Journal of Ambient Intelligence and Smart Environments 1 (2019) 1–17
IOS Press

Evolving Models for Incrementally Learning Emerging Activities

Juan Ye and Elise Callus [*]

Abstract. Ambient Assisted Living (AAL) systems are increasingly being deployed in real-world environments and for long periods of time. This significantly challenges current approaches that require substantial setup investment and cannot account for frequent, unpredictable changes in human behaviours, health conditions, and sensor deployments. The state-of-the-art methodology in studying human activity recognition is cultivated from short-term lab or testbed experimentation, i.e., relying on well-annotated sensor data and assuming no change in activity models. This paper propose a technique, *EMILEA*, to evolve an activity model over time with new types of activities. This technique novelly integrates two recent advances in continual learning: Net2Net – expanding the architecture of a model while transferring the knowledge from the previous model to the new model and Gradient Episodic Memory – controlling the update on the model parameters to maintain the performance on recognising previously learnt activities. This technique has been evaluated on two real-world, third-party, datasets and demonstrated promising results on enhancing the learning capacity to accommodate new activities that are incrementally introduced to the model while not compromising the accuracy on old activities.

Keywords: Activity recognition, Continual learning, Smart home

1. Introduction

Ambient Assisted Living (AAL) refers to sensing, communication, and intelligence technologies deployed in living environments with the aim to improve quality of life [22]. Recently AAL has made great progress through the use of emerging sensing, machine learning (esp. deep learning), and robotic technologies. It spans a wide range of applications and we take an example in personal healthcare in smart home environments. An environment can be deployed with passive infrared motion sensors to track users' whereabout, RFID sensors to detect users' interactions with everyday objects, and resource monitoring sensors to monitor consumption of water, electricity, and gas. These sensor data will be collected and analysed to predict users' activities, which can be further used to health tracking and disease diagnosis.

With the existing AAL systems, we can monitor and recognise people's daily activities [32], track their health [30], and provide assistance with their completion of daily activities [27]. This success is enabling the move towards large-scale, in-the-wild, and long-term deployment of AAL systems. This move however comes with its own challenges, notably that neither the sensing technologies being deployed, nor people's activity routines or health conditions, remain constant. This creates a need for continual learning in AAL systems. Continual learning is a subfield in machine learning, referred to as the ability to continually learn over time by accommodating new knowledge while retaining previous knowledge [20].

Most of the existing activity models that are built on supervised learning classifiers do not support this ability, as they need to retrain their models with all the data. For example, a system may collect a collection of sensor data on two activities such as 'watch TV' and 'sleep', where a classifier is trained on these data to be able to recognise them. Then after a while, the model might need to be extended to recognise a new activity 'do rehabilitation exercise'. Then either the classifier needs to re-train the model with all the data on these three activities, which can be undesirable if the num-

[*] Ye et al. are in the School of Computer Science, University of St Andrews, UK.
jy31@st-andrews.ac.uk

ber of activities accumulates to a large number [1]. Alternatively, we might use an updatable classifier so that new instances can be used to incrementally and iteratively train the model, but the problem is that the classifier might suffer from *catastrophic forgetting* – the performance on recognising previous activities might be compromised by the update [19].

In this paper, we present *EMILEA* to evolve an activity model incrementally with new types of activities, which is built on recent advance in continual learning – Net2Net [8] and Gradient Episodic Memory (GEM) [19]. The former provides operations to extend the architecture of a neural network iteratively to enhance the learning capacity on an increasing number of activities. The latter mitigates the effect of catastrophic forgetting by controlling the gradient update while taking into account of the activities that have been learnt before. We conduct a comprehensive evaluation of the proposed technique on two real-world datasets to assess the strength and limitation of *EMILEA*, which sheds light on the future design of continual learning techniques for human activity recognition.

The rest of the paper is structured as follows. Section 2 reviews the literature of evolving activity models in human activity recognition and continual learning. Section 3 describes the approach including problem definition, workflow and the key components. Section 4 introduces the evaluation methodology and Section 5 presents the results and discusses the limitation of *EMILEA*. The paper concludes in Section 6.

2. Related Work

In this section, we will review recent work on discovering and recognising new activities with a particular focus on how to evolve the models and also briefly look into continual learning techniques in the field of machine learning.

2.1. New Activity Discovery and Recognition.

In recent years, there is an increasing number of work devoted to discovering and recognising new activities. Clustering and one-class classifier are the most popular approaches. The idea is to add new clusters and classifiers for each new type of activities. Gjoreski et al. have used an agglomerative clustering technique to enable real-time clustering of streaming data [13]. To validate clusters for new activities, they have proposed two temporal assumptions on human activities; that is, a human activity usually lasts for a certain period of time and there should not be frequent transitions between activities. With these assumptions, they have filtered short outliers and been able to more accurately discover meaningful clusters.

Ye et al. use distance-based clustering to incrementally learn and recognise new daily routine activities such as preparing breakfast or sleeping from binary sensors embedded in a smart home [31]. An activity profile is built on top of each pre-defined activity using training data and is modelled as a cluster. Mathematically proved sufficient statistics are summarised on each cluster in order to enable model drift without the need of storing any historical sensor data. Then each incoming sensor data will be assessed on each activity profile, and if the sensor data does not match any existing activity profile; *i.e.*, not falling into the corresponding cluster, then it is considered as abnormal and stored in a candidate pool. A clustering technique is consistently running on the candidate pool to identify converged clusters whose sufficient statistics do not significantly change. Once identified, the centre node of the cluster is taken for annotation query and an activity profile is built on this new cluster.

Shin et al. use Support Vector Data Description (SVDD) with a Gaussian kernel to detect abnormal activities of elder people, such as weakness or fall, based on features extracted from infra-red motion sensor data collected in houses [26]. The idea is to form a hypersphere that encompasses all positive instances with the minimal volume. An anomalous instance is the data point that falls out of the hypersphere.

For clustering techniques and one-class classifiers, it is less a problem that a new activity is just another cluster(s) or another classifier. But simply adding a new cluster or a new one-class classifier will make the model fragile; for example, there might be overlapping between clusters, which often needs to re-build. To tackle this issue, Fang et al. have proposed a hierarchical mixture model where each sub-model, built on a conditional independent von Mises-Fisher distribution, corresponds to a type of activity [12]. When a new activity is discovered, a sub-model will be created and added to the hierarchy. Then the contributing parameters on each sub-model will be updated.

Cheng et al. have adapted a zero-shot learner to recognise a new activity with limited training data [11]. A knowledge-driven model encodes the semantic relationship between high-level activities and low-level sensor attributes generated from accelerometer data. To include a new activity, domain experts and developers need to manually add new attributes and update the activity-attribute matrix with manually specified relationship between attributes and this new activity.

2.2. *Continual learning*

In this section, we describe recent advance in continual learning in the field of machine learning, including regularisation and dynamic approaches and memory replay, that can be applied to mitigate this problem [20].

2.2.1. *Dynamic Architectures*

Approaches to alleviate forgetting include changing the architecture of the network when new information is received. The model consisting of a different number of neurons or layers from the previous model is then retrained.

Rusu et al. [25] have introduced a *progressive network* that stops any changes on the network and expands it by adding a new sub-network for the new data. The main idea is to keep a pool of models that are pretrained with previous knowledge and add lateral connections to them for the new task [9]. To mitigate forgetting, the parameters (θ_N) for previous tasks N are never modified while the new parameters (θ_{N+1}) are learned for the new task *N+1* [20]. Furthermore, this does not deteriorate the performance of previous tasks. One drawback of using this technique is that the network can become complex with the increasing number of tasks learned. Since a new network is learned for each task and it is connected to the previous network, the complexity of the network structure and parameters will increase very quickly [9, 20].

Aljundi et al. [1] introduce *ExpertGate* that consists of a network of experts where each *expert* is a model trained on a specific task. The new tasks are added to the previously trained models in a sequential order in which the knowledge is transferred. A gating mechanism is built to decide which expert is required for activation. This removes the requirement of loading all models which is memory efficient as each model can be computationally intensive [9].

Chen et al. [8] propose a Net2Net approach where the networks can be widen (adding more neurons) and deepen (adding more layers), and knowledge from the previous network will be transferred or preserved in the newly constructed network.

2.2.2. *Regularisation Approaches*

This subsection introduces approaches which enforce constraints when the neurons' weights are updated such as Learning without Forgetting technique and the Elastic Weight Consolidation approach.

Li and Hoiem [18] propose an approach called *Learning without Forgetting* (LwF), where the shared parameters (θ_s) and the parameters of the old tasks (θ_o) influence how the model learns parameters for the new task (θ_n). The latter parameters are updated in a way that do not drastically decrease the performance on the old tasks. LwF has faster learning rates when compared to joint training [6] as it does not need access to training and testing data for previous tasks. One of the advantages of this approach is that the old training data does not need to be retained as it will not be used to re-adjust the network [18]. However, one drawback is that if the tasks are very different, performance may decrease.

Kirkpatrick et al. [16] have proposed *Elastic Weight Consolidation* (EWC) to slow the training of the weights related to the tasks so that the expertise on old tasks can be retained. EWC is evaluated on the MNIST dataset [17] where the new task consists of a modification in the order of the input pixels of the images in the dataset. The results are promising and indicate that EWC can perform well on models that have the catastrophic forgetting limitation [9].

Another model called Gradient Episodic Memory (GEM), introduced by Lopez-Paz et al. [19], supports continual learning by using an episodic memory and by supporting backward transfer of important knowledge to previous tasks. The episodic memory stores a subset of data from the previous task to prevent the GEM model from increasing the loss on previous tasks when training of the current task. Parisi et al. [20] discuss how more memory is required during training for the GEM when compared to other regularisation techniques such as EWC.

Rebuffi et al. [23] have proposed *incremental Classifier and Representation Learning* (iCaRL), which makes the use of stored exemplars for the old tasks. Examples represent the most important information on the tasks and for each task that is introduced, a set of exemplars dedicated to that particular class is created [9]. iCaRL classifies the new sample into a class based on which class that has the most similar exemplars to it [9]. iCaRL also replays the stored data during training which mitigates forgetting [28]. In this approach, resources are slowly increased with the number of classes introduced [9].

EMILEA is built on two of the above techniques – Net2Net and GEM, with stored examples from learnt classes, borrowed from iCaRL. This allows the ex-

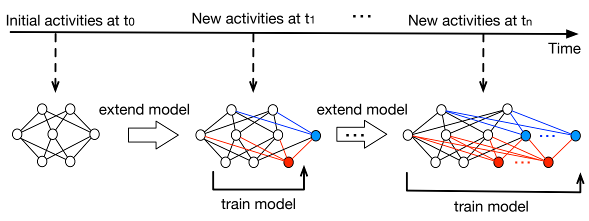

Fig. 1. *EMILEA* workflow

pansion of the network to learn more activities while at the same time mitigating the catastrophic forgetting effect by controlling the gradient updates with the stored examples. Net2Net and GEM target continual learning from different aspects. GEM updates parameters for the original network to optimise learning towards new classes while not reducing the loss on old classes, while Net2Net creates networks having a different structure from the original. *EMILEA* novelly integrates these two together to expand the network while adjusting the parameters at the same time.

3. Proposed Approach

We define *continual learning* in activity recognition as continually and incrementally learning activities in a sequential manner. Let $D_{t_n} = (\{(x_i, y_i)\}_{i=1}^{N_{t_n}}, C_{t_n})$ be training data arriving at a time t_n, where each example (x_i, y_i) is composed of a feature vector $x_i \in \mathcal{X}$, and y_i is an activity label $y_i \in C_{t_n}$. C_{t_n} is a set of new activities available at the time t_n and $C_{t_n} \cap (C_{t_{n-1}} \cup C_{t_{n-2}} \cup ... \cup C_{t_0}) = \emptyset$; that is, the new activities have not been observed in the previous training data. The goal of *EMILEA* is to learn and extend an activity model $f_{t_n} : \mathcal{X} \to C$, where $C = C_{t_n} \cup C_{t_{n-1}} \cup C_{t_{n-1}} \cup ... \cup C_{t_0}$.

Figure 1 presents the workflow of *EMILEA*. It starts at the time t_0 with a base set of activities, and *EMILEA* constructs a model f_{t_0} to recognise the activities in C_{t_0}. When there arrives a set of new activities C_{t_1}, then the model f_{t_0} will be expanded to accommodate the classification capability on both C_{t_0} and C_{t_1}. This process will be repeated whenever there are new activities to learn.

In this process, we need to address two questions: (1) how to extend the model f_{t_n} and (2) how to prevent *catastrophic forgetting* – a classic problem in continual learning. For question (1), we want to transfer the knowledge from the previous model $f_{t_{n-1}}$ such that the classification capability on $C_{t_{n-1}}$ is preserved and learning at the time t_n can focus on new activities, rather than learning all the activities from scratch. For question (2), catastrophic forgetting is the phenomena where learning new classes may compromise the performance on the previous classes [15]. Figure 2 illustrates the catastrophic forgetting problem; that is, when the model is extended and trained with new activities, the model is optimised to recognise new activities only, so the accuracy on the new activities is high (nearly 100%) while the accuracy on the old classes stays low (less than 1%).

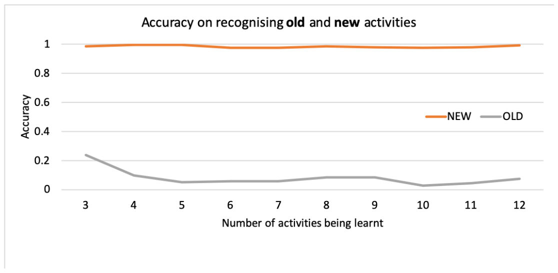

Fig. 2. An example to illustrate catastrophic forgetting

To tackle the above two questions, we novelly integrate two advanced continual learning techniques from the machine learning community: *Net2Net* [8] and *Gradient Episodic Memory* (GEM) [19]. The Net2Net introduces operations to extend a network with more neurons and layers to enhance the learning capacity in order to classify an increasing number of classes. GEM alleviates the forgetting effect by controlling the gradient updates to balance the performance on old and new classes. In the following, we will give a detailed description on these two techniques and present an algorithm on integrating them to tackle continual learning in activity recognition.

3.1. *Model Extension*

Net2Net introduces two operations to extend the network: *Net2WiderNet* – adding neurons to the hidden layer and *Net2DeeperNet* – adding layers to the network. Here we focus on *Net2WiderNet* as the sensor data are often simpler (i.e., with less dimensions and correlations between dimensions) than images. For simplicity, we call the *Net2WiderNet* as *Net2Net*.

The principle of Net2Net is to add neurons to a hidden layer and redistribute the weights and biases of that layer and the layer after. Let layer l and $l+1$ be fully connected layers, and layer l has m inputs and p outputs, and layer $l+1$ has p inputs and n outputs. Assume that we will extend the layer l with q new neurons, we will need to update the weights $\mathbf{W}^{(l)}(\in \mathbb{R}^{m\times p})$ and $\mathbf{W}^{(l+1)}(\in (\mathbb{R}^{p\times n}))$. The principle behind Net2Net is *function preserving* – leveraging the functions or knowledge in the previous model so that the model still can recognise the old classes. To achieve this, the model will be extended with the existing neurons and the new weights on both old and new neurons will be initialised with the original weights. This is in comparison to a non-preserving approach where a new set of neurons will be introduced and the weights on them will be randomly initialised.

It starts with randomly sampling q neurons from the original p neurons at layer l:

$$g(j) = \begin{cases} j & j \leqslant p \\ \text{random sample from}\{1, 2, ..., p\} & j > p \end{cases}$$

Then the new weight matrix $\mathbf{U}^{(l)}(\in \mathbb{R}^{m\times(p+q)})$ is constructed by firstly copying all the original weights $\mathbf{W}^{(l)}$ to $\mathbf{U}^{(l)}$ and then copying the weights on the sampled neurons to the new q neurons. For the weight matrix $\mathbf{U}^{(l+1)}(\in \mathbb{R}^{(p+q)\times n})$, it starts with the same process as for $\mathbf{U}^{(l)}$ but accounts for the replication by dividing the weights on the replicated neurons.

$$\mathbf{U}^{(l)}_{k,j} = \mathbf{W}^{(i)}_{k,g(j)} \tag{1}$$

$$\mathbf{U}^{(l+1)}_{j,h} = \frac{1}{|\{x|g(x) = g(j)\}|}\mathbf{W}^{(l+1)}_{g(j),h} \tag{2}$$

where $k \in [1, m]$, $j \in [1, p + q]$, and $h \in [1, n]$. After the replication process, a small amount of noise will be added to break the symmetry.

The extension of the model often starts from the last layer (the output layer) to accommodate new classes and/or from the second last layer to enhance the learning capability to discriminate a larger set of classes. Net2Net supports expanding the model at multiple layers. In this situation, the expansion will start from the second last layer and gradually move forward to the previous layers. The weights initialisation will be done iteratively layer after layer. Algorithm 1 illustrates the process.

Algorithm 1 Neural network expansion

```
Input: $N_E$ – a list consisting of the number of neurons
       to extend at each layer, starting from the second
       last layer and moving forward layer by layer
Input: $\mathbf{W}$ – a list consisting of the weight matrices
       from the previous model, whose layers map to
       the layers in $N_E$
for $l \in 1, 2, ..., |N_E|$ do
    generate a mapping function $g^{(l)}$ :
      $\{1, 2, ..., |\mathbf{W}^{(l)}|[1]\} \rightarrow \{1, 2, ..., |\mathbf{W}^{(l)}|[1] + N_E[l]\}$
    $c_j \leftarrow 0$ for $j \in 1, 2, ..|\mathbf{W}^{(l)}|[1] + N_E[l]$ do
        $c_{g^{(i-1)}(j)} \leftarrow c_{g^{(i-1)}(j)} + 1$
    end
    for $j \in 1, 2, ..., |\mathbf{W}^{(l)}|[1] + N_E[l]$ do
        $\mathbf{U}^{(l)}_{k,j} \leftarrow \frac{1}{c_j}\mathbf{W}^{(l)}_{g^{(i-1)}(k),g^i(j)}$
    end
end
Output: $\{\mathbf{U}\}$: the transformed weight matrices for a
        wider network
```

3.2. Forgetting Effect Mitigation

Now we will describe how to use GEM to mitigate the forgetting effect. GEM assumes a memory space to host examples from the previous old classes. Learning new classes is to minimise the loss function on both old and new classes;

$$\mathcal{L}(f^\theta, \mathcal{M}_k) = \frac{1}{|\mathcal{M}_k|}\sum_{(x_i,k,y_i)\in\mathcal{M}_k} \mathcal{L}(f^\theta(x_i, k), y_i) \tag{3}$$

where $\mathcal{M}_k$ holds the examples on the old classes in the previous task arriving at the time k ($< t_n$) in the memory and θ is the network parameters, such as weights and biases.

To prevent forgetting, GEM guarantees that the loss at previous tasks does not increase after each parameter update. That is, when observing a new training example (x, t_n, y) at the current time t_n,

$$\begin{aligned} &\text{minimize}_\theta\ \mathcal{L}(f^\theta(x, t_n), y) \\ &\text{subject to } \mathcal{L}(f^\theta_{t_n}, \mathcal{M}_k) \leqslant \mathcal{L}(f^\theta_{t_{n-1}}, \mathcal{M}_k) \\ &\ \text{for all } k < t_n, \end{aligned} \tag{4}$$

To assess whether the new update will increase the loss or not, GEM leverages the examples held in the memory by computing the angle between their loss gradient vector and the proposed update; that is, the above equation (4) will be re-phrased as the following:

$$\langle g, g_k \rangle := \left\langle \frac{\partial \mathcal{L}(f^{\theta}(x, t_n), y)}{\partial \theta}, \frac{\partial \mathcal{L}(f^{\theta}, \mathcal{M}_k)}{\partial \theta} \right\rangle \geqslant 0 \quad (5)$$

$$\text{for all } k < t_n$$

If all the inequality constraints in the equation (5) are satisfied, then the proposed gradient update g is unlikely to increase the loss on previous classes. Otherwise, there is at least one previous class that would experience an increase in loss after the update. In this situation, the proposed gradient g will be projected to the closest gradient $\tilde{g}$ that satisfy the constraints in Equation (5). That is,

$$\begin{array}{l} \text{minimize}_{\tilde{g}} \frac{1}{2}||g - \tilde{g}||_2^2 \\ \text{subject to } \langle \tilde{g}, g_k \rangle \geqslant 0 \text{ for all } k < t_n. \end{array} \quad (6)$$

The primal of a Quadratic Program (QP) with inequality constraints is applied to solve the above equation, which is described as:

$$\begin{array}{l} \text{minimize}_{v} \ \frac{1}{2} v^T G G^T v + g^T G^T v \\ \text{subject to } v \geqslant 0, \end{array} \quad (7)$$

where $g^T g$ is constant and $G = -(g_{t_0}, g_{t_1}, ..., g_{t_{n-1}})$ [19]. This equation is a QP calculated on the number of classes observed so far. Once the dual problem (Equation (7)) is solved for v^*, the projected gradient update is calculated using $\hat{g} = G^T v^* + g$ [19]. The calculated gradient is then applied using the optimiser.

The overall algorithm of *EMILEA* is presented in Algorithm 2.

Algorithm 2 *EMILEA*

```
Input: $D_{t_0}$ – training data on the first batch of activities $C_{t_0}$
Input: $N_E$ – a list consisting of the number of neurons to extend at each layer, starting from the second last layer and moving forward layer by layer
Input: W – a list consisting of the weight matrices from the previous model, whose layers map to the layers in $N_E$
$m \leftarrow$ build_model train(m) $\mathcal{M} = \{\}$
while new training data $D_{t_n}$ arrives do
    m $\leftarrow$ extend_model(m, $N_E, W$)
    for epoch training_epoch do
        for batch training_batches do
            $g \leftarrow \bigtriangledown_{\theta}\ l(f_{t_n}^{\theta}(batch), y)$
            $g_k \leftarrow \bigtriangledown_{\theta}\ l(f_{t_n}^{\theta}, M_k)$ for all $k < t_n$
            $\hat{g} \leftarrow project(g, g_{t_0}, g_{t_1}, ...g_{t_n-1})$
            $\theta \leftarrow \theta - \alpha\hat{g}$
        end
    end
    $\mathcal{M} \leftarrow \mathcal{M} \cup samples(D_{t_n})$
end
```

4. Experiment and Evaluation Methodology

The objective of *EMILEA* is to assess the performance of recognising both old and new activities over time by incrementally introducing new activities.

4.1. Evaluation Process

The evaluation process works as follows. *EMILEA* is initially trained on randomly sampled 2 activities with 50% of their training data. Then we gradually extend the model with a new activity a time, which is randomly sampled from the remaining set of activities. For all the learnt activities, we hold out $p\%$ of the training data for retraining the model. When a new activity is introduced, the network will be expanded and re-initialised with the previous model's weights and biases. Then it will be trained with the new class' training data and the holdout data from the old classes.

At each expansion, we evaluate four types of accuracy:

– *New* – accuracy on the test data of the new activities C_{t_n}, which is lastly trained on;
– *Old* – accuracy on the test data of the old activities that have been learnt $C_{t_{n-1}} \cup C_{t_{n-2}} \cup ... \cup C_{t_0}$;
– *All* – accuracy on the test data of the old and new activities;
– *Base* – accuracy on the test data of the initially sampled activities C_{t_0}.

4.2. Comparison Techniques

We consider to compare with the baseline approaches and also variations of configurations in *EMILEA*. More specifically,

– **Offline** – train a model with all the activities (with 50% training percentage);
– **Naive** – set up model the same as **Offline** with the output layer matching to all the activities and train the model with adding an activity a time (again with 50% training percentage);

- **GEM** – set up the model the same as **Naive** but apply the GEM approach to update the gradients with holdout data, which is $p\%$ of training data on each previous activity.
- **Net2Net** – set up the model the same as **Naive** with the output layer only containing the selected base activities (which are 2), and then gradually extend the network architecture. Here we consider only extending the last output layer and the second last hidden layer with a different number of neurons. Also the holdout data is used together with the training data on new activities to retrain the model.
- **Net2Net+GEM** – the same as **Net2Net** but apply GEM to update the gradient.

4.3. *Selected Datasets*

We consider two publicly available, third-party datasets for evaluating the performance of *EMILEA*. The first dataset is *PAMAP2* – Physical Activity Monitoring Dataset [24]. It contains 12 activities, including lying, standing, sitting, ironing, and house cleaning. These activities and their distribution are recorded in Figure 3. The sensor data are collected on 9 subjects with 3 initial measurement units on each subject's dominant arm, chest, and dominant side ankle.

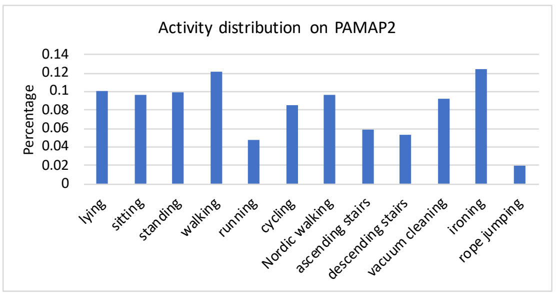

Fig. 3. Activity Distribution on *PAMAP2*

The other dataset is *DSADS* – Daily and Sports Activities Dataset [2–4]. It contains 19 activities, including sitting, running on a treadmill, exercising on a stepper, and rowing. Each of these activities is performed by 8 subjects for 5 minutes. The sensor data are collected on 8 subjects with 5 accelerometer units on each subject's torso, right arm, left arm, right leg, left leg.

As the paper does not aim for feature extraction, we do not work on the raw accelerometer data on these two datasets, but on the extracted feature datasets [29]. For each sensor, 27 features are exacted, including mean, standard deviation, and correlations on axises. The statistics of these two datasets are listed in Table 1. Both datasets are ideal for validating *EMILEA* as they contain a large number of activities and have a high-dimensional feature space, which increases the challenge of continual learning.

Table 1
Dataset Description

Dataset	No. Samples	No. Activities	No. Features
DSADS	9120	19	405
PAMAP2	7312	12	243

4.4. *Model Configuration and Parameter Training*

As *EMILEA* will expand the network with continually increased activity classes, the initial configuration of the network should be small to avoid overfitting and reduce computational cost [8]. To decide the initial configuration, we consider to use two designs with 2 and 3 layers respectively. With respect to the numbers of neurons, we have run grid search with different numbers of neurons and choose the architecture that achieves the best accuracy on the randomly selected 2 base activities. In the end, we have settled the model with 2 hidden layers with 20 neurons and 40 neurons per hidden layer for the PAMAP2 and DSADS datasets respectively.

We set the learning rate as 0.001 and the batch size as 16. The learning rate is chosen using a grid parameter search to decide the best accuracy while also decreasing the cost. We start from 0.005 to 0.001 with a step size 0.001, and the setting is chosen because the model is already initialised with the previous model's weights and the gradient update will benefit from a small learning rate.

In order to determine the number of training epochs, we run an experiment to see at which epoch the accuracy averaged on the activities that have been learnt so far will stabilise. We use the setting of Net2Net with GEM and a holdout percentage of 5%. For each new activity, we train the model with 20 epochs. Figure 4 presents the accuracy on old, new and all activities over time on the PAMAP2 dataset. As we can see, for majority of activities, the accuracy on all the activities will stabilise after 10 epochs. We can also observe that the accuracy on new activities is often high but occasionally reduces significantly (e.g., on the 8th activity). This is caused by the activity variability between the current and previous learnt activities. The order of learning new classes matters in continual learning. Ideally, the model can gain higher performance if it trains with easier classes and then gradually learns classes with increasing difficulty level [5, 10].

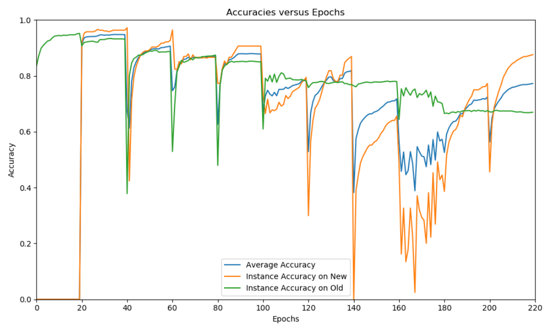

Fig. 4. Accuracy on old, new, and all activities over time on introducing a new activity a time on PAMAP2

5. Results and Discussion

This section will present the results and discuss the limitation of *EMILEA*.

Table 2

Summary result on PAMAP2

Model	Holdout Percentage	Neuron Expansion	Accuracy		
			Base	New	All
Offline			0.78 (0.07)		
Naïve			0.04 (0.05)	0.94 (0.10)	0.22 (0.04)
GEM	0.01		0.5 (0.04)	**0.75 (0.02)**	0.55 (0.02)
	0.05		**0.58 (0.06)**	0.6 (0.01)	**0.65 (0.01)**
Net2Net	0.01	0	0.25 (0.05)	0.98 (0.01)	0.36 (0.04)
		1	0.28 (0.15)	0.98 (0.01)	0.35 (0.07)
		2	0.26 (0.14)	0.98 (0.01)	0.33 (0.05)
		3	**0.32 (0.15)**	**0.98 (0.01)**	**0.37 (0.07)**
	0.05	0	0.50 (0.11)	0.93 (0.02)	**0.60 (0.08)**
		1	0.51 (0.11)	0.93 (0.04)	0.58 (0.04)
		2	0.42 (0.18)	0.93 (0.03)	0.53 (0.08)
		3	**0.55 (0.12)**	**0.96 (0.01)**	0.56 (0.04)
Net2Net +GEM	0.01	0	0.54 (0.15)	0.46 (0.19)	0.42 (0.11)
		1	0.54 (0.06)	0.43 (0.20)	0.41 (0.09)
		2	0.53 (0.10)	**0.81 (0.08)**	0.54 (0.06)
		3	**0.63 (0.06)**	0.78 (0.20)	**0.67 (0.19)**
	0.05	0	**0.70 (0.08)**	0.67 (0.09)	0.73 (0.07)
		1	0.64 (0.03)	0.77 (0.08)	0.73 (0.09)
		2	0.67 (0.12)	**0.79 (0.06)**	**0.74 (0.05)**
		3	0.64 (0.11)	0.75 (0.11)	0.73 (0.07)

Table 3

Summary result on DSADS

Model	Holdout Percentage	Neuron Expansion	Accuracy		
			Base	New	All
Offline			0.68 (0.13)		
Naïve			0.06 (0.04)	0.99 (0.00)	0.19 (0.03)
GEM	0.01		0.33 (0.12)	**0.86 (0.02)**	0.41 (0.02)
	0.05		**0.65 (0.14)**	0.65 (0.05)	**0.6 (0.03)**
Net2Net	0.01	0	0.33 (0.03)	0.98 (0.01)	0.44 (0.05)
		2	0.31 (0.06)	0.96 (0.02)	0.42 (0.01)
		4	0.32 (0.11)	0.97 (0.03)	**0.45 (0.07)**
		6	0.22 (0.08)	**0.99 (0.00)**	0.39 (0.03)
		8	**0.38 (0.12)**	0.98 (0.01)	0.43 (0.04)
	0.05	0	0.65 (0.12)	0.94 (0.05)	0.67 (0.04)
		2	0.63 (0.09)	0.95 (0.03)	0.68 (0.07)
		4	0.63 (0.11)	**0.95 (0.02)**	**0.70 (0.07)**
		6	**0.75 (0.13)**	0.94 (0.03)	0.68 (0.05)
		8	0.74 (0.10)	0.92 (0.05)	0.68 (0.06)
Net2Net +GEM	0.01	0	0.31 (0.09)	0.92 (0.02)	0.59 (0.02)
		2	0.31(0.10)	0.92 (0.03)	0.55 (0.05)
		4	0.41 (0.16)	0.92 (0.05)	0.57 (0.05)
		6	**0.44 (0.17)**	**0.92 (0.01)**	**0.59 (0.04)**
		8	0.42 (0.12)	0.90 (0.05)	0.57 (0.03)
	0.05	0	0.69 (0.15)	0.75 (0.05)	0.72 (0.04)
		2	0.73 (0.06)	0.77 (0.01)	0.75 (0.04)
		4	0.72 (0.03)	0.76 (0.03)	0.74 (0.04)
		6	**0.74 (0.09)**	**0.77 (0.05)**	**0.76 (0.03)**
		8	0.67 (0.15)	0.74 (0.05)	0.75 (0.04)

Table 2 and 3 presents the comparison of accuracy with different variants of design and baseline approaches on the PAMAP2 and DSADS datasets. We run each setting 10 times and present the mean and the standard deviation for each type of accuracy. We consider holdout percentages as 1% and 5%, which are set low to reduce the memory cost. On the PAMAP2 dataset, we expand the number of neurons from 1 to 3. Because the initial network architecture on DSADS dataset is larger, we increase the number of neurons from 2 to 8 in order to achieve observable enhancement on learning capacity.

5.1. *Performance on Recognising Base Activities*

The *Base* accuracy measures the accuracy on the first input activities C_{t_0}, which is used to assess the forgetting effect. The combination of Net2Net and GEM have achieved the best accuracy. On the PAMAP2 dataset, with 1% holdout percentage (which corresponds to 3 examples per activity type), Net2Net+GEM achieves the accuracy of 63%, which is 31%, 13%, and 59% higher than the Net2Net, GEM, and Naive approaches. GEM is the second best performing approach, 18% and 46% higher than Net2Net and Naive. This shows that GEM helps maintain the accuracy on the activities that have been trained long ago. Net2Net that extends the network architecture does improve the learning capacity to accommodate an increasing number of activities. However, without GEM, with a small number of holdout examples, Net2Net still suffers the forgetting effect. The reason is that the holdout data is only 1% or 5% of training data, which is significantly less than the training data on the new activities.

Figure 5 presents the trend of base accuracy (with 1% holdout percentage) over time on the PAMAP2

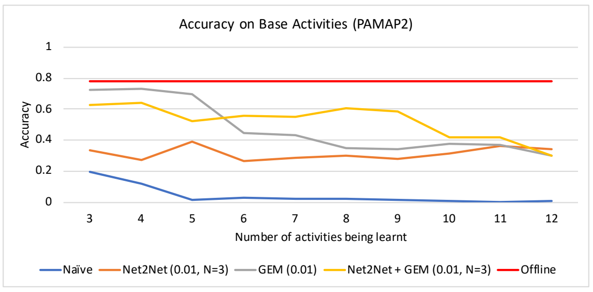

Fig. 5. Accuracy on recognising *base* activities on PAMAP2

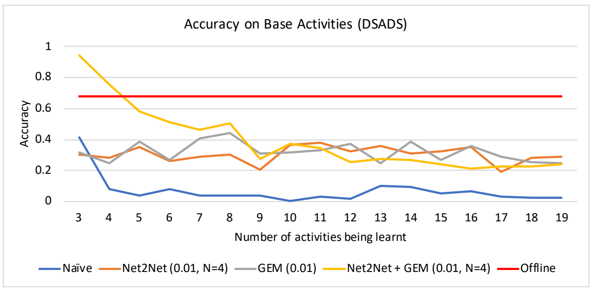

Fig. 6. Accuracy on recognising *base* activities on DSADS

dataset. Net2Net+GEM has maintained more consistent performance after adding 9 activities. GEM's performance drops after 5 activities. The base accuracy on Net2Net stays low; i.e., between 20% and 40%. Clearly Naive has no capability of recognising any base activities at all.

When the holdout percentage increases to 5%, the gap of the base accuracy between GEM and Net2Net reduces; with GEM is 3% higher than Net2Net. Net2Net+GEM still achieves the best accuracy 70%.

On the DSADS dataset, we can observe the similar benefit of the combination of Net2Net and GEM. With 1% holdout data (which corresponds to 2 examples per activity type), Net2Net+GEM achieves the base accuracy of 47%, which is 9%, 14%, and 41% higher than Net2Net, GEM, and Naive approach. With 5% holdout data, Net2Net+GEM achieves the base accuracy of 77%, which is 2%, 12% and 71% higher than Net2Net, GEM, and Naive approach.

The increase on holdout data (from 1% to 5%) has more significant impact on the DSADS dataset than on the PAMAP2 dataset, which leads to 30%, 37%, and 32% increment on base accuracy of Net2Net+GEM, Net2Net, and GEM. The reason behind is that DSADS has more activity types (19 in DSADS and 12 in PAMAP2) and the difficulty level of discriminating these activities is also higher (68% of offline accuracy on DSADS and 78% of offline accuracy on PAMAP2). The increased holdout data can potentially help optmise towards the base activities. Figure 6 presents the trend of base accuracy (with 1% holdout percentage) on the DSADS dataset. The accuracy of Net2Net+GEM on DSADS drops much earlier than on PAMAP2, after 5 activities and dips around 30% after 9 activities. GEM and Net2Net both keep the accuracy between 30% and 40% over time.

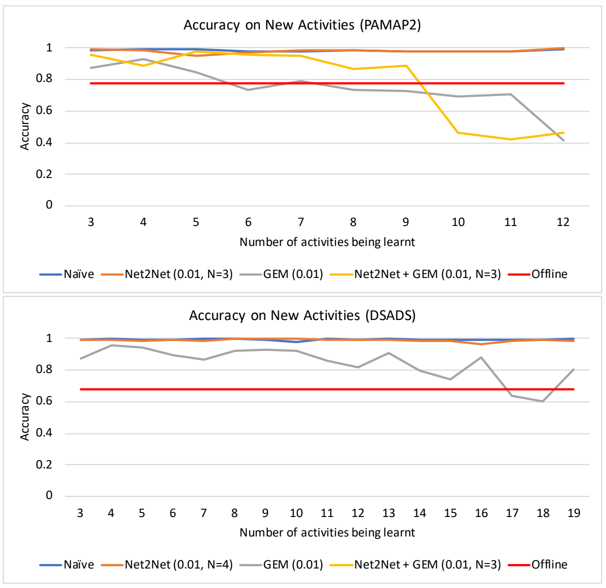

Fig. 7. Accuracy on recognising *new* activities on both PAMAP2 and DSADS

5.2. *Performance on Recognising New Activities*

The *New* accuracy is the accuracy on recognising new activities C_{t_n} immediately after training, which is used to indicate the learning capacity of the model; that is, whether the model can accommodate new classes. As we can see, Net2Net and Naive achieve the best new accuracy: above 95% on both PAMAP2 and DSADS datasets. GEM suppresses the learning on new activities, and the more holdout data we have, the lower accuracy is achieved on new activities. This is due to the fact that GEM needs to guarantee the loss on old classes does not increase, which might stop the loss on new classes from dropping quickly.

Figure 7 presents the trend of new accuracy on both PAMAP and DSADS datasets. On the DSADS dataset, Net2Net and Net2Net+GEM maintain high accuracy on recognising new activities. On the PAMAP2 dataset, the new accuracy of Net2Net+GEM drops significantly. After checking the inference results, some activities in the PAMAP2 dataset can be too similar to each other to distinguish. For example, the new accuracy drops from 77%, to 58%, 49%, and then to 17% when learning the following activities one by one: ironing, descending stairs, standing, and ascending stairs. However, the new accuracy on ascending stairs can be 100% when learning after vacuum cleaning and rope jumping. This shows that in continual learning, the challenge in activity recognition is not only on the activity itself but also the learning sequence. It is different from offline learning: the model will aim to optimise the parameters to discriminate all the classes. In continual learning, the model will optimise the parameters to discriminate the classes on hand and only adjust the parameters to accommodate new classes. However, if distinguishing the new class from the old classes requires drastic update on parameters, then the learning will not be effective, leading to low accuracy on new activities.

5.3. *Performance on Recognising All Activities*

Every time after learning a new activity, we test on all the activities that have been learnt so far ($C_{t_n} \cup C_{t_{n-1}} \cup ... \cup C_{t_0}$). We average the accuracy at each time and report the *All* accuracy in Table 2 and 3, which is used to reflect the balanced overall performance on learning both old and new activities over time. On the PAMAP2 dataset, with 1% holdout percentage, Net2Net+GEM still achieves the best all accuracy of 67%, which is 30%, 12%, and 45% higher than Net2Net, GEM, and Naive approaches. With 5% holdout percentage, it achieves the accuracy of 74%, which is very close to the offline accuracy of 78% and is 14%, 9%, 52% higher than Net2Net, GEM, and Naive approaches. Figure 8 presents the accuracy trend on recognising old activities. GEM and Net2Net+GEM perform the best over time and towards the end, GEM achieves slightly better accuracy on old activities than Net2Net+GEM. Another observation is that the accuracy of Net2Net+GEM has dropped significantly after learning 10 activities. This is due to the interference between classes (that is, activities have overlapping patterns). When this happens and there only exists a small amount of holdout data on the old classes, then the network will be optimised towards to the new classes and will be less able to recognise the interfering old activities.

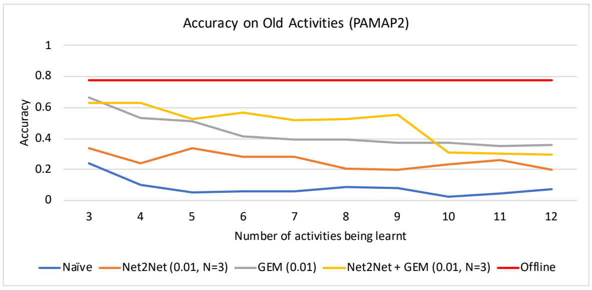

Fig. 8. Accuracy on recognising *old* activities on PAMAP2

On the DSADS dataset, with 1% holdout percentage, Net2Net+GEM still achieves the best all accuracy of 59%, which is 14%, 18%, and 40% higher than Net2Net, GEM, and Naive approaches. With 5% holdout percentage, it achieves the accuracy of 75%, which is better than offline accuracy of 68% and is 5%, 15%, 46% higher than Net2Net, GEM, and Naive approaches. Figure 9 presents the trend of accuracy on recognising old activities. In this case, Net2Net+GEM consistently outperforms than the other alternatives. At the beginning, DSADS achieves better accuracy than the offline approach, because learning with fewer activities (which are 2 or 4) is easier than learning 19 activities altogether.

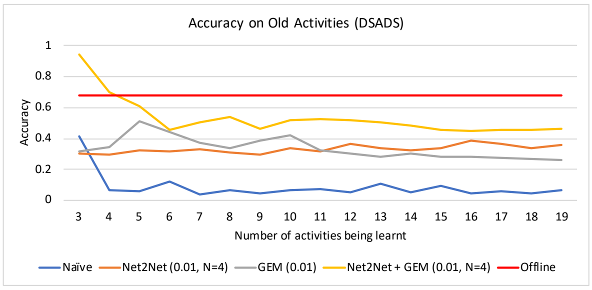

Fig. 9. Accuracy on recognising *old* activities on DSADS

5.4. *Discussion*

Continual learning in activity recognition can be a challenging task. In this paper, we propose a novel combination of Net2Net and GEM to extend the model to deal with the requirement on learning an increasing number of activities over time. The approach achieves much better accuracy in recognising old and new activities compared to Net2Net and GEM alone, and naive approach.

Table 4

Comparison of training time on PAMAP2

Dataset	Offline	Naïve	GEM		Net2Net		Net2Net+GEM	
			0.01	0.05	0.01	0.05	0.01	0.05
DSADS	0.43s	0.58s	4.3hr	15hr	0.38s	0.41s	0.2hr	0.8hr
PAMAP	0.17s	0.35s	0.9hr	1.9hr	0.31s	0.43s	0.2hr	0.5hr

5.4.1. *Computation Cost*

GEM works well in tackling the forgetting effect, maintaining the consistent accuracy on base and old activities. However, the computation cost on GEM is high. Table 4 shows the training time averaged per epoch on a modest computer[1] with Intel Core i5 8400, 32GB memory, and 2 $\times$ 500GB SSD.

GEM takes longest to train, and it reaches to 15 hours with holdout percentage 5% on the DSADS dataset. The reason is that DSADS has 19 activities and for each iteration, GEM needs to make sure the loss on each activity does not decrease. Therefore, the more activities, the more checking needs to be done and gradient updates will take longer. Figure 10 shows the increase in training time (in logarithm) after adding a new activity a time on the DSADS dataset. One way to improve the computation time is to relax the constraint; that is, not enforcing not compromising the accuracy on the holdout data in all the previous tasks [7].

An interesting observation is that Net2Net+GEM takes 20 times less than GEM alone. After investigating, we find that it is difficult to guarantee no decrease in the loss on old classes in GEM, especially when the number of old classes is large. However, after extending the network with Net2Net, adding new neurons and redistributing the weights has weakened the loss on old classes and made the inequality constraint in Equation (5) easier to be satisfied.

Net2Net alone takes less time than the naive approach and similar to the offline approach. Now the question is: *would we achieve similar accuracy by increasing the holdout percentage on the Net2Net alone approach*? To answer this question, we run another set of experiments to increase the holdout percentage on both datasets and see when we can achieve comparable *All* accuracy on Net2Net+GEM with 5% holdout percentage. On DSADS in Figure 11, when

[1] Due to the high number of experiments, we are unable to run experiments one by one on a GPU machine. Instead, we run the experiments in parallel on the cluster nodes hosted in our school. The cluster nodes are computing resources shared with all the researchers, so the computation time can fluctuate due to the competition of memory and computation with the other tasks running at the same time. Therefore, the computation time recorded here is just an indication.

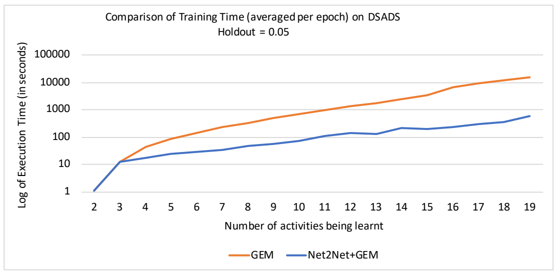

Fig. 10. Comparison of execution time (in logarithm) between GEM and Net2Net+GEM after each training

the holdout percentage increases to 10%, Net2Net can achieve similar accuracy to Net2Net+GEM. Compared to 0.8 hours on Net2Net+GEM and 15 hours on GEM, the training time per epoch on Net2Net only takes 0.5 seconds, which is more affordable with resource-constrained devices and the requirement for real-time training in human activity recognition. The increased holdout percentage only results in holding 228 more examples in memory. On PAMAP2 in Figure 11, the holdout percentage needs to increase to 20% to reach comparable accuracy to Net2Net+GEM, which means that the memory needs to host 548 more examples. But again the gain on the training time is significant, which is 0.35 seconds compared to 0.5 hours and 1.9 hours on Net2Net+GEM and GEM.

Since the increase in the holdout data can improve the accuracy of continual learning, then one future direction could be adaptive holdout data management. For example, as the number of activities grows, the system might not be able to accommodate the same amount of holdout data for each learnt task. Then the questions are: can we dynamically reduce the holdout data on some of the old classes so as to accommodate data from new classes? If so, then we can look into the selection of holdout data in terms of different criteria, including *recency* – how recent the data is, *diversity* – whether the selected data covers the sensor feature space, and *difficulty-to-learn* – whether the data leads to high loss in training, indicating the complexity of tasks.

5.4.2. *Impact of Model Expansion*

On Table 2 and 3, we have also observed that adding more neurons to the network will enhance the learning capacity but not significantly and the improvement between different numbers of neurons is within 3%. We have only attempted one way to extend the network by widening the second last layer, and there are many other options to explore, including adding more layers

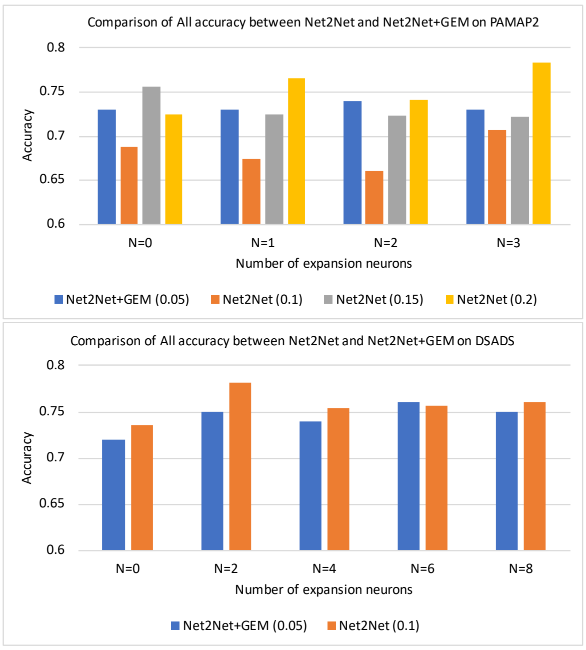

Fig. 11. Comparison of *all* accuracy between Net2Net and Net2Net+GEM

or adding neurons to different layers. However, how to determine the optimal expansion strategy is difficult. It would be desirable to depend on the new data D_{t_n} to dynamically determine whether we need to expand the network for the new set of activities, and how much to expand. An interesting future direction can be investigating self-organising network [21, 33] – evolving the architecture of the network when necessary.

5.4.3. *Accuracy Improvement*

After investigating our approach with an extensive set of experiments, we have built a comprehensive performance profile of *EMILEA* and thus identify the following areas to improve. First of all, we can increase the number of training epochs, however, which incurs higher computation cost. We might be able to perform training on a powerful GPU-powered workstation and deploy the learnt model on a resource-constrained device for activity recognition. Secondly, we only randomly sample training data as holdout data for each activity, however, these holdout data might not be representative. We will look into clustering techniques to select centroid examples and also consider to use more advanced techniques to assess the diversity of these examples so as to cover the whole input space [14].

6. Conclusion and Future Work

In this paper, we present *EMILEA* – evolving model for incrementally learning emerging activities, which is the very few first attempt that applies continual learning techniques to human activity recognition. Through extensive experiments on two real-world datasets, we have demonstrated the advantage of *EMILEA*, especially Net2Net in learning new activities over time. GEM helps mitigate catastrophic forgetting but the computation cost is too high, which might not be feasible for sensor-based human activity recognition. With the followup experiments on increasing the holdout percentage on Net2Net, we find that if the system can afford more memory; holding more examples in memory, Net2Net alone will be a better option to go, which can achieve comparable accuracy and also is more computationally efficient and thus affordable in real-world sensor-based human activity recognition systems. In terms of deployment and application, *EMILEA* can be employed in conjunction with new activity discovery techniques; that is, once a new set of activities is discovered, *EMILEA* will be activated to learn them.

In the future, we will extend the current technique to tackle evolving feature space; that is, when a new sensor is deployed and the input feature space is changed.

References

[1] R. Aljundi, P. Chakravarty, and T. Tuytelaars. Expert gate: Lifelong learning with a network of experts. pages 7120–7129, 07 2017.

[2] K. Altun and B. Barshan. Human activity recognition using inertial/magnetic sensor units. In *Proceedings of the First International Conference on Human Behavior Understanding*, HBU'10, pages 38–51, Berlin, Heidelberg, 2010. Springer-Verlag.

[3] K. Altun, B. Barshan, and O. TunÃČÂğel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605 – 3620, 2010.

[4] B. Barshan and M. C. YÃijksek. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, 57(11):1649–1667, Nov 2014.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[7] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[8] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. 11 2015.

[9] Z. Chen and B. Liu. *Lifelong Machine Learning*. Morgan et Claypool Publishers, 2018.

[10] J. L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1):71 – 99, 1993.

[11] H. C. et al. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *Proceeding of MobiSys '13*, pages 361–374.

[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*, pages 1–1, 2019.

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

[14] Z. Gong, P. Zhong, and W. Hu. Diversity in machine learning. *IEEE Access*, 7:64323âĂŞ64350, 2019.

[15] R. Kemker, A. Abitino, M. McClure, and C. Kanan. Measuring catastrophic forgetting in neural networks. *CoRR*, abs/1708.02072, 2017.

[16] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[18] Z. Li and D. Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

[19] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840, 2017.

[20] G. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 02 2018.

[21] G. I. Parisi, J. Tani, C. Weber, and S. Wermter. Lifelong learning of human actions with deep neural network self-organization. *Neural Networks*, 96:137 – 149, 2017.

[22] P. Rashidi and A. Mihailidis. A survey on ambient-assisted living tools for older adults. *IEEE Journal of Biomedical and Health Informatics*, 17(3):579–590, 2013.

[23] S. Rebuffi, A. Kolesnikov, and C. H. Lampert. icarl: Incremental classifier and representation learning. *CoRR*, abs/1611.07725, 2016.

[24] A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109, June 2012.

[25] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[26] J. H. Shin, B. Lee, and K. S. Park. Detection of abnormal living patterns for elderly living alone using support vector data description. *IEEE Transactions on Information Technology in Biomedicine*, 15(3):438–448, 2011.

[27] A. Tapus, M. J. Mataric, and B. Scassellati. Socially assistive robotics [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14, 2007.

[28] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *CoRR*, abs/1904.07734, 2019.

[29] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, Mar 2018.

[30] D. L. Weeks, G. L. Sprint, V. Stilwill, A. L. Meisen-Vehrs, and D. J. Cook. Implementing wearable sensors for continuous assessment of daytime heart rate response in inpatient rehabilitation. *Telemedicine and e-Health*, 24(12):1014–1020, 2018.

[31] J. Ye, L. Fang, and S. Dobson. Discovery and recognition of unknown activities. In *Proceedings of Ubicomp '16 Adjunct*, pages 783–792. ACM, 2016.

[32] J. Ye, F. Zambonelli, and S. Dobson. Lifelong learning in sensor-based human activity recognition. *IEEE Pervasive Computing*, 2019. To appear.

[33] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks, 2017.