water flow sensors.

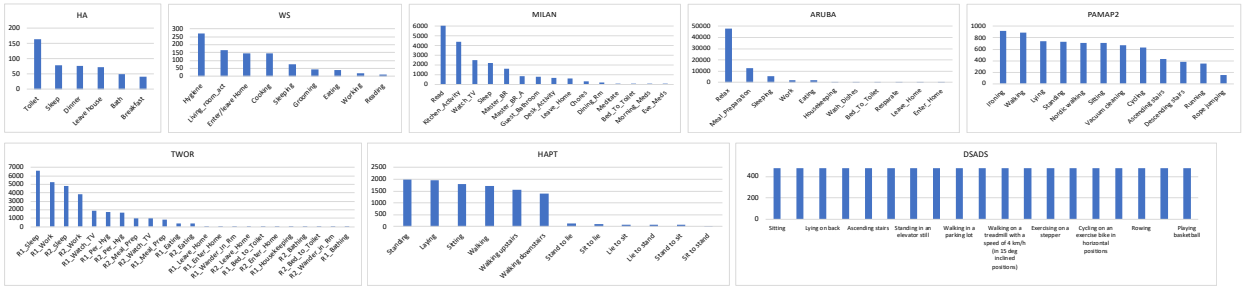


Figure 3: Activity histograms on all the selected datasets.

On these datasets, we apply state-of-the-art techniques [13] to segment the raw sensor data into a 60-second interval and generate features as an activation ratio of each sensor in the interval.

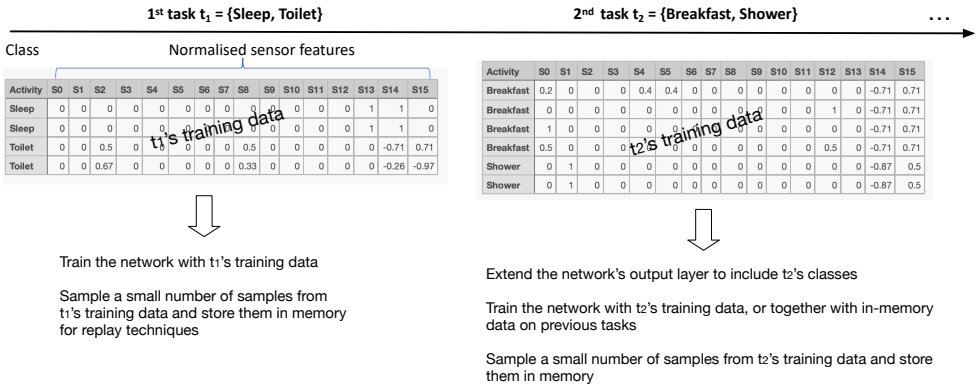
Accelerometer sensor data Physical Activity Monitoring Dataset (*PAMAP2*) [35] contains 12 activities such as sitting, lying, and house cleaning. The data is collected on 9 users with 3 accelerometer units on each user's chest, dominant arm and side ankle. Daily and Sports Activities Dataset (*DSADS*) [3] contains 19 activities such as running, rowing, and sitting. The data is collected on 8 users with 5 accelerometer units on each user's torso, arms, and legs. On these datasets, we use the sensor features that have been extracted from raw accelerometer data; that is, on each accelerometer unit, 27 features are generated, including mean, standard deviation, correlations, and spectrum peak position. Human Activity Recognition Dataset (*HAPT*) [36] contains 12 daily activities collected on 30 subjects with a smartphone (Samsung Galaxy S II) on their waist. There are 561 features extracted from accelerometer and gyroscope readings.

4.2. Evaluation Process

We follow the state-of-the-art task-incremental evaluation methodology [39], where we assign 2 randomly sampled classes to each task and form a sequence of $|C|/2$ consecutive tasks, where $|C|$ is the total number of classes in a dataset. In practice, the number of classes in each task can vary, depending on the arrival of new classes and update cycle of the model. Here, we set the number of classes as 2, as a good trade-off to assess the incremental learning capability and the training time.

Also, we employ the stratified train-test split [39] to split each class's data into 70% for training and 30% for testing. As accelerometer datasets often have multiple users, to avoid data leakage, we split training and testing data of each class by users; that is, we use 70% of users' data for training and the remaining 30% of users' data for testing.

Given a task sequence, the network will be initialised and trained with the first task's training



Activity	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Sleep	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Sleep	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Toilet	0	0	0.5	0	0	0	0	0	0.5	0	0	0	0	0	-0.71	0.71
Toilet	0	0	0.67	0	0	0	0	0	0.33	0	0	0	0	0	-0.26	-0.97

Activity	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Breakfast	0.2	0	0	0	0.4	0.4	0	0	0	0	0	0	0	0	-0.71	0.71
Breakfast	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-0.71	0.71
Breakfast	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.71	0.71
Breakfast	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.5	-0.71	0.71
Shower	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-0.87	0.5
Shower	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-0.87	0.5

Figure 4: An example of task-incremental evaluation

data. Then for each subsequent task, the network's output layer will be extended to include its new classes and the network will be trained with the task's training data and in-memory data. The in-memory data are sampled from all the previous tasks' training data. The size of in-memory data is determined by the memory constraint of a particular HAR system, and it will contribute to retaining old knowledge and thus affect the accuracy of old class classification. Figure 4 presents an example of task-incremental evaluation on a HAR dataset. As the performance of continual learning can be subject to task sequences, to reduce the bias, we generate several random task sequences (i.e., 30) and report averaged accuracy.

4.3. Evaluation Metrics

When training a new task, we compute three types of accuracy. (1) *Base* accuracy – the accuracy of recognising the activity classes in the first task; (2) *Old* accuracy – the accuracy of recognising all the old activity classes that have learnt before the current task. Both base and old accuracy will indicate the stability of the model; (3) *New* accuracy – the accuracy of recognising the new activity classes in the current task, which will indicate the plasticity of the model; and (4) *All* accuracy – the accuracy of recognising all the activity classes that have learned so far, which will indicate the overall performance of the model. The accuracy is measured in *F1-scores*, which balances precision and recall. As most of the HAR datasets have an imbalanced class distribution, we use *F1-macro* and *F1-micro*.

To understand the retention of knowledge for a given task j at an incremental task k, we employ a commonly used measure – forgetting score (FS) [5]. After a model is trained incrementally till task $k > j$, the forgetting score f_k^j is computed as the difference between the maximum accuracy

gained for the task j throughout the learning process. However, this score does not concern with the difficulty level of each task. For example, the accuracy decreasing from 60% to 40% on a difficult task suggests more forgetting than the accuracy decreasing from 90% to 70% on an easy task. To address this limitation, we propose an adapted forgetting score by normalising it with the best accuracy that can be achieved on a task. That is,

$$f_k^j = 1 - \frac{a_{k,j}}{\max\limits_{l \in 1,..,k-1} a_{l,j}},$$

where $a_{k,j}$ is the accuracy for a previous task j at the current task k[2]. Finally, the average forgetting FS_k at $k-$th task is normalised against the number of tasks seen previously, i.e.,

$$FS_k = \frac{1}{k-1} \sum_{j=1}^{k-1} f_k^j.$$

A large FS score implies server forgetting. At the extreme, when FS is 1, it suggests that the knowledge on the old task is completely forgotten.

4.4. Baseline

Besides the selected models, we consider two baselines that serve as an upper and lower bound of the performance of continual learning. (1) **Offline batch learning**: we train a network with the training data on all the classes simultaneously, and (2) **finetuning**: we do not use any holdout samples for replay and only use training data on each new class to update the model with the plain cross-entropy loss.

4.5. Architecture Configuration

While the original implementations of our selected techniques use a variety of architectures[3] and classifiers[4], we try to maintain a fair comparison premise by using the fully connected networks across all our experiments. To retain the best characteristics of these techniques, we follow the state-of-the-art methodology [5, 21] to conduct grid search for the network architecture on each dataset. Table 2 provides a list of optimised configurations. The initial learning rate (LR) is reduced by a factor of 10 after every scheduler step following the first scheduling epoch. For the

[2]We ignore classes whose maximum accuracy is not greater than 0, avoiding the arithmetic error.

[3]Most of the vision-based methods use standard CNN architectures.

[4]For instance, iCaRL uses nearest-of-mean classifier.

choice of the number of neurons in the hidden layers, we rely on fractions of feature dimensions of each dataset. For the particular case of training GEM, we tune the initial learning rate to 0.01 for TWOR and ARUBA, and to 0.001 for DSADS, as a larger learning rate makes it difficult to converge and leads to low accuracy. The scheduling epochs for TWOR and ARUBA are set to 40 and that for DSADS to 50 while the wight decay rates are set to $1e^{-6}, 1e^{-4}$ and $2e^{-6}$ for TWOR, DSADS and ARUBA respectively.

Table 2: Hyperparameter configuration for training

Hyperparameter	HA	WS	MILAN	ARUBA	TWOR	PAMAP2	HAPT	DSADS
Hidden layer dimensions	[16, 8, 8]	[32, 16, 16]	[56, 28, 14]	[62, 31, 15]	[43, 21, 10]	[486, 243, 121]	[1122, 561, 280]	[405, 202, 202, 101]
Batch size	15	15	15	64	15	20	20	20
Initial LR	0.01	0.01	0.01	0.1	0.1	0.01	0.01	0.01
LR scheduler step	50	40	70	40	30	50	40	50
First scheduling epoch	110	90	70	40	30	100	60	100
Epochs till convergence	200	200	200	200	150	200	160	200
Weight decay rate	1.00E-04	1.00E-04	2.00E-04	2.00E-06	1.00E-04	1.00E-04	2.00E-04	1.00E-04

For each comparison technique, we run grid search on their own hyperparameters, and select the best value that leads to the highest accuracy on each dataset. We list the setting in the following. For LwF, the loss balance weight λ_o is set as 1.6 for all the datasets. For R-EWC, the regularisation coefficient λ in Eq 4 is set as 5 for ARUBA and TWOR and 3 for all the other datasets. For MAS, the regularisation coefficient λ in Eq 5 is set as 0.1 for TWOR, ARUBA, and MILAN datasets and 0.25 for all the other datasets.

5. Results and Analysis

This section presents the results in response to the research questions raised in Section 4. For each question, we summarise the key results in bold, followed by the analysis.

5.1. Overall Comparison

Table 3 and 4 report the mean and standard deviation of micro- and macro-F1 scores on 10 selected and 2 baseline models across 8 datasets over 30 runs. The last column of these two tables averages the accuracy across all the datasets and demonstrates which technique works best. For all the models that use memory replay, we randomly sample 6 samples from each old class after training each task.

The offline baseline provides the reference accuracy which implies the difficulty level of each dataset. As shown in Table 3 and 4, the datasets that have more balanced activity distribution and

Table 3: Comparison of micro-F1 scores on comparison techniques (The best accuracy is highlighted in bold.)

Model	HA	WS	MILAN	ARUBA	TWOR	PAMAP2	HAPT	DSADS	Average
Offline	91.84	94.14	84.20	96.82	77.29	88.07	94.91	82.06	89.30
ILOS	82.97 +/- 09.71	**90.98 +/- 03.34**	62.81 +/- 09.52	**85.52 +/- 16.18**	**66.36 +/- 04.39**	**80.37 +/- 03.09**	**81.20 +/- 03.50**	60.91 +/- 04.46	**76.39**
WA-MDF	79.91 +/- 09.59	87.95 +/- 06.84	59.75 +/- 12.09	82.09 +/- 08.38	61.09 +/- 07.17	79.39 +/- 03.63	79.80 +/- 05.03	60.96 +/- 03.98	73.87
WA-ADB	81.97 +/- 19.11	89.81 +/- 03.11	**64.28 +/- 04.89**	73.76 +/- 21.59	55.71 +/- 08.74	76.73 +/- 03.81	75.74 +/- 05.82	59.52 +/- 03.89	72.19
BiC	**84.40 +/- 08.74**	89.17 +/- 04.31	47.09 +/- 15.27	64.81 +/- 26.70	57.63 +/- 11.70	76.53 +/- 03.23	75.65 +/- 05.19	63.86 +/- 03.36	69.89
GEM	81.07 +/- 19.35	85.60 +/- 04.31	63.01 +/- 04.80	79.71 +/- 05.27	54.55 +/- 14.95	67.62 +/- 04.43	77.37 +/- 04.39	51.99 +/- 07.66	70.12
LUCIR	65.08 +/- 31.22	80.68 +/- 16.48	46.08 +/- 18.14	56.05 +/- 27.30	62.22 +/- 30.88	76.95 +/- 02.31	77.78 +/- 05.37	**71.69 +/- 02.39**	67.07
iCaRL	76.80 +/- 12.39	87.52 +/- 04.92	34.53 +/- 15.19	54.23 +/- 28.18	49.83 +/- 16.06	71.48 +/- 03.75	73.02 +/- 06.10	59.50 +/- 04.24	63.36
R-EWC	18.75 +/- 09.05	13.16 +/- 09.94	07.36 +/- 08.02	04.79 +/- 12.69	05.60 +/- 06.79	17.43 +/- 05.38	16.91 +/- 09.52	05.68 +/- 01.34	11.20
MAS	15.10 +/- 06.79	10.40 +/- 07.61	05.74 +/- 07.63	03.13 +/- 04.81	04.48 +/- 05.82	17.39 +/- 05.34	16.91 +/- 09.53	05.70 +/- 01.39	09.86
LwF	14.88 +/- 7.34	10.44 +/- 07.59	08.80 +/- 10.15	12.49 +/- 20.22	04.84 +/- 07.50	17.44 +/- 05.35	16.96 +/- 09.54	05.77 +/- 01.49	11.45
Finetuning	13.76 +/- 06.86	09.90 +/- 07.66	05.73 +/- 07.63	03.11 +/- 04.82	03.91 +/- 05.34	17.40 +/- 05.35	16.90 +/- 09.52	05.40 +/- 00.72	09.51

easier-to-separate classes gain higher micro- and macro-F1 scores; *e.g.*, WS (94.14% and 87.41% in micro- and macro-F1) and PAMAP2 (94.91% and 84.78% in micro- and macro-F1). The imbalanced datasets often result in higher micro-F1 but lower macro-F1; *e.g.*, ARUBA (96.82% and 58.82% in micro- and macro-F1) and TWOR (77.20% and 45.35% in micro- and macro-F1). As lower bound, the finetuning baseline suffers the most catastrophic forgetting, with the overall micro-F1 9.51% and macro-F1 2.11% averaged across all the datasets, which are significantly lower than the accuracy achieved from offline. In comparison with the above baselines, we draw the following observations on the selected continual learning techniques.

Table 4: Comparison of macro-F1 scores on comparison techniques (The best accuracy is highlighted in bold.)

Model	HA	WS	MILAN	ARUBA	TWOR	PAMAP2	HAPT	DSADS	Average
Offline	79.50	87.41	56.26	58.82	45.35	84.78	84.45	81.70	72.28
ILOS	69.89 +/- 12.38	79.16 +/- 06.48	38.51 +/- 07.04	**48.82 +/- 10.97**	36.12 +/- 05.09	**76.40 +/- 04.01**	68.31 +/- 4.02	58.71 +/- 04.76	59.49
WA-MDF	68.51 +/- 11.71	78.17 +/- 05.54	40.01 +/- 08.86	48.01 +/- 06.97	37.17 +/- 03.77	75.72 +/- 04.02	67.33 +/- 04.18	59.71 +/- 04.32	59.37
WA-ADB	73.25 +/- 19.83	**80.37 +/- 04.19**	**45.68 +/- 04.08**	44.60 +/- 12.45	36.41 +/- 05.22	73.27 +/- 03.66	64.68 +/- 03.58	58.13 +/- 04.07	**59.55**
BiC	**74.51 +/- 08.10**	80.11 +/- 04.23	31.82 +/- 08.72	37.59 +/- 13.06	34.83 +/- 08.70	72.75 +/- 03.82	64.79 +/- 04.19	62.55 +/- 03.41	57.37
GEM	70.87 +/- 19.27	70.68 +/- 07.31	41.20 +/- 02.88	38.87 +/- 07.16	27.91 +/- 08.46	62.92 +/- 04.79	55.55 +/- 05.44	50.35 +/- 08.34	52.29
LUCIR	57.42 +/- 34.24	72.31 +/- 15.97	34.97 +/- 12.63	35.97 +/- 12.85	**53.43 +/- 32.82**	72.74 +/- 02.26	**68.68 +/- 04.38**	**70.54 +/- 02.74**	58.26
iCaRL	68.12 +/- 13.03	77.93 +/- 04.92	24.94 +/- 08.98	34.43 +/- 11.06	29.14 +/- 09.49	67.91 +/- 03.47	60.89 +/- 04.69	57.37 +/- 04.67	52.59
R-EWC	06.82 +/- 05.26	04.10 +/- 04.08	01.46 +/- 01.83	00.67 +/- 01.45	00.73 +/- 00.94	05.12 +/- 01.28	04.36 +/- 02.77	00.75 +/- 00.68	03.00
MAS	04.22 +/- 02.48	02.46 +/- 02.41	00.82 +/- 01.13	00.56 +/- 00.78	00.48 +/- 00.82	04.99 +/- 01.27	04.42 +/- 02.72	00.72 +/- 00.60	02.33
LwF	03.75 +/- 01.54	02.38 +/- 02.26	00.98 +/- 01.06	01.63 +/- 02.28	00.36 +/- 00.54	05.05 +/- 01.36	04.49 +/- 02.65	00.76 +/- 00.65	02.43
Finetuning	03.38 +/- 01.45	01.91 +/- 01.38	00.67+/- 00.81	00.51 +/- 00.76	00.31 +/- 00.40	05.01 +/- 01.28	04.41 +/- 02.74	00.68 +/- 0.83	02.11

Rehearsal methods significantly outperform regularisation-alone methods, as the averaged difference in micro- and macro-F1 is around 50%. More specifically, from R-EWC to iCaRL, micro-F1 increases from 11.20% to 63.36% in Table 3, and macro-F1 from 3% to 52.59% in Table 4. Regularisation alone does not help retain the knowledge of the original model as LwF, MAS and EWC only produce roughly the lower bound accuracy. For example, the success of LwF depends on the similarity of new tasks to old tasks. Distribution shift between old and new tasks will result

in a discrepancy in the KD loss when predicting the class probability using the old model. The errors are accumulated over incremental learning and will significantly impact its performance [25]. In HAR, each activity class can have a drastically different sensor feature signature, so in our experiment these regularisation-only methods perform much worse than they do in computer vision experiments.

Among rehearsal methods, ILOS, WA-MDF, and WA-ADB perform the best, improving on the basic rehearsal method iCaRL over 13% in micro-F1 and 7% in macro-F1. With memory replay, retraining the model is often affected by the imbalance between a large number of new task's data and a small number of in-memory samples. These methods have effectively avoided optimising the model towards the majority classes and thus better retained the knowledge of the old classes.

GEM does not perform better than iCaRL, as it only achieves an increase of 7% in micro-F1 and the same macro-F1 score as iCaRL. We have also considered the improved version of GEM: A-GEM [6]. However, in our experiments, A-GEM often produces worse accuracy than GEM; for example, on PAMAP2 dataset, A-GEM achieves micro-F1 of 37.86% and macro-F1 of 28.35%, which is more than 30% lower than GEM. One possible reason is that activities can have diverse patterns, so when A-GEM down-samples the holdout data, it has an even smaller coverage of space and thus its accuracy is worse than GEM.

5.2. Balance between Stability and Plasticity

To look into stability and plasticity over incremental learning, Figure 5 presents task-level micro-F1 of *base*, *old*, *new*, and *all* classes on each technique. As we can see, regularisation methods demonstrate better plasticity as with the increase of tasks, the new accuracy of LwF and EWC remains high. These methods also exhibit poor stability as their base and old accuracy stays at the bottom of the plots, even from the second task on. Overall, the weight alignment methods in the rehearsal category achieve a better balance between stability and plasticity, as the overall accuracy of ILOS, WA-MDF, and WA-ADB suffers less steep drop. Besides, we can observe that each technique has a different performance profile on these 8 datasets, implying the characteristics of the datasets might impact the effectiveness of each technique.

Rehearsal methods that tackle imbalance are less plastic to new classes. With a direct tuning of the new model's logits based on the previous model, ILOS can help surpass complex regularisation operations in retaining the knowledge but it is not able to quickly adapt to new

classes. We also observe that LUCIR performs poorly on new tasks but is robust at preserving old knowledge. An intuitive explanation could be that the design of marginal ranking reinforces the model's confidence at recognising ground truth embedding for old class samples after multiple incremental training steps.

To further inspect the catastrophic forgetting, we present the new forgetting score FS in Figure 6. FS on these three regularisation methods is at the upper bound of 1.0 for the first task and stays much higher than the other techniques for the following tasks, suggesting that these techniques suffer an almost immediate total forgetting effect where they cannot come back from. The high forgetting scores of R-EWC conform to the finding of [21] stating that EWC-based methods are poor at learning new categories incrementally.

Among rehearsal-based methods, ILOS and WA-ADB exhibit the best knowledge retention as their averaged FS across the tasks on all the datasets is the lowest; *e.g.*, the old classes only lose 20% of their best accuracy throughout continuous learning. In contrast, iCaRL and LUCIR are the worst with their FS being 30%, which is averaged over all the datasets in Figure 6. GEM shows a relatively greater tendency of increase in FS as the number of incremental tasks grows. This is evident across the plots for DSADS, ARUBA, TWOR, WS and HA.

We also observe that after a certain number of incremental tasks, the inertia of forgetting on rehearsal methods is relieved. For instance, iCaRL, BiC, LUCIR and ILOS reach the threshold on ARUBA at the 3rd task while GEM and WA-ADB attain this at the 4th task following which their respective FS witnesses either a plateau or start decreasing. There is no clear correlation between the forgetting effect and the number of tasks being learnt, as it can be dataset-specific, especially the interference between the old and new activities.

There exists a strong effect of the amount and distribution of training data attributes on forgetting. For the rehearsal methods, we can see that the datasets that have a long tail distribution (i.e., many activities have low frequency) have much higher forgetting scores. For example in Figure 3, TWOR has 19 out of 23 activities whose occurrence is less than 6%, MILAN has 11 out of 15 activities whose occurrence is less than 8%, and ARUBA has 8 out of 11 activities whose occurrence is less than 3%. As shown in Figure 6, FS of the rehearsal based techniques on these datasets is around 15% higher than those on the other datasets.

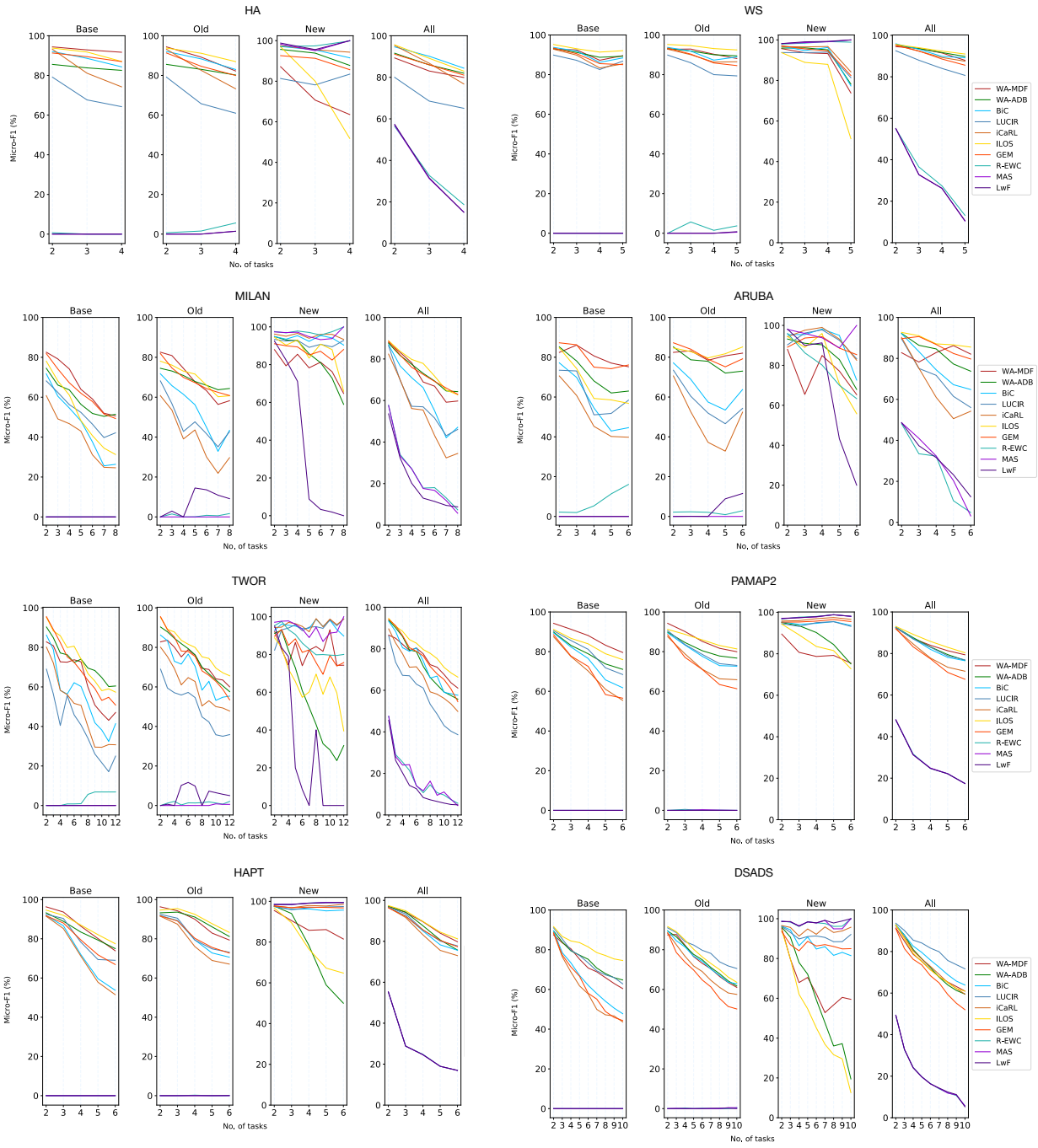


Figure 5: Accuracy comparison of task-level performance. It records micro-F1 scores on *Base*, *Old*, *New* and *All* classes.

5.3. Effect of Regularisation

From the above results, we can see that regularisation alone does not demonstrate any advantage from a naive finetuning approach, but will their performance be improved when combined with

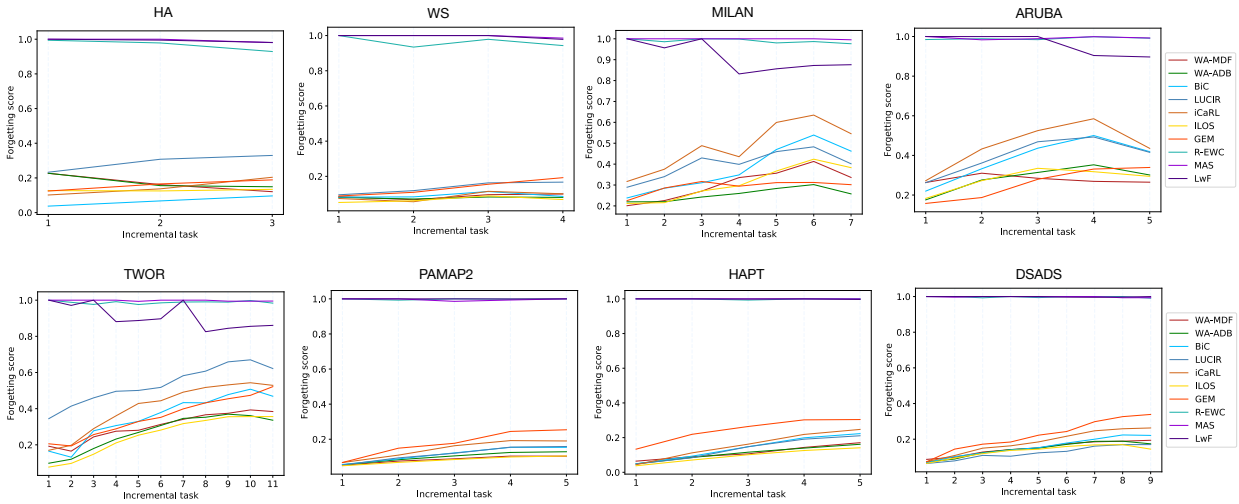


Figure 6: Comparison of forgetting scores on selected techniques

memory replay? If so, which regularisation is more effective in HAR? To answer these two questions, we design the following experiment that uses holdout samples with each regularisation term. More specifically, we look into the following settings: (1) cross-entropy (CE) only (as a baseline without any regularisation) in Eq 4, (2) KD in Eq 1, (3) EWC in Eq 4, (4) MAS in Eq 5, (5) LUCIR discrimination loss (LUCIR-DIS) $\mathcal{L}_{dis}^{G}$ in Eq 10, (6) LUCIR marginal ranking loss (LUCIR-MR) $\mathcal{L}_{mr}$ in Eq 11, (7) LUCIR combination loss (LUCIR) in Eq 12, (8) ILOS cross-entropy loss (ILOS-CE) in Eq 7 and (9) ILOS combination loss (ILOS) in Eq 8. To make a fair comparison, we set up the same setting for each technique, including randomly sampling holdout data from old classes' training data and employing the same training procedure.

Figure 7 compares micro- and macro-F1 scores of different regularisation terms with memory replay. We draw the following observations. Firstly, LUCIR-MR and ILOS produce better accuracy than a simple cross-entropy loss. As shown in Figure 7, ILOS and LUCIR-MR have produced averaged 77% and 61% in micro-F1, which are 13% and 7% higher than the plain CE loss. A possible reason is that LUCIR-MR is dedicated to separating classes and ILOS adjusts the logit output to balance the old and new classes.

Secondly, the other regularisation terms do not improve the results from the CE loss. We can only see 5% and 3% improvement in micro- and macro-F1 from the other regularisation terms over CE in Figure 7. It seems that memory replay and the regularisation terms are dealing with the same problem: *interclass discrepancies*. Therefore, there might not be a distinct advantage for

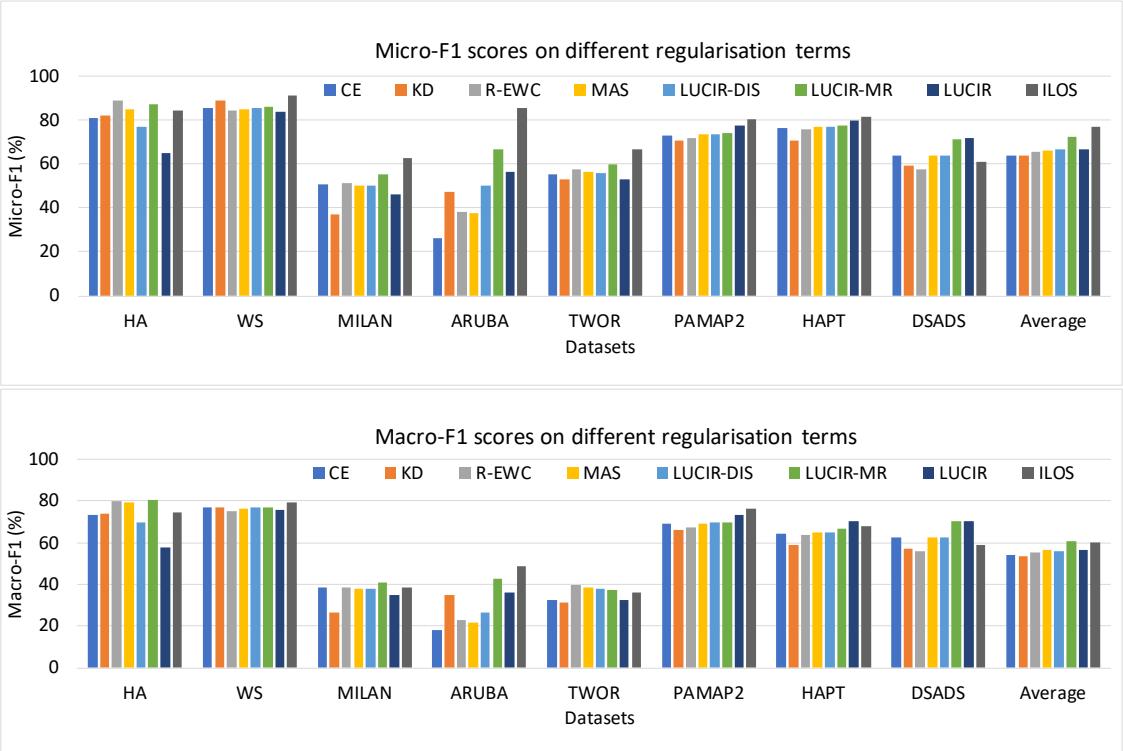


Figure 7: Performance comparison of different regularisation terms with memory replay

regularisation. This finding is consistent with the latest theoretical work [24]. The terms that decrease the difference between micro- and macro-F1 the most are inhibiting the learning outcome if used in conjunction with memory replay. For example, ILOS and R-EWC have the lowest F1 scores with memory replay. Looking back at Figure 5, we can see that ILOS is inhibiting new classes from being learned.

5.4. Effect of Sampling

Since rehearsal methods can improve knowledge retention significantly, now the questions are (1) how many samples are needed to store in memory and (2) what sampling strategy is most effective for selecting these samples that are representative for old classes. To investigate these questions, we experiment different holdout sizes from 2 to 15 with a step size of 2 on widely adopted sampling techniques, including *random* sampling, *herding* [43], *exemplar* sampling, *Frank-Wolfe Sparse Representation* (FWSR) sampling [8], and *boundary* sampling. Herding is to select the top s samples that are the closest to the mean of each class [43]. FWSR sampling selects a subset of the data that effectively describes the entire data set. Exemplar sampling selects the centroid data

points of each class. Boundary sampling [11], a recent sampling technique for incremental learning, selects exemplars on the class decision boundary and overlapping region based on local geometrical and statistical information. Figure 8 compares the micro-F1 scores of sampling techniques on different sample sizes on 4 selected methods and 4 datasets.

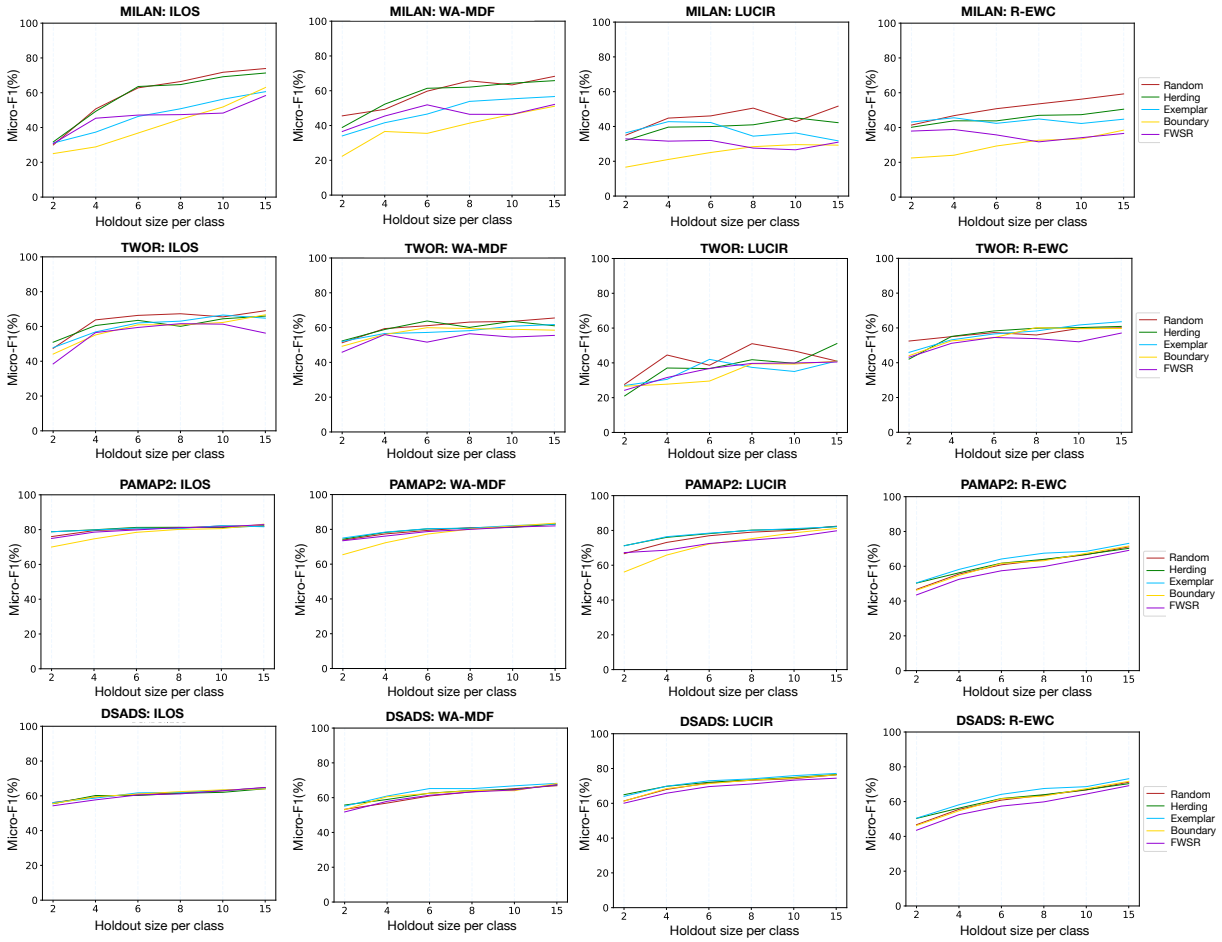


Figure 8: Comparison of accuracy on different sampling techniques and sample sizes.

Holdout size does not impact much on the accuracy in that the accuracy on ILOS, WA-MDF, and LUCIR in Figure 8 does not vary much with the increase of the holdout size. For most of the datasets, the accuracy converges when the holdout size is around 4 or 6.

In terms of the sampling techniques, random (in red), exemplar (in dark green), and herding (in light green) work better than the other two more complex techniques, as presented in Figure 8. Boundary sampling (in orange) works the worst, whose accuracy often stays at the

bottom. Sensor data often contains noise and outliers, so it is difficult to characterise geometric shape or precise class boundary of an activity, and thus both boundary and FWSR do not work well. Exemplar and herding try to capture most representative samples, and work well when the holdout size is small; *i.e.*, 2. Random sampling works generally well and is consistent with the results on images [44], as randomness seems to have better coverage in the entirety of data space with minimal bias for certain data distributions.

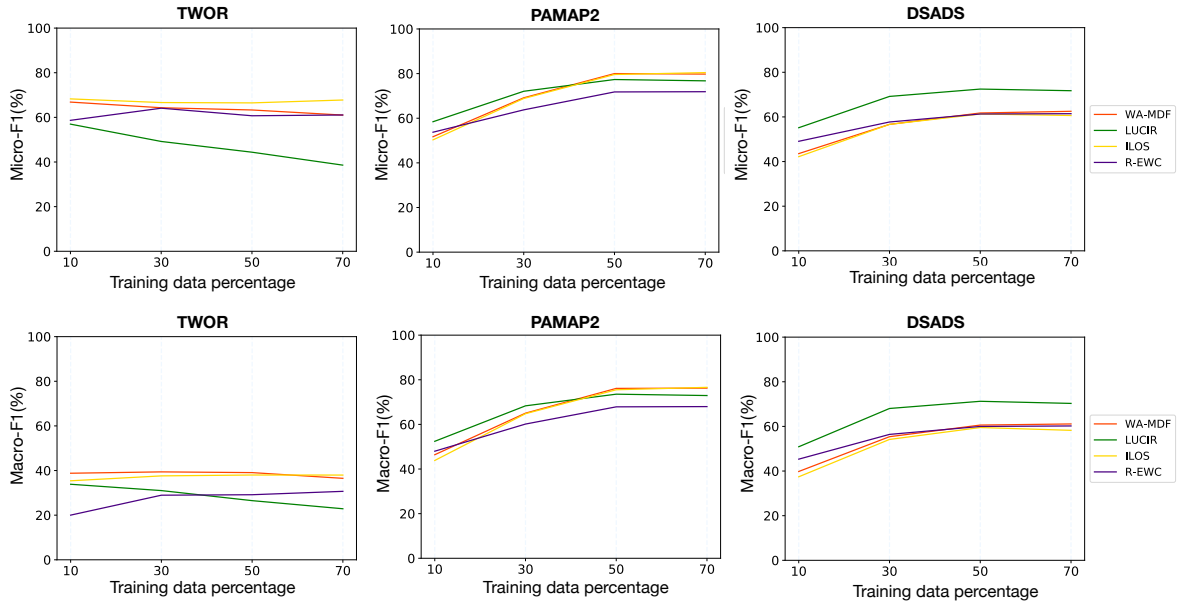


Figure 9: Comparison of accuracy on different training data percentage. The results demonstrate that the selected lifelong learning techniques are insensitive to training data size.

5.5. Effect of Training Data Sizes

Training data for new classes can be more difficult to acquire in a continual learning setting in HAR, as new classes are discovered when the system is already deployed and running. For example, it often relies on users' voluntary self-annotation when the sensor system is deployed in the real world. These experiments are designed to assess the impact of training data size on the selected techniques. We reduce the training data percentage from 70% to 10% with a step size of 20% and report their micro- and macro-F1 in Figure 9. The results show that these techniques are not sensitive to training data size, as the accuracy does not change after 30% training data. For datasets with high imbalance (TWOR), the accuracy of the selected methods does not change, staying at 66% in micro-F1 and 36% in macro-F1. The reason is that some of the classes have

very little samples; e.g., the activity 'R1_bath' only has 19 samples, and the increase in training percentage still does not lead to many samples. We observe that LUCIR decreases accuracy with the increase in training data. The reason is that TWOR has many difficult-to-separate activities, which makes it more challenging to find good anchors when more training data is available.

5.6. Computation Cost Analysis

Computation cost is an important consideration in HAR since devices are usually under memory and computational power constraints. Table 5 presents the computation time of all the selected methods on training a single incremental task and as well as the whole task sequence. For each dataset, we highlight the technique with the longest training time. We report the averaged training time in the last column for the overall comparison. All the training is performed on a modest computer with Intel Core i5 8400, 32GB memory, and 2×500GB SSD.

Table 5: Comparison of computation time (in seconds) of training each incremental (*Incre.*) task and all the tasks (*Total*). (The longest training time is highlighted in bold.)

Model	HA		WS		MILAN		ARUBA		TWOR		PAMAP2		HAPT		DSADS		AVG	
	Incre.	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*	*Incre.*	*Total*
ILOS	2.11	7.28	2.90	13.39	30.07	227.08	47.74	254.17	22.41	260.39	30.58	171.19	84.59	470.80	33.52	33.52	31.74	179.73
WA-MDF	1.98	6.91	2.70	12.58	27.87	211.60	45.44	242.70	21.93	255.23	19.53	112.76	92.23	503.18	32.33	32.33	30.50	172.16
WA-ADB	1.99	6.94	2.72	12.68	29.90	226.34	45.63	243.42	21.30	248.23	21.83	124.60	86.11	482.14	32.43	32.43	30.24	172.10
BiC	2.61	8.74	3.42	15.31	43.92	322.51	53.93	283.09	38.81	439.46	20.77	114.23	81.80	451.57	28.56	28.56	34.23	207.93
GEM	2.55	8.61	3.34	15.14	**67.49**	**488.94**	73.64	383.29	**59.97**	**673.61**	**121.08**	**623.63**	**596.56**	**3050.39**	**116.55**	**116.55**	**130.15**	**670.02**
LUCIR	**3.90**	**12.90**	**5.36**	**23.73**	54.97	406.45	**73.90**	**388.13**	40.54	464.18	35.39	191.95	148.54	818.96	34.53	34.53	49.64	292.60
iCaRL	2.05	7.14	2.61	12.20	28.50	216.18	40.91	219.41	21.36	249.13	30.16	168.22	79.25	445.69	21.14	21.14	28.25	167.39
R-EWC	1.45	5.34	2.00	9.79	29.96	212.40	42.86	229.51	20.41	238.66	23.48	132.11	144.10	774.97	25.38	25.38	36.20	203.52
MAS	1.45	5.32	1.98	9.75	27.78	211.09	42.25	226.46	20.30	237.29	27.38	149.20	133.76	719.23	25.37	25.37	35.03	197.96
LwF	1.27	4.76	1.73	8.67	23.41	179.64	38.00	204.65	17.38	204.53	18.61	108.49	72.82	413.09	18.44	18.44	23.96	142.79
Finetuning	0.93	3.74	1.25	6.78	17.08	135.85	28.61	157.91	12.54	152.03	15.62	92.82	69.19	406.77	11.77	11.77	19.62	120.96

As shown in Table 5, **GEM and LUCIR are the most expensive ones** among the others, especially the incremental training time of GEM (i.e., 130s) is 6 times higher than finetuning on average (i.e., 20s). This is because since the quadratic optimisation in GEM is computationally expensive. LUCIR needs to find good anchors, which incurs an extra cost, so it takes significantly longer training time than the other 8 techniques. The rehearsal based methods take longer to compute than the methods purely based on regularisation. For example, LwF, as the simplest technique, is the least computationally expensive; that is, 20s longer in total training time than finetuning on average. Comparing the weight alignment techniques, BiC is more expensive as it requires to tune the bias correction layer, which happens after a new task has converged. This is slightly more expensive (i.e., 30s more in total training time) than the relatively simple correction methods of WA-MDF/ADB.

Table 6: Summary of selected techniques

Type	Techniques	Key Idea	HAR Limitations			
			Imbalanced Activity Distribution	*Inter-class Similarity*	*Computation Cost*	*Sensitivity to the amount of training data*
Regularisation	**LwF**	Use knowledge distillation (KD) to retain knowledge on old classes	No	No	Low	No
	EWC	Inhibit changes on parameters that are important to old tasks and the importance of parameters is learnt from Fisher Information Matrix	No	No	Low	No
	MAS	Similar to above and the importance of parameters is estimated on their sensitivity to updates	No	No	Low	No
Rehearsal	**iCaRL**	Similar to LwF but store represenative samples from old classes in memory for replay	No	No	Low	No
	ILOS	Adjust norms on old class output in order to put more weight on them	No	No	Medium	No
	GEM	Constrain gradient update to avoid decreasing performance on old classes	No	No	High	No
	LUCIR	Minimise the cosine similarity of features of old and new classes to reduce the impact of updates and apply margin-ranking to enhance in-class separation	Yes	Yes	High	No
	WA-MDF	Align the norm of weight vectors of new classes to those of old classes in order to deal with imbalance	Yes	No	Low	No
	WA-ADB	Re-scale the weight vectors of majority and minority classes	Yes	No	Low	No
	BiC	Introduce a linear layer at the end of the original network to adjust the weight on new and old classes	Yes	No	Medium	No

6. Discussion

This section summarises the above results and provides guidelines on what type of techniques to use under what conditions for HAR systems. Table 6 lists the key ideas of each technique and how well they tackle HAR limitations.

6.1. Imbalanced Activity Distribution

Among the selected techniques, ILOS and WA-ADB produce higher macro-F1 scores than the others on more skewed datasets such as WS, MILAN and ARUBA. WA-ADB accounts for the ratio of training samples in each class at each incremental training step. For ILOS, the averaging of output logits reduces the magnitude of the logits of new dominant classes.

6.2. Inter-class Similarity

Inter-class similarity is the key characteristic of HAR. Some activities can have very similar sensor signatures. For example in the DSADS dataset, the activity classes between 'lying on right' and 'lying on back' exhibit high correlation, which can result in overlapping decision boundaries. LUCIR and GEM have attempted to tackle this problem, and LUCIR outperforms GEM with 6% increase in macro-F1. In addition, LUCIR has achieved the highest accuracy on TWOR dataset that faces the largest challenge of inter-class separation; that is, distinguishing meal preparation from one resident to another. A promising future direction in HAR is to introduce regularisation terms that enhance the discriminability of the network; e.g., the contrastive loss [10] that is dedicated to learning the difference between two sets of data.

6.3. Computation Cost

Except for GEM and LUCIR, the other selected methods take about twice of training time as finetuning. This is acceptable for most modern devices that run human activity recognition. In terms of memory requirements, as the performance of these methods is not sensitive to the holdout size, so we advise a small number (i.e., 2 or 4 data points per class) will be sufficient. Note that most of the regularisation methods require to store the network parameters in memory to compare before and after an update. For a large neural network that consists of a large number of parameters, this might even be considerably more costly than keeping samples. Also, in our experiments, there is no advantage of sophisticated sampling techniques over random sampling of our datasets.

6.4. Scarcity of Labelled Data

The selected techniques are not sensitive to training data size, which is good for incremental learning when labelling new activities is even more difficult. In our experiments, 30% of the training data will be sufficient. However, given a large dataset, 30% still means a large number of samples; e.g., DSADS needs to label 144 samples per class. In the future, we could look into few-shot learning algorithms to further reduce the number of training samples on new classes.

7. Conclusion and Future Work

This paper presents a comprehensive, empirical evaluation of recent continual learning techniques in a task-incremental setting. We seek answers to essential research questions in HAR. We find that rehearsal techniques will lead to the best performance on most of the selected HAR datasets. They can be computationally cheap and do not require much memory space. Sophisticated regularisation terms or gradient updates fall short on their promises. The regularisation terms that help to deal with imbalances and inter-class separation achieve more promising accuracy on HAR datasets. In the following, we will focus on the future developments that HAR techniques have to have.

The state-of-the-art continual learning techniques have paved a promising future for continual learning in HAR. However, most of these techniques are either set in a non-realistic continual learning scenario such as permutation MNIST [39], or in scenarios with distinct task boundaries. We envision a continual learning system in a real-world HAR deployment where new activities can occur spontaneously and interweave with the old activities. Therefore, it needs to be able to

discover new activities first. This requires to combine the existing continual learning techniques with techniques for anomaly detection; i.e., detecting whether the current sensor data conforms to any existing activity pattern. In HAR, new activity discovery has been extensively studied [46]. The question is how to combine these two types of techniques and form a feedback loop, where a new activity is discovered and fed into a network for extension in an automated fashion without any human intervention.

Furthermore, acquiring annotations on new tasks can be challenging, as they rely on user input, which can be unavailable or imprecise. Unlike the existing scenario for most of the continual learning techniques, where the training data is abundant (e.g., 1000 or 2000 examples per class) and well-annotated. In HAR, research interest lies into how to obtain annotations and how to produce robust HAR system in the face of imprecise, insufficient annotation.

Due to sensor degradation, sensor readings will drift over time, and users' behaviour patterns may change due to their health condition. Both will lead to changes in the distributions of the activities over time. This can be considered as *concept drift*, a common problem in streaming data. This adds complication when there is a need for not only learning new tasks but also adapting the model on old tasks. A recent approach that provides a self-constructing methodology to extract hidden layers and neurons from streaming data demonstrates its effectiveness in learning non-stationary data [33].

In the future, we believe that given the promising results on the existing continual learning techniques, a system-level approach for enabling longfor HAR can move on to a system-level approach: how to discover new tasks where there is no clear task boundary, deal with sensor noise and concept drift, and more importantly, tackle noisy and scarce annotations on datasets.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. *ECCV*, 2018.

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR 2017*, pages 7120–7129, 07 2017.

[3] Kerem Altun, Billur Barshan, and Orkun Tunael. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605 – 3620, 2010.

[4] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-

end incremental learning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 241–257, Cham, 2018. Springer International Publishing.

[5] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H.S. Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[6] Arslan Chaudhry, Ranzato Marc'Aurelio, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[7] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv e-prints*, page arXiv:1511.05641, 11 2015.

[8] Gary Cheng, Armin Askari, Kannan Ramchandran, and Laurent El Ghaoui. Greedy Frank-Wolfe Algorithm for Exemplar Selection. *arXiv e-prints*, nov 2018.

[9] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48:480–485, 2009.

[10] Shuyang Dai, Yu Cheng, Yizhe Zhang, Zhe Gan, Jingjing Liu, and Lawrence Carin. Contrastively smoothed class alignment for unsupervised domain adaptation. *ArXiv*, abs/1909.05288, 2019.

[11] S. Dang, Z. Cao, Z. Cui, Y. Pi, and N. Liu. Class boundary exemplar selection based incremental learning for automatic target recognition. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–11, 2020.

[12] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *CVPR 2019*, 2019.

[13] 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.

[14] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *AISTATS 2020*, Palermo, Italy, 2020.

[15] Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *ECCV 2020*, 2020.

[16] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. Incremental learning in online scenario. *ArXiv*, abs/2003.13191, 2020.

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv e-prints*, mar 2015.

[18] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR 2019*, 2019.

[19] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *NeurIPS*, pages 13647–13657, 2019.

[20] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. In *NeurIPS 2020*, 2020.

[21] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI-18*, pages 3390–3398. AAAI Press, 2018.

[22] B. Kim and J. Kim. Adjusting decision boundary for class imbalanced learning. *IEEE Access*, 8:81674–81685,

2020.

[23] 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, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *arXiv e-prints*, dec 2016.

[24] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect memory and is NP-hard. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 5327–5337, Virtual, 13–18 Jul 2020. PMLR.

[25] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

[26] Timothée Lesort, V. Lomonaco, A. Stoian, D. Maltoni, David Filliat, and N. Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Inf. Fusion*, 58:52–68, 2020.

[27] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[28] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. López, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268, 2018.

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc.

[30] Paul Merolla, Rathinakumar Appuswamy, John V. Arthur, Steven K. Esser, and Dharmendra S. Modha. Deep neural networks are robust to weight binarization and other non-linear distortions. *ArXiv*, abs/1606.01981, 2016.

[31] Dariusz Mrozek, Anna Koczur, and Bożena Małysiak-Mrozek. Fall detection in older adults with mobile iot devices and machine learning in the cloud and on the edge. *Information Sciences*, 537:132 – 147, 2020.

[32] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019.

[33] Mahardhika Pratama and Dianhui Wang. Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams. *Information Sciences*, 495:150 – 174, 2019.

[34] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR 2017*, 2017.

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

[36] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert SamÃ , Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754 – 767, 2016.

[37] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[38] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.

[39] van de Ven, Gido M, and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[40] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. *Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software*, pages 165–186. Atlantis Press, Paris, 2011.

[41] Mirko Viroli and Franco Zambonelli. A biochemical approach to adaptive service ecosystems. *Information Sciences*, 180(10):1876–1892, 2010.

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

[43] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 1121–1128, New York, NY, USA, 2009. Association for Computing Machinery.

[44] Junfeng Wen, Yanshuai Cao, and Ruitong Huang. Few-shot self reminder to overcome catastrophic forgetting. *ArXiv*, abs/1812.00543, 2018.

[45] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382, 2019.

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

[47] Juan Ye and Elise Callus. Evolving models for incrementally learning emerging activities. *Journal of Ambient Intelligence and Smart Environments*, 12:313–325, 2020.

[48] Juan Ye, Pakawat Nakwijit, Martin Schiemer, Saurav Jha, and Franco Zambonelli. Continual activity recognition with generative adversarial networks. *ACM Trans. Internet Things*, 2(2), March 2021.

[49] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. ICLR, 2018.

[50] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shutao Xia. Maintaining discrimination and fairness in class incremental learning. *arXiv preprint arXiv:1911.07053*, 2019.