University
of
St Andrews

Continual Learning with GANs

Pakawat Nakwijt
Matriculation number: 180008901

MSc Artificial Intelligence
School of Computer Science

August 2, 2019

Abstract

The smart home is a cost-effective way to improve people's living. Many highly accurate solutions were proposed recently [33, 35]. However, one major obstacle towards the real-life smart home is the ability to adapt to change [39]. Driven by this challenge, continual learning plays a major role by introducing a new ability to solve new problems and, at the same time, to retrain the previously acquired knowledge. Many studies have been done in image recognition [30, 20, 40, 18] but no study on the smart home domain.

In this paper, I tackled such challenges using a technique called "generative replay". Inspired by continual learning ability in human, two agents were implemented; one generative model was implemented to mimic all trained lesson while one fully-connected neural network was gradually trained to capture structured knowledge. Two architectures were proposed namely multi-GAN and single-GAN. The former represents each target class by individual GAN but the latter used only one GAN to capture all classes. I hypothesised that the multi-GAN could benefit another in term of efficiency and flexibility while the single-GAN could be used better in a scalable manner.

Through an extensive experimental evaluation, it shows that both two models are successfully integrated into the smart home domain. It outperformed the traditional model to gain new knowledge and also maintained 92.4% accuracy of the previously learned task. In overall, it could reach up to 80.1% of the expected accuracy.

Besides that, apart from catastrophic forgetting, interclass interference and GAN instability are addressed through many pieces of evidence throughout the report. They have been shown that they are crucial challenges that prevent the model to reach proper performance but have not been addressed before in the literature.

The implementation of the project can be downloaded at `https://gitlab.cs.st-andrews.ac.uk/pn37/continual-learning/`

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is *10,690* words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker and to be made available on the World Wide Web. I retain the copyright in this work.

Date ____________ **Signature** ____________

Pakawat Nakwijit

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor Dr Juan Ye for her advice and support throughout this project. Without her guidance and persistent, this dissertation would not have been possible.

I would like to thank all members of my family for giving me invaluable encouragement to pursue my own dream.

In addition, a thank you to all my friends I meet here. You have made the year in St Andrews become a remarkable memory. Lastly, I would like to thank the Royal Thai government scholarship for giving me this opportunity.

Contents

Chapter 1

Introduction

The smart home is an automated system developed to improve people's quality of daily life. Elderly and disabled people could receive better home care support and easier life environment. It is also a cost-effective way to monitor and save utility cost. To do so, smart homes are equipped with sensors and actuators to collect and process data. These different sensor technologies can be combined to build an artificial intelligence system that is capable of providing feedback to house owners in the form of a self-adjusted environment or smart recommendation. Identifying human activity is a fundamental task to include smartness to the system.

However, existing solutions were usually designed specifically to classify on a particular set of activities. The models cannot perform successfully in the context of smart home because new activities can be exposed and activity routines can be changed every day [39]. Without continually learning ability, the ideal smart home might not be possible.

Previous attempts to design a continual learning system got inspiration from how a human brain learns. Complementary Learning Systems (CLS) theory suggests that a dual-memory system is key to generalise new knowledge while retaining episodic memories [24]. The study shows that people use two parts of our brain namely hippocampus and the neocortex to build up incrementally and continuously learning ability. One is used to quickly learn an upcoming lesson while another one is gradually trained to capture structured knowledge. Shin et al. proposed a simple solution to imitate this brain activity called generative replay. They applied two neural networks; one network was used as a classifier together with a generative model as experience storage. Their technique helps the system avoid knowledge lost by replaying previously learned knowledge while the system is learning [30]. They showed that their system is able to continuously learn over time and also maintain its ability on previously learned tasks without significant degradation.

Despite Shin et al.'s solution success as a continual learning system but it is only evaluated on a simple image classification task. Different results could be seen in activity recognition because of activity variability, class imbalance and noise. New difficulties being specific to activity recognition could be observed. Moreover, with the increasing popularity of Generative Adversarial Networks (GANs) as a generative model, many studies show that it has succeeded in generating high-quality images [37]. It could be efficiently used as a replay module in this problem. This dissertation aims to develop a continual learning framework for activity recognition using generative replay technique and to explore the potential of using GANs as the memory replay module. Further improvements in term of continual ability, generality and robustness along with new challenges are thoroughly discussed.

Chapter 2

Context survey

This chapter consists of reviews of the background and related works to this project. An overview of the definition of activity recognition and the general approaches to deal with it are briefly discussed. The state-of-the-art continual learning models are also presented following with GAN's concept and its various.

2.1 Sensor-based Activity Recognition

Activity recognition is a task to discover high-level contextual information about human activity from raw sensor inputs. Chen et al. characterized it into four basic subtasks; choosing appropriate sensors to monitor and capture a behaviour, collecting and processing perceived information, building models to conduct reasoning, and inferring activities [8]. Activity recognition is commonly classified by type of sensor used into two categories; vision-based and sensor-based. The approaches in the first category use visual data such as images or videos as a clue to recognise activity patterns. It intensively uses computer vision techniques such as edge detection and noise reduction. Despite visual data is intuitive and information-rich to human but it is complicated to machines so more subtasks are necessary to be done, for example, feature extraction, structural modelling, movement segmentation, action extraction and movement tracking [8]. It also severely suffers from privacy and ethical concerns [7]. The sensor-based techniques cope with the problem differently by using smart sensors. Accelerometers, gyroscopes, light, and sound sensors are combined to generate a broad picture of the people activity in the house. Even these sensor data are not explicitly readable but it fully captures all information needed to activity recognition. They are also cheaper to collect and process.

Sensors can be attached to a person (wearable device) or an activity environment (dense sensing). Wearable sensors are usually embedded in clothes, eyeglasses or mobile phone. Accelerometer and GPS sensors are commonly used because they inherently associate to basic movements such as sitting, standing and walking. However, critics have argued that wearable sensors alone may not be enough to differentiate high-level activities such as drinking water and making tea because there is not informative connection to its environment. It also has serious technical issues in term of usability. Size, ease of use and battery life span critically impact how people willing to use [7].

Dense sensing based activity monitoring, in contrast, embeds sensors within environments so it can identify human activity through human interactions with objects in the environment. A large number of sensors are deployed in daily used objects or places wiring together with sharing communication networks. Due to the advances and maturity of supporting technologies, cheaper sensors, lower-cost storage and higher-capacity processing become prevalence. These have pushed forward the development of object-based activity recognition. However, it has not been practically deployed because of its complexity to handle the exploding number of data produced by these sensors. Incompleteness and uncertainty from sensors, background noise and ambiguity of human action are commonly observed.

Early attempts are to exploit handcraft knowledge to classify or predict activities through formal logical reasoning. None or small samples are required to construct knowledge structures which can be represented as schemas, rules, or networks. However, it failed to be used in practice due to its limitation to deal with incompleteness and uncertainty of the data.

Data-driven approaches were later widely adopted in this field because of its capability of handling uncertainty and its simplicity to capture a complex relationship by leveraging the exploding amount of data. Traditional classification algorithms such as Hidden Markov Models or Conditional Random Fields could achieve satisfactory results with accuracy up to 80-90% [33]. Despite that, a survey by Wang et al. suggested that neural models especially deep neural techniques are recently getting attention in the literature. By designing deliberately, it could outperform the existing technique to recognise higher-level or more complicated context-aware activities [35].

This study focuses mainly on modelling a neural network with sensor data. However, I do not aim to improve the accuracy of the classifier but I would rather more concentrate on infusing the ability to learn continuously to the existing solutions.

2.2 Continual learning

Current state-of-the-art neural networks become broadly accessible to everyone affecting how people live and work. Despite its impressive performance on a wide variety of problems, these machine learning models were designed specifically to perform on only a particular task. Once it was trained, it cannot re-train without forgetting previously acquired knowledge. Besides, these agents typically learned from stationary batches of training data, but, in reality, the information might appears in the form of a stream of data and is incrementally available over time. Human behaviours are changing through time [39]. A new activity might be introduced after new smart devices are installed. New patterns might be commenced after new members. Activity routine might also be shifted due to injury. The existing learning algorithms mostly fail in these situations.

To build intelligent autonomous agents interacting in the real world, continual learning ability becomes more and more important. Pascanu et al. framed the term "continual learning" in NeurIPS workshop 2018 with five properties [23].

1. It should be able to lean with no boundaries between tasks.
2. It should be able to transfer knowledge from the previously seen tasks to new ones.
3. It should be able to maintain previously learned tasks – avoid catastrophic forgetting.
4. It should have a bounded size.
5. It should not have direct access to past tasks.

Following these properties, a key challenge is to avoid catastrophic forgetting. This term was coined by French in 1999 describing a problem that typical training algorithms cannot incrementally learn new tasks [9]. While a model is learning from a new piece of information, it will interfere with previously learned knowledge which eventually results in greatly reducing the accuracy of the known tasks. Moreover, because data from all tasks might not be simultaneously available during training, once the model forgets the knowledge of the past tasks, it might not be possible to re-train to acquire that knowledge again.

Three broad approaches are used to get through this challenge, namely regularization, architectural and rehearsal strategies.

2.2.1 Regularization strategy

As inspired by continual learning in the human brain, it suggests that after learning occurs, a process called "synaptic consolidation" reduces the plasticity of synapses that relate to previously learned tasks making them harder to change. It, therefore, reduces the interference from newly received data.

The regularisation in neural networks does similarly by controlling the way the network updates its weight for a balance between learning new tasks and retaining learned knowledge. The importance of weights will be estimated in terms of their impact on previous tasks' performance. It penalizes changes in the important parameters of the previous tasks. This will slow down changes in that corresponding parts of the networks.

Elastic Weight Consolidation by Kirkpatrick et al. uses the Bayesian approach to quantify the importance of parameters in term of the posterior distribution [16]. On the assumption that there are many sets of parameters that lead to low errors for a certain task, a common set of parameters that can produce a low error in all task does exist. Their goal is to constrain the network's parameters to stay in the low-error region and explore those sets of parameters to find the one that fits all tasks. A similar method was proposed by Zenke, Poole, and Ganguli with a slightly different way to formulate the importance of parameters.

The main limitation of these approaches, however, is that they highly depend on the relevance of the tasks. The more irrelevant the problems are, the more difficult it can learn. The hypothesised set of parameters might not exist so this leads to a bias toward maintaining existing knowledge rather accepting the new information.

Li and Hoiem introduced another different method called "learning without forgetting". It is not to regulate the network's weights but to maintain the predicted probabilities for all previously learned classes [18]. It will penalise the model when the new prediction distribution is different from those of previous prediction distribution. This method can also be used to transfer knowledge from one large model to the smaller one [12].

2.2.2 Architectural strategy

Another straightforward way to avoid forgetting in neural networks is to dynamically accommodate neural resources for upcoming information leaving more room to learn a new thing. For example, to increase the number of neurons or to update connectivity patterns or to add task-specific components to the network.

Progressive network by Rusu et al. resolves the forgetting by introducing a new neural network for each new task [28]. The previous knowledge will be transferred to the new network through lateral connections. Because there is no interference to the learned networks, so there is no catastrophic forgetting. In many cases, it also accelerates the learning process simultaneously.

Masse, Grant, and Freedman coped the problem differently by allocating a different subnetwork for a different task [20]. During the training step, only a subset of nodes in the network will be assigned randomly to participate in each task. As inspired by activities in the human brain, sparse and mostly non-overlapping sets of dendritic branches are active for any one task. This reduces the interference between tasks and also increase the stability of the known knowledge by distributing those informative data throughout the network.

These architectural techniques are simple and consistent but one major drawback is scalability. Increase in the number of learned tasks often leads to the complexity of the architecture to grow. It is a serious concern when it is deployed as a lifelong model.

2.2.3 Rehearsal strategy

Rehearsal strategy is another neuroscience-inspired solution. Complementary learning systems theory suggests that human continual learning ability is a result of the complementary contribution of the hippocampus and the neocortex [17]. The hippocampus function in rapid learning of new information while the neocortex is learning generalities of the tasks using its long-term storage. The dual-memory models were proposed. It consists of two complementary components called the classifier and the generator; one capable of generalizing knowledge across experiences and another one retains specific memories of episodic-like events.

Lopez-Paz et al. proposed a framework called Gradient of Episodic Memory. It uses a subset of the observed examples as replay samples. Even they showed that it improved overall performance but it obviously requires storing all past data, which is storage inefficient and impractical. They also highlighted that the accuracy of their system highly depends on the size of saved samples. Under limited memory assumption, Shin et al. rather use pseudo-data as replay samples. They are generated by a generative adversarial network that is trained to mimic all previous data [30]. Because this generative replay model does not hold the entire dataset but only tiny networks, it is more plausible to scale. They also showed that only a slight decrease in performance of the previously learned tasks is observed. Another replay strategy called iCaRL by Rebuffi et al. [25]. This method stores only a tiny subset of data as "exemplars" as the representative information of the class. These exemplars will be used to augment the classifier result using the nearest centroid. A survey by Ven and Tolias shows that these replay-based approaches surpass other approaches in all various scenarios [34].

However, continual learning is a relatively new subject. The majority of the study is only focusing on image-related tasks such as object recognition, image classifier. There still is no study adopting these techniques in other domains. This project will explore a way to use these techniques and develop additional components to improve the accuracy specifically to the smart home domain.

2.3 Generative adversarial networks

In 2014, Goodfellow et al. proposed a generative model called "Generative Adversarial Networks (GANs)" [10]. Many studies have shown that it is capable of mimic new data at surprising levels of accuracy which later widely adopted in the deep learning community [37].

Generally, the problem of generating a new image of a certain object can be rephrased into a problem of generating a random vector that follows given probability distribution. However, training a network directly to express the right transform function which represents a complex probability distribution is difficult. Goodfellow et al. provided a solution to this problem by using a minimax two-player game. Two players; a generator and a discriminator are competing with each other. This process then enforces the generated distribution to become closer to the true distribution or achieve what is called *Nash Equilibrium.*

On the one hand, the generator's objective is to generate new samples mimicking the given data. On the other hand, the discriminator's objective is to identify the validity of the data. It could be thought of as analogous to a game of cat and mouse between a counterfeiter and police. While the counterfeiter is learning a new way to fake money, the police are constantly improving detecting technique. The competition between them makes these two agents progress with respect to their respective goals.

Formally, the generator is defined by $G(z)$. By giving a noise z, it then converts that noise vector into a synthetic sample. The discriminator is defined by $D(x)$. It estimates the probability whether the input comes from the actual dataset or be generated by the generator.

To represent two objectives of the zero-sum game, the discriminator D will be trained to maximize the probability of assigning the correct label by maximizing $E_{x \sim p_r(x)}[logD(x)]$. Meanwhile, the generator G is trained to trick the discriminator, in other words, it is expected to minimize the gap between a generated instance and a genuine instance. Given a fake sample $G(z) : z \sim p_z(z)$, the generator's target is to minimize $E_{z \sim p_z(z)}[log(1 - D(G(z)))]$

By combining both aspects together, the loss function of this minimax game can be formulated as follows.

$$min_G max_D L(D, G) = E_{x \sim p_r(x)}[logD(x)] + E_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (2.1)$$

2.3.1 Conditional Generative Adversarial Networks

The Conditional Generative Adversarial networks (CGANs) was proposed by Mirza and Osindero in 2014 [22]. They aim to gain controllability to direct the data generation process by conditioning the model on additional information. It will provide a new way to control what categories of data being generated. Their model is as simple as vanilla GAN but the input vectors will be extended to embed additional information – which usually is a class label. Only the discriminator and generator loss terms are changed to be conditional on that new piece of information.

$$min_G max_D L(D, G) = E_{x \sim p_r(x)}[logD(x|y)] + E_{z \sim p_z(z)}[log(1 - D(G(z|y)))] \quad (2.2)$$

2.3.2 Wasserstein Generative Adversarial Networks

However, it is challenging to train a GAN model because of its instability and slow or even fail to converge. Low dimensional supports and vanishing gradient can be generally observed while GAN is being trained [1]. These cause imbalance between discriminator and generator which also has a sequel effect to instability during training progress. Training a proper discriminator also faces a dilemma [1]. While the discriminator has almost perfect accuracy, the gradient of the loss function will decrease down to zero making learning progress becomes slow or fail to converge. In another hand, while the discriminator has poor accuracy, the generator does not have accurate feedback resulting in the unsatisfied final result. It can be seen that training GANs is non-trivial.

One possible solution is a new loss function. Arjovsky, Chintala, and Bottou introduced a new metric called Wasserstein distance as GAN loss function [3]. The Wasserstein distance measures the distance between two probability distributions represented by the minimum energy cost of transforming one probability distribution to the shape of the other distribution. Arjovsky, Chintala, and Bottou suggested that it provides a smooth and more meaningful representation of the distance which might stable GAN's learning process. Further WGAN improvement from Gulrajani et al. suggested another term called "gradient penalty" to avoid unstable training by controlling the magnitude of discriminator's weight making it satisfies Lipschitz continuity condition [11].

However, training GANs is still challenging. Non-convergence can be observed even in WGAN with gradient penalty. In addition, mode collapse where the model can only produce a few modes of data and overfitting where the generated data is just a duplication of the trained input are generally observed. By the time of writing, there is no universal solution to this problem.

Chapter 3

Design

This chapter provides a summary of the design consideration beginning with an overview of the dataset followed by how the dataset is transformed to be feature vectors. The framework architecture is also presented. Lastly, the formal definition of tasks and training process.

3.1 Dataset

The twor.2009 datasets from the CASAS was primarily used in the project [5]. It is hand-segmented dataset recording activities of daily living of two residents. Sixteen activities are labelled in the dataset. This testbed consists of six types of sensor: motion sensors, door sensors, light switch sensors, water flow sensors, burner sensors and item sensors. The sensor layout is shown in figure 3.1.

An example of the data is shown in table 3.1. It consists of four columns; timestamp, sensor identifier, sensor state and activity label. Each row in the table describes a sensor events when a resident has interacted with an object embedded with a sensor or has moved in a certain location in the room. The activity labels are only marked at the beginning and the end of the activities.

Timestamp	Sensor ID	Sensor state	Activity Label
2009-02-06 17:15:22.428030	M41	ON	R2_work_at_computer begin
2009-02-06 17:15:27.645659	M41	OFF	
2009-02-06 17:15:30.882620	M40	ON	
...	...	...	...
2009-02-06 17:47:16.229419	M45	OFF	R2_work_at_computer end

Table 3.1: Example of sensor events in CASAS twor.2009 dataset

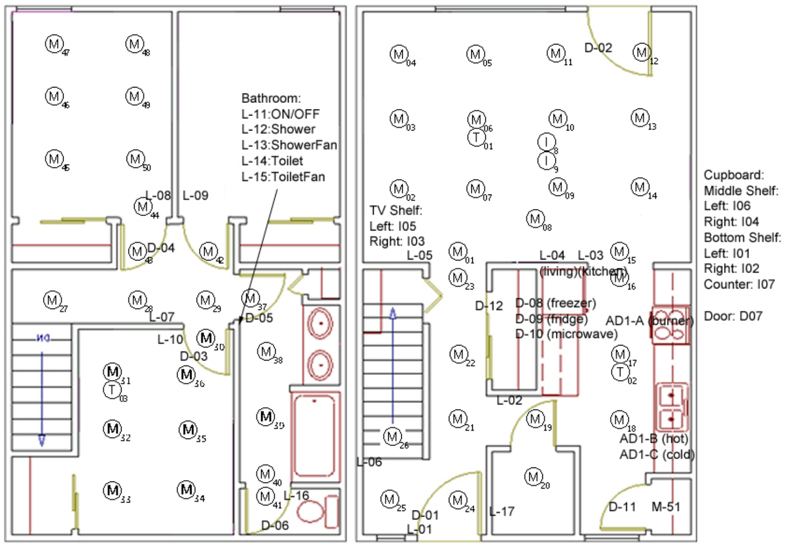


Figure 3.1: Sensor layout of the CASAS twor.2009 dataset [5]

3.1.1 Overview of Data

Figure 3.2, 3.3 demonstrate the distribution of activity class and the average number of sensor event per activity of each class respectively. The left chart shows that *Cleaning*, *R1_work_at_dining_room_table* and *Wash_bathtub* have less than ten records. They are a relatively small number of data so they were filtered out. Only 10 activities were selected from the rest including R1_work_at_computer, R2_work_at_computer, R1_sleep, R2_sleep, R1_bed_to_toilet, R2_bed_to_toilet, R2_prepare_dinner, R2_watch_TV, R2_prepare_lunch and R1_work_at_dining_room_table.

One observation that is important to point out is that each activity has a vastly different number of sensor events shown in figure 3.3. Because of the vast difference in duration an agent spent to perform a certain activity. It could be problematic in term of unbalanced data which will be discussed more in the next section.

3.1.2 Feature Representation

Note that different sensors behave differently; for example, temperature sensors are usually set as periodic measurements throughout the day but motion sensors are used as triggered sensors providing ON signal when motion is detected, and then an OFF signal if motion is no longer detected. Because the aim of this project does not focus on overcoming the state-of-art approaches' accuracy, by avoiding the over-complexity process, only binary sensors are used throughout this project.

One important observation I found during data preprocessing is that triggered sensors could have two different behaviours. One has been discussed in the pre-

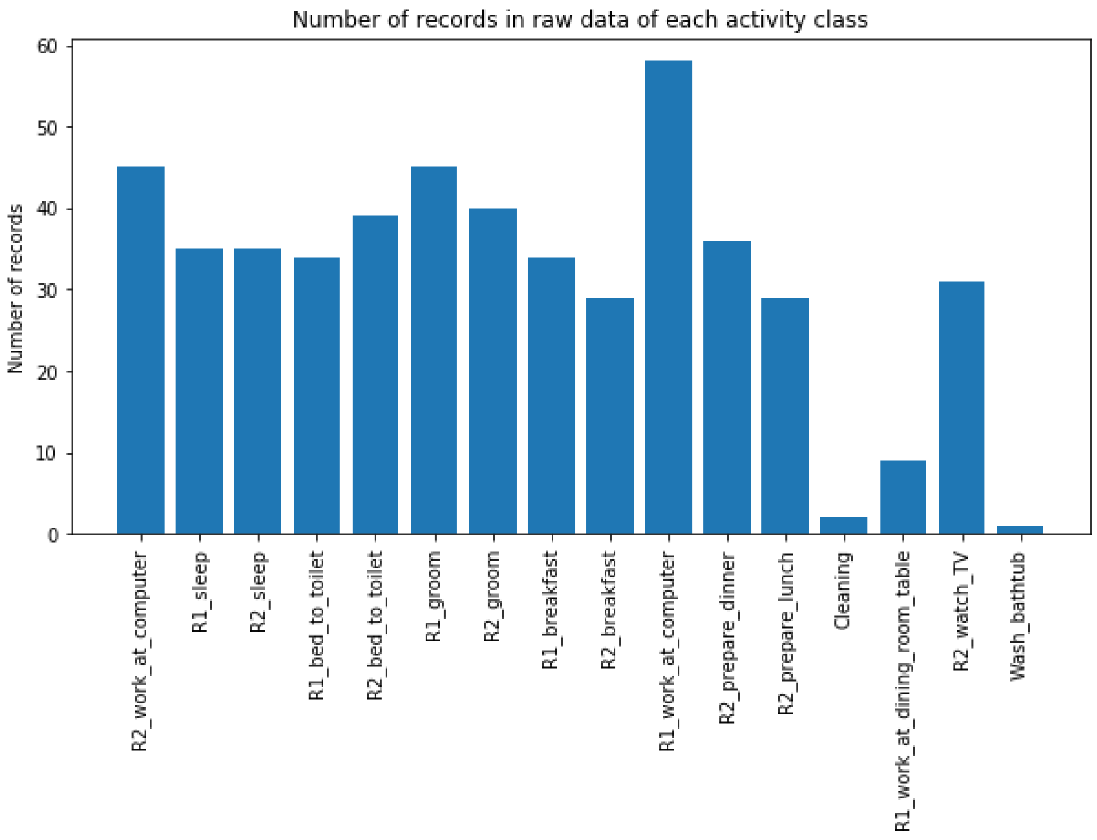


Figure 3.2: A bar chart showing the class distributions of the CASAS twor.2009 dataset

vious section while another can be observed in the door sensors as am example [33]. Despite, door sensors return a binary state of action as to motion sensors do but once their state changes, it remains the same value until the next activation. However, Twomey et al. suggest that state change of sensor is significantly more meaningful due to the fact that the changes represent an interaction so this difference can be ignored.

In the project, I only considered the state-change representation of the triggered sensors. The raw data were converted into a set of time slices; each activity was sliced within a time window of five minutes and fifteen minutes. The overlap between those slices was also considered. I note that the time window is largely arbitrary. Because it is not the main focus of the study, only briefly investigation is carried on. However, in practice, this window affects directly to the time delays of the system.

Under the construction of the state change representation, state changes were

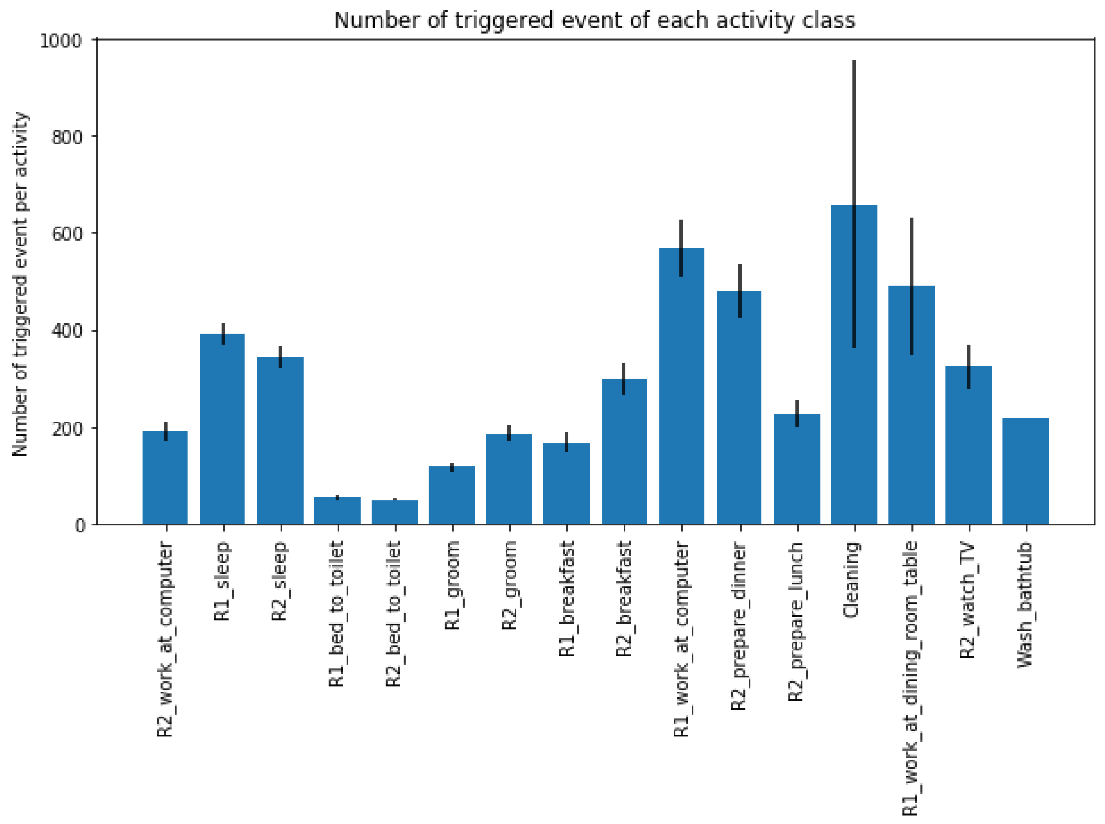


Figure 3.3: A bar chart showing the average durations of activities of interests

counted within the time window for each sensor. The analogue value will be normalised across the dataset. The data then will be normalised again as a row-by-row basis. As a result, the final vector could be interpreted as a density of sensor events under that particular time window. This normalisation introduced continuity to the data which is an important assumption for many parts I used especially oversampling and GANs.

The resulting datasets consist of 6766, 3371 and 55775 rows for the dataset with five minutes, fifteen minutes time window and two minutes overlapped respectively. Examples of the feature representations are shown in Table 3.2.

3.2 Architecture Overview

As a dual-memory model, the training process into two steps. At the first training session, the classifier and the generator receives a new dataset. They then learn to classify and mimic the given information respectively. When the next datasets

Activity Label	AD1-A	AD1-B	D01	D02	D03	...	M51
R2_work_at_computer	0.0	0.0	0.5	0.3	0.2	...	0.0
R2_work_at_computer	0.0	0.0	0.0	0.8	0.0	...	0.1
R2_work_at_computer	0.0	0.0	0.0	0.0	0.0	...	0.0
R1_sleep	0.0	0.0	0.7	0.25	0.0	...	0.0
R1_sleep	0.1	0.15	0.0	0.0	0.0	...	0.0

Table 3.2: Example of the feature representation

arrive, a fixed number of replayed samples drawn from the previous generator are mixed with those new data. This new dataset will be managed to train both the classifier and the generator. The balance between the importance of a new task and known tasks can be parameterised by weights in classifier's loss function. This process will be repeated until all classes are trained. Figure 3.4 gives an overview of our system.

Two architectures of the generator are considered here. First, the multi-GANs model represents each target class by a separated GAN. Only actual data are used while the generator is being trained. Second, called single-GAN model, it uses only one GAN for all classes. Instead of training separately, the sequential training procedure is applied using both actual and replayed samples. These designs followed the hypothesis that the continual learning model could get higher performance when the generator uses one GAN per class. It also implies flexibility and higher quality of generated data. But it might face with the scalability problem because of its unbounded size which the single-GAN could be more advantageous in this aspect. Differently to the multi-GAN, this single-GAN might suffer more severely from catastrophic forgetting and GAN instability. The pros and cons will be discussed thoroughly in the evaluation section.

3.3 Task Protocols

In this project, I focus on one scenario called *"incremental class learning"*. It can be defined as a model that can identify a given task and recognize a task identity [34, 14]. The classifier will be implemented as a multi-headed output layer. The class probability for each label will be assigned by its own output unit. The number of output units will be dynamically increased by the number of new classes on each training session. As a result, the network's output vector could be interpreted as a probability vector across all previously learned classes.

Similar to a definition by Shin et al., A model H is defined as a tuple $< S, G >$ where a generator G is a generative model that is trained to mimic previously

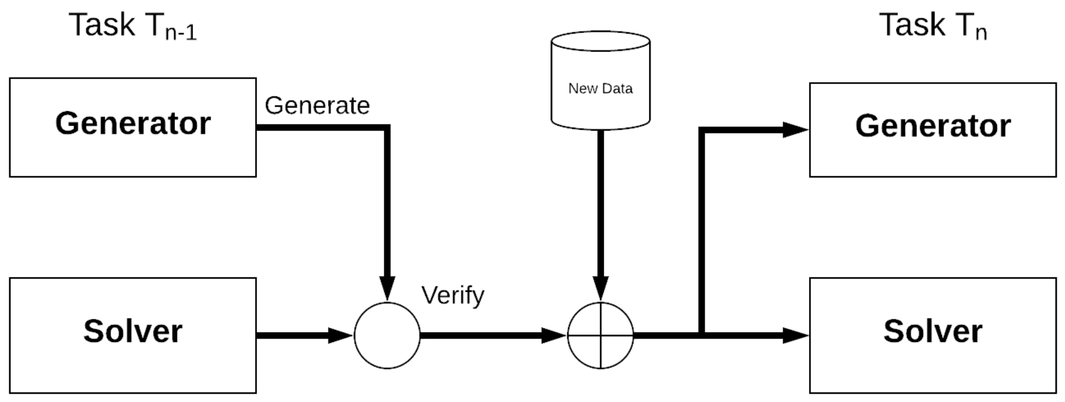


Figure 3.4: Architecture of the framework consisting of two key components called a classifier and a generator

trained datasets and a classifier S is a simple neural model that can incrementally learn all tasks in the task sequence T [30].

The task sequence T of N tasks is defined mathematically as $T = (T_1, T_2, T_3, ..., T_N)$. A task T_i is to optimize a model to classify between two classes of activity C_i1, C_i2 trained by feature vectors $< f_1, f_2, ...f_M >$ as discussed earlier. The classifier's loss function could be formalised as

$$L(S_i) = rE_{(x,y)\sim T_i}[L(S(x), y)] + (1-r)E_{x'\sim G_{i-1}}[L(S_i(x'), S_{i-1}(x'))] \tag{3.1}$$

The main goal of the model is to minimize the sum of losses among all tasks in the given task sequence. On a training session, the loss function will be calculated from actual loss and replayed loss regulating by a factor r representing how important of the new task. For the later experiments, r=0.5 representing equivalent important was used. The vary of this parameter will be discussed more in the evaluation. Lastly, as following the standard configuration in training neural nets, cross entropy is mainly used as function L.

Chapter 4

Implementation

The project was implemented in Python. **Pytorch** is intensively used as the main library because it provides a wide variety of building blocks to build neural networks, leveraging maximum flexibility and computation power of CPUs and GPUs. The core implementation in the project was adopted from the implementation of a study by Ven and Tolias [34]. Several pieces of code are from https://github.com/soumith/ganhacks, https://github.com/xuqiantong/GAN-Metrics and https://github.com/eriklindernoren/PyTorch-GAN

The following sections consist of how the framework treats the input dataset, implementation decisions in the classifier and generator to improve the accuracy and robustness and generality of the models. Lastly, the comparison methods is also presented.

4.1 Datasets

Three different sets of feature vectors were examined. The offline classifier in the testbed shows an average accuracy across all tasks at 0.884 ± 0.071, 0.889 ± 0.076, and 0.653 ± 0.082 for five- and fifteen-minute non-overlapped windows and for two-minute overlapped windows respectively.

It showed that introducing overlapping between the sliding window does not increase accuracy. It sometimes results in a less desirable model. It might be because the overlapping introduces significant amount of irrelevant data. These could be consider as noise making the data harder to classify.

Considering different window size, the testbed shows a similar result between those two. An increase in window size (in a certain range) might not change the data's key properties. I argue that the smaller window size could be more applicable as an online model because it has a smaller time delay which increases interactivity between the system and users. By contrast, the bigger time window significantly reduces the amount of data which also reduces the memory needed and training time. It might be a better option for a problem that can be trained offline.

Therefore, the dataset with five-minute non-overlapped window was only used for the rest of the project.

4.1.1 Oversampling

As discussed in the design section, the imbalance was observed (see figure 3.2 and 3.3). It causes a bias in a model. A learning algorithm could simply classify everything as the majority class and still be high accuracy. The result is misleading and useless because the model cannot work in general. There is no additional knowledge captured by the model.

Oversampling was implemented to handle the imbalanced problem. I used Synthetic Minority Over-sampling Technique (SMOTE) [6]. SMOTE fixes the problem by synthesis new data from the minority class. These new data will be selected and labelled by the k-nearest neighbours based on those that already exist.

4.2 Classifier Model

Following the state-of-the-art configuration, a fully-connected neural network with two hidden layers is used as a classifier. ReLU activation function was used in all hidden layers to avoid vanishing gradient. Adam optimizer is applied. As highlighted by Kingma and Ba, Adam optimizer is a better options for problems with noisy data, sparse gradients and non-stationary objectives. The experimental results from recent studies also confirmed that Adam optimizer could yield better results in both generative and classifier models [27, 11] ans also in continual model [40, 13]. The learning rate and the batch sizes were set as the default setting (learning rate=0.01, batch sizes=5). Only the number of hidden units per layer was thoroughly examined by grid search validation (it shows that 500 hidden units per layer was the optimum setting)

4.2.1 Regularisation

One significant issue to the overall performance of our approach is interclass interference. The problem is observed by T-SNE technique. Figure 4.1 shows that there are up to 10 classes that clustered together. Considering two tasks T_i, T_j where $\{C_{i1}, C_{j1}\}$ both belong to a same cluster and also $\{C_{i2}, C_{j2}\}$ but $\{C_{i1}, C_{i2}\}$ are not from different clusters and also $\{C_{j1}, C_{j2}\}$.

It could be seen that it is easy to build a model T_i, T_j because there is a clear seperation hyperplane between $\{C_{i1}, C_{i2}\}$ and also between $\{C_{j1}, C_{j2}\}$.

But because of the first condition, the separation hyperplane of T_i and T_j tend to be similar. Therefore when two models are merged or trained together, they will interfere with each other. See figure 4.2 for an example; T_i consists of class {R2_prepare_dinner, R2_watch_TV} and T_j consists of class {R2_prepare_lunch, R1_work_at_dining_room_table}. The accuracy of the offline model when they are trained individually is {0.99, 1.00} but {0.64, 0.84} when they are trained together.

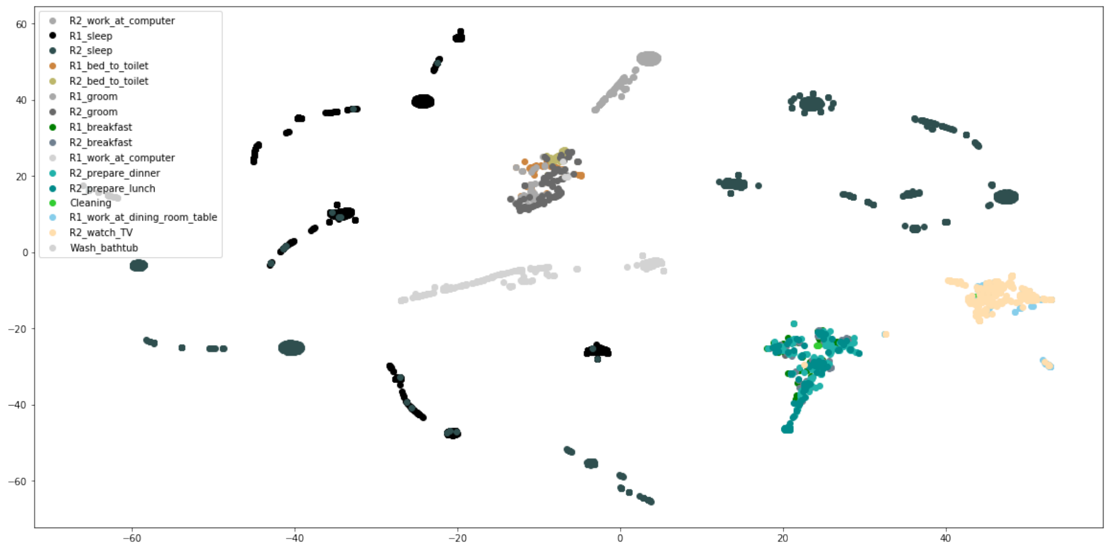


Figure 4.1: T-SNE across all classes

To tackle the interference, I introduced a regularisation component. The changes that affect the previous task's accuracy will be penalised so it will slow down the interference. Two techniques are tested here; Elastic Weight Consolidation(EWC) and Knowledge distillation [16, 12].

Apart from the interclass interference, several common difficulties in the smart home domain were also found such as activity variability where the same activity may be performed differently by different individuals, interclass similarity where different classes show very similar characteristics in the input space. (see figures 4.4a- 4.3b for activity variability and figures 4.4a -4.4b for interclass similarity). In addition, class imbalance where only a few activities occur often while most activities occur rather infrequently and noisy data where several sources of noise could be introduced such as by sensor failure and activity ambiguity are also addressed in other components of the framework.

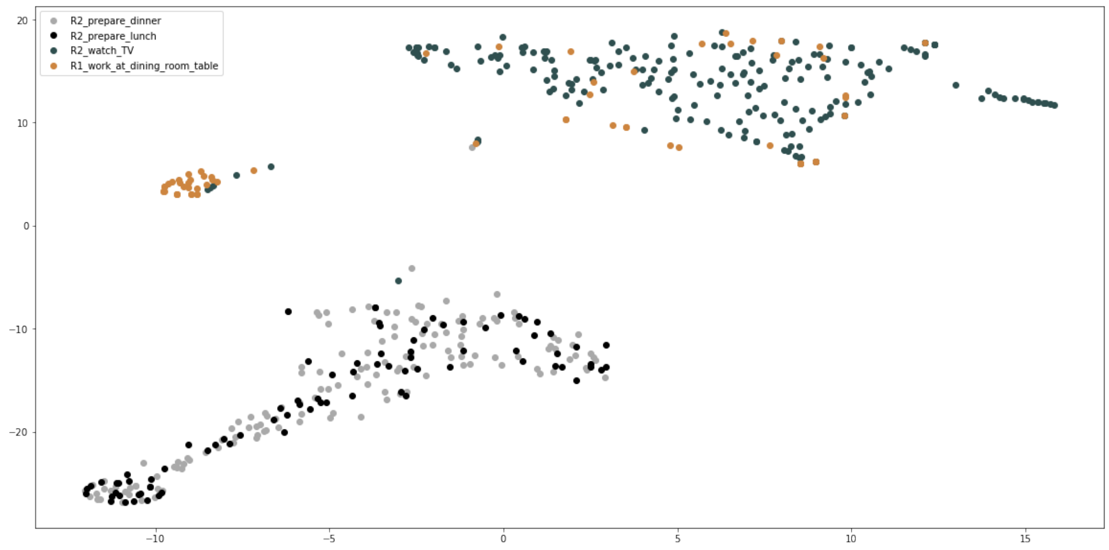


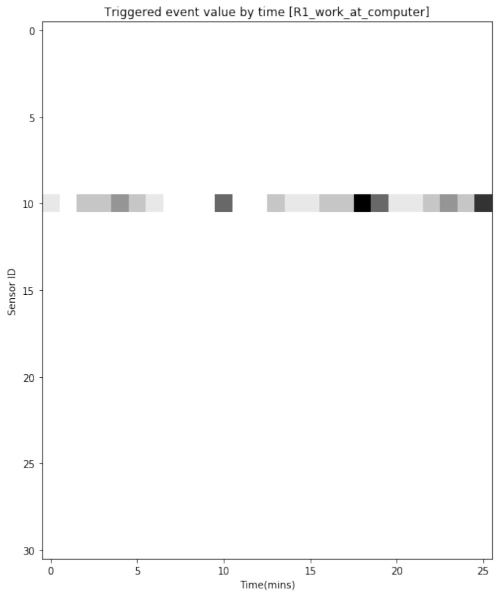
Figure 4.2: T-SNE on four classes: {R2_prepare_dinner, R2_watch_TV R2_prepare_lunch, R1_work_at_dining_room_table}

4.3 Generative Models

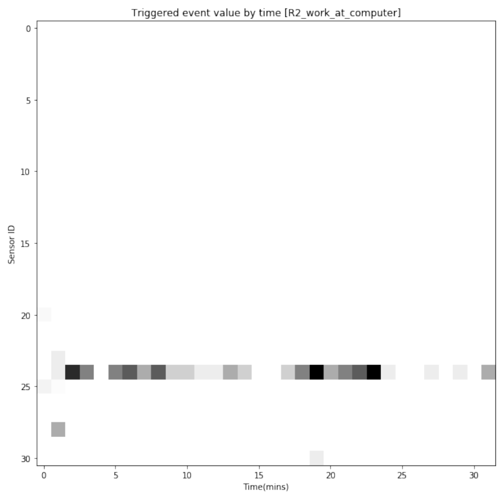
With the same justification for the classifier, the generative model is implemented as two fully-connected neural networks with two hidden layers. ReLU and Adam optimizer are applied. Due to the experimental results, 100 of hidden units per layer was set to all subnetworks.

In this project, I experiment on four different various of GANs including vanilla GAN [10], Wasserstein GAN [3], Conditional GAN [22] and CWGAN (it is a conditional GAN but uses Wasserstein distance as a loss function).

The first two were used only in multi-GANs architecture while the last two were used only in single-GAN architecture. This constraint was applied to maintain its simplicity but also having a controlability over what categories of data being generated. Non-conditional GANs are typically trained faster and shows higher stability so they are a better option for multi-GANs architecture. However, because they cannot be specific to generate a certain label which is necessary for the single-GAN version of the generator so conditional GANs were used instead. Futhermore, due to the increase in the complexity of the input space of conditional GANs to handle all classes in one model, they require more training iteration. 5000 training iterations were used for all conditional models and 1000 for others. It is only different learning parameter between multi-GANs and single-GAN architectures.



(a) A heat map from a record of activity "R1_work_at_computer"



(b) A heat map from a record of activity "R1_work_at_computer"

Figure 4.3: Heat maps showing how sensor events distributed in a particular activity. The darker colour indicates the higher number of triggered events of a sensor at a period of time. These two show that the variety of activity from the same class.

4.3.1 Self-verifying

The quality of replayed samples plays an important role to maintain the accuracy of the known tasks, To ensure high standard quality, the samples will be tested with the previous classifier to verify its label. The samples with discrepancy results will be discarded before being used as the replayed. This module will prevent accidental noise that could be generated by GAN. It could also be used as a warning indicator to the generator when it does not behave appropriately.

4.3.2 Gradient Penalty in Wasserstein GAN

There is a growing consensus among GAN researchers that training GAN is a hard problem [11]. Many GAN models suffer a problem that they failed to converge. Figure 4.5 shows changes of WGAN's discriminator and generator loss while it was being trained. It clearly shows that it did not converge properly which might be because of the unbalance between the generator and discriminator. By experimenting on the smart home dataset, adding gradient penalty term gives a higher accuracy on average so I decided to apply this term in all WGAN models for the latter experiments.

4.3.3 Increase stability by introducing instance noise

A different technique to increase stability to GANs is to introduce instance noise the inputs of the discriminator while it is training [2, 21].

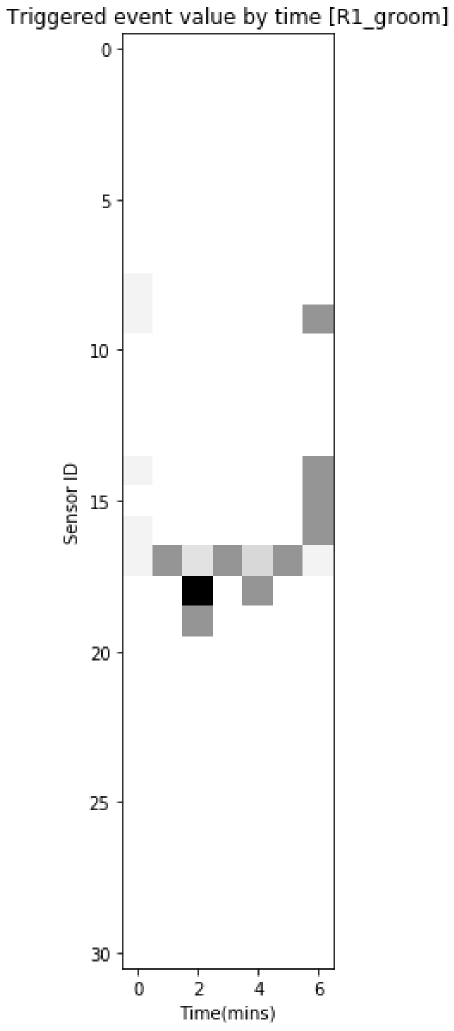
Because JS-divergence and KL-divergence become meaningless when the probability mass between actual data and fake data do not overlap. It leads to instability of GANs [2]. However, instance noise will smoothen those distributions so they become overlapped and stabilise the training process. In the implementation, the volume of instance noise will be damped down to zero by each epoch. A similar technique as proposed recently by Sajjadi et al. [29]. Instead of manually adjusts the volume of the noise, they used another network that automatically adjusts itself as a lens [29]. Due to its complexity, only simple implementation of Gaussian instance noise was used.

4.4 Comparison Method

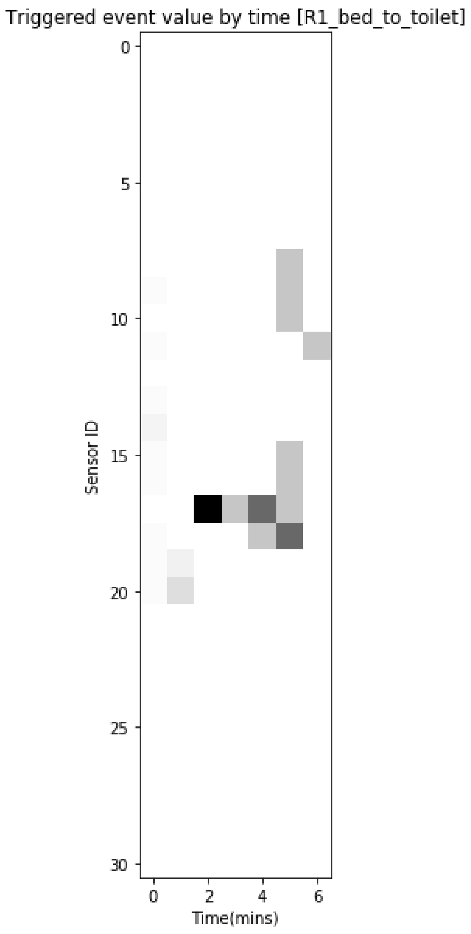
Four methods were implemented.

- **Offline**: The model was trained all at once using data from all tasks so far. It can be seen as a multi-task learning model which the whole network is shared to all tasks. It considers being an ideal model for the experiment. All metrics will be calculated using this model as the expected model.
- **None**: The model was sequentially trained one task at a time without any data from the previous tasks. It will be treated as lower bound for the experiment.
- **Exact Replay**: The model was sequentially trained on a new task and all data from previously learned tasks. It represents a model with the best quality of replayed samples.
- **Generative Replay**: It is my proposed model. The model was sequentially trained using a new set of data and generated data from the generator.

For a fair comparison, the same neural network architecture was used for all methods above. The network configuration will be discussed in the later section.



(a) A heat map from a record of activity "R1_groom"



(b) A heat map from a record of activity R1_bed_to_toilet

Figure 4.4: Heat maps showing how sensor events distributed in a particular activity. The darker colour indicates the higher number of triggered events of a sensor at a period of time. These two show the similarity pattern that can be observed even from two different class.

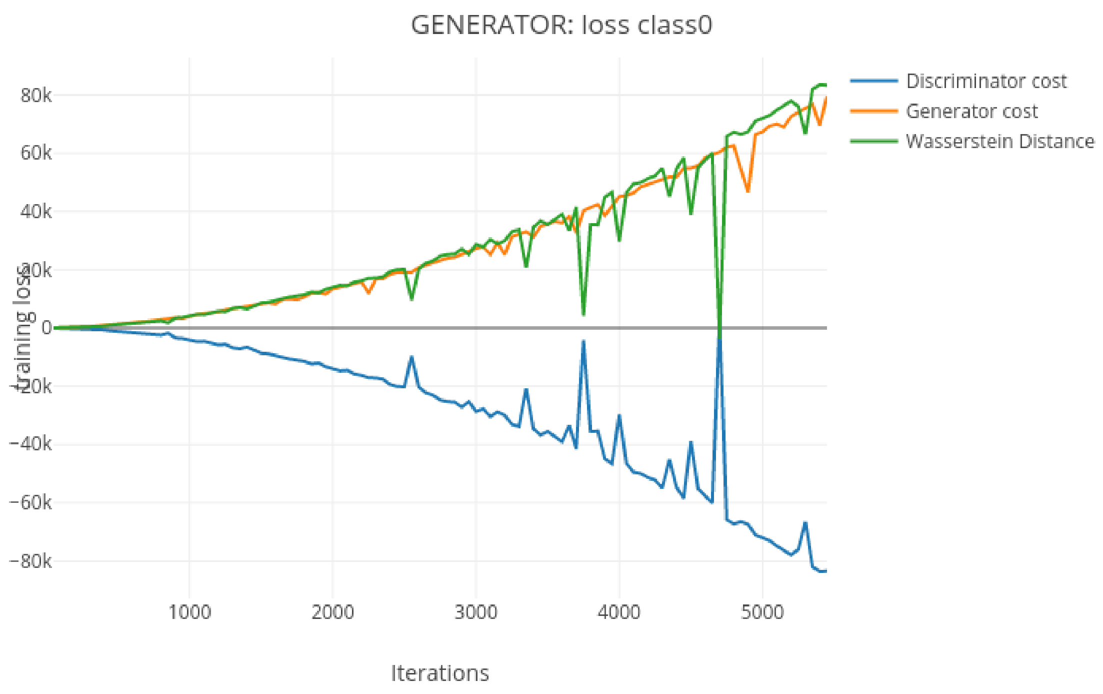


Figure 4.5: Changes in loss value of WGAN's discriminator and generator while it was being trained.

Chapter 5

Evaluation

This chapter provides extensive evaluations of the framework. The evaluation is divided into three parts. The first part presents how GANs were evaluated in terms of effectiveness and instability. The second part presents experimental evaluation to assess how well the framework can incrementally learn and analyse the effects the hyperparameters of the model have on the overall performance. And, lastly, a different dataset was trained to demonstrate the generalisability of the developed approach. Different metrics were set up and used to compare to all methods and also to existing literature. All three parts aim at providing comprehensive performance of the framework and discussion of the way future work could be done.

5.1 Evaluate the generators

To evaluate the generators, 10 different orders of tasks were trained. The results from each task sequence will be compared with the offline model trained from the same task sequence. All measurements were done after all tasks were trained. The average and standard error were reported.

The next section, I will introduce three metrics that were used to evaluate how GANs perform. Their strengths and limitations will also be discussed.

5.1.1 Measuring the quality of GANs

It is difficult to qualify the quality of the generative model. Because the majority of research on GAN usually works with images, the evaluation is dominated by visual inspection and qualitative description which is subjective and possibly misleading. Measurement of dissimilarity between actual and generated probability distribution becomes a widely accepted metric in the literature. Kullback-Leibler divergence (KLD) and Jensen-Shannon divergence (JSD) are often used in theoretical studies. Many GAN papers use various of them as a loss function. But Xu et al. pointed out that those divergence cannot be used as the GAN's quality measurement because both divergences assume that the input distributions need to be known and well-defined [38]. This could not apply to GANs because its distribution is typically unknown and only finite samples can be observed.

Xu et al. also suggests three sample-based metrics for GANs evaluation, namely, Inception score, Kernel MMD and 1-Nearest Neighbor score. Despite their paper mainly considers metrics for visual models, these metrics can be adopted with only minimal changes so as to make them work with non-visual generators.

Inception score is widely adopted in the literature. It evaluates GANs by measuring certain properties of generated results with the pre-trained model on the ImageNet called the Inception network [31]. To make it works with my generators, the inception network was replaced by the offline classifier.

The inception score is defined by the average KL divergence (the expectation of the log difference between two probability distributions) between the probability distribution of a certain label y recognised by the Inception model given an input x – $p(y|x) : x \sim P_g$ and the marginal distribution obtained from all the samples – $p(y)$. The score is computed by the equation below.

$$IS(P_g) = exp(E_{x \sim P_g}[KL(p_M(y|x)||p_M(y))] \tag{5.1}$$

Higher $p_M(y|x)$ indicates that the generated vectors can be recognised by the offline model so it resembles real data. The term $p_M(y)$ indicates how variety the output is so it will gain higher $p_M(y)$ if it produced evenly distribution of classes. The IS score will get higher if the both conditions are true. Even though the Inception score is a reasonable metric to measure with the quality and diversity of generated samples but it fails to observe mode collapsed phenomenon in GAN – when the generator can only produce limited varieties of samples [4]. In addition, by changing the inception network to the offline classifier, the interpretation of the results might not be comparable with other studies but it is intepretable in this context.

Kernel MMD (Maximum Mean Discrepancy) represents the dissimilarity by the distance between samples drawn independently from two distributions P_g and P_r for some kernel function k. It will be computed by the following equation.

$$M_k(P_g, P_r) = E_{x,x' \sim P_g}[k(x, x')] + E_{y,y' \sim P_r}[k(y, y')] - 2E_{x \sim P_g, y \sim P_r}[k(x, y)] \tag{5.2}$$

Xu et al. show that it is really sensitive to kernel used and the sample space. However, the results from their experiments also shows that Kernel-MMD is the most satisfiable measurement due to its ability to identify several problems in

GANs and its low computational complexity [38]. Following the standard setting, the Gaussian kernel will be used in this project.

Lastly, 1-Nearest Neighbor Score is computed by the leave-one-out accuracy of a 1-nearest neighbour classifier trained. The model will be trained with positive samples from true distribution P_g and negative samples from generated distribution P_r. It asserts that the classifier should yield about 0.5 accuracy when GANs achieve the high quality results because the average distance at any given point to both distributions are about the same when two distributions are close and overlapped. It can also indicate the cases that P_g is overfit to P_r (in extreme, GANs can only re-generates the exact input) by observing when the score is lower than 0.5 and it will go zero when $P_g = P_r$. TPR and TNR can also be used to indicate when mode collapse occurs [38]. Low TPR reflects that a majorty of real samples are surrounded by generated samples indicating that the generator can captures different modes of real data. In contrast, high TNR indicates that the generated samples are surrounded by samples from the same class with a few mode centers pointing that mode collapse occurs.

Perhaps the most serious disadvantage of this score is how 1-NN classifier determines distance. The choice of feature space used become critical. Theis, Oord, and Bethge show that Euclidean distance which is normally used by 1-NN classifier is significantly sensitive to small changes especially in higher-dimensional data. They show that a couple of visually identical images can result in a large Euclidean distance [32]. However, because the dataset used here has only 71 features which is a relatively small number compared to an image dataset. The data also be normalised before it was used. Therefore the severity of the problem should be less than the theoretical concern.

5.1.2 The Effectiveness of GANs

Three metrics that I introduced in the previous section were used to measure the dissimilarity between generated data distribution P_g and actual data distribution P_r – Inception score (IS), Kernel-MMD, 1-NN Score(Acc, TPR, and FPR). They will provide the numerical measurement of the sample quality of the generators. Three additional metrics were also used to assess how efficient it is to solve the activity recognition task – test accuracy from the offline model (Acc_{off}), training time ($T_{train}(s)$) and model size.

	IS	k-MMD	1-NN			Acc_{off}	$T_{train}(s)$	Model size
			Acc	TPR	FPR			
Multi-GANs with GAN	**9.846± 0.096**	0.109 ± 0.005	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	**0.891 ± 0.018**	**191.816± 2.281**	1.49 Mb
Multi-GANs with WGAN	9.634 ± 0.206	**0.108 ± 0.006**	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.882 ± 0.024	400.615 ± 3.998	1.49 Mb
Single-GAN with CGAN (1000 epochs)	6.769 ± 0.337	0.266 ± 0.010	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.697 ± 0.047	234.455 ± 1.710	0.15 Mb
Single-GAN with CWGAN (1000 epochs)	1.819 ± 0.293	0.955 ± 0.103	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.256 ± 0.119	372.805 ± 3.642	0.15 Mb
Single-GAN with CGAN (5000 epochs)	9.469 ± 0.267	0.151 ± 0.012	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.875 ± 0.021	1434.469± 1.498	0.15 Mb
Single-GAN with CWGAN (5000 epochs)	7.892 ± 0.438	0.253 ± 0.010	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.831 ± 0.042	3081.804± 14.606	0.15 MB

Table 5.1: Result for each model

The results of the experiments with different model architectures show that multi-GANs outperform the others (see table 5.1). It seems possible that it is because there is no interference between GANs while it was being trained but, as a trade-off, its model size linearly increases by the number of learned classes.

Moreover, it also suggests that Wasserstein loss function had weak positive effects on the generators. With the same number of training iterator, they took approximately two times longer training time. Despite the theoretically proof shows that an equilibrium does exist for a Wasserstein GAN but that does not guarantee that the model has to converge to the equilibrium [3]. In the experiment, it was often found that WGAN had worse results.

Given the same training iteration, it can be seen that both single-GAN architectures perform badly when they were trained with only 1000 epochs but they gained better quality after 5000 epochs. Arguably, this further justifies my motivation for using more training iteration in these conditional models. However, because of the additional iteration, these took more than 7.5 to 16 times longer than the fastest model. This will be a major concern of adopting a single-GAN model in practice.

Furthermore, the 1-NN score clearly shows that mode collapse occurred in all models. It indicates that the generators can only produce a certain type of given data. The mode collapse is also visualised by T-SNE technique (see figure 5.1). The generated samples are clustered together so a simple classifier can easily be built to separate generated from the actual data. It could reduce the generality of the classifier especially in a situation that has a high variety of activity pattern is observed. However, the results would seem to suggest that the mode collapse did not cause a significant accuracy drop. It is likely that the GANs could capture enough properties that are sufficiently used to solve the tasks. It is arguably not important to mimic dataset perfectly.

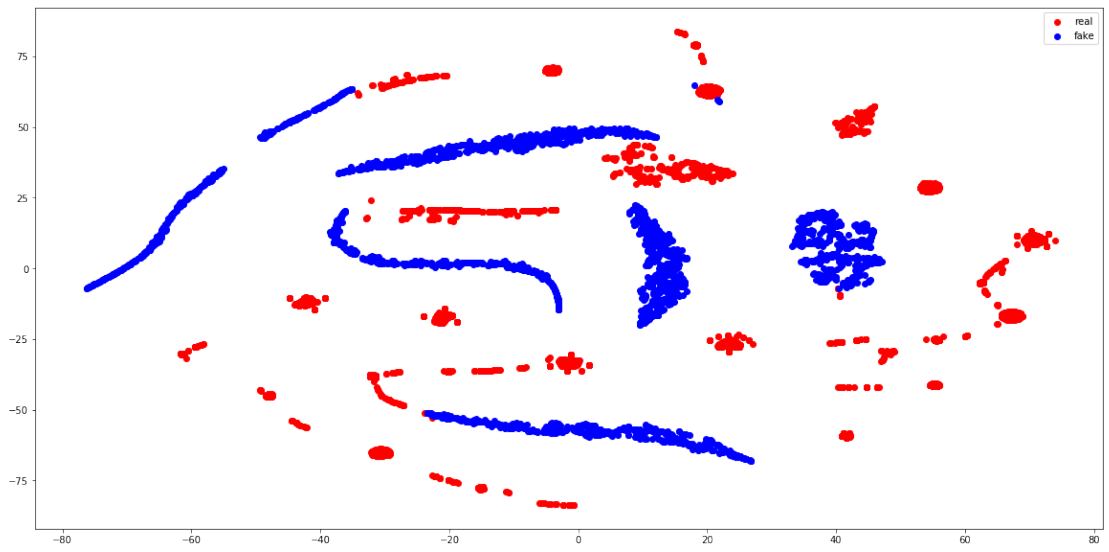


Figure 5.1: T-SNE plot between real data and generated data

5.1.3 The Instability of GANs

Instability might be a major drawback of using GANs in the continual learning model. Figure 5.2 shows that the divergence between discriminator and generator could be observed in several classes. It might lead to a more serious problem when it is used for life-threatening activities such as detecting a fall of a elderly person. At the time of writing, there is no way to guarantee the stability of GANs.

Considering the complexity of the input space, both conditional GANs seems to suffer more serious instability. The example from Figure 5.2 confirms this assumption. The CGAN's generator loss was wildly fluctuating at all classes.

Surprisingly, this stability issue did not lead to a major decrease in the performance of the framework. The instable GANs could still have moderately good accuracy. However, it might contribute to the mode collapse that was explained in the previous section. Arguably, the accurate samples could boost up the accuracy. The results of the exact replay model in the next section have already confirmed this assumption (see figure 5.2). More extend study should be done to investigate thoroughly how the instability affects the continual model.

(a) vanilla GANs (b) WGANs (c) CGAN (d) CWGAN

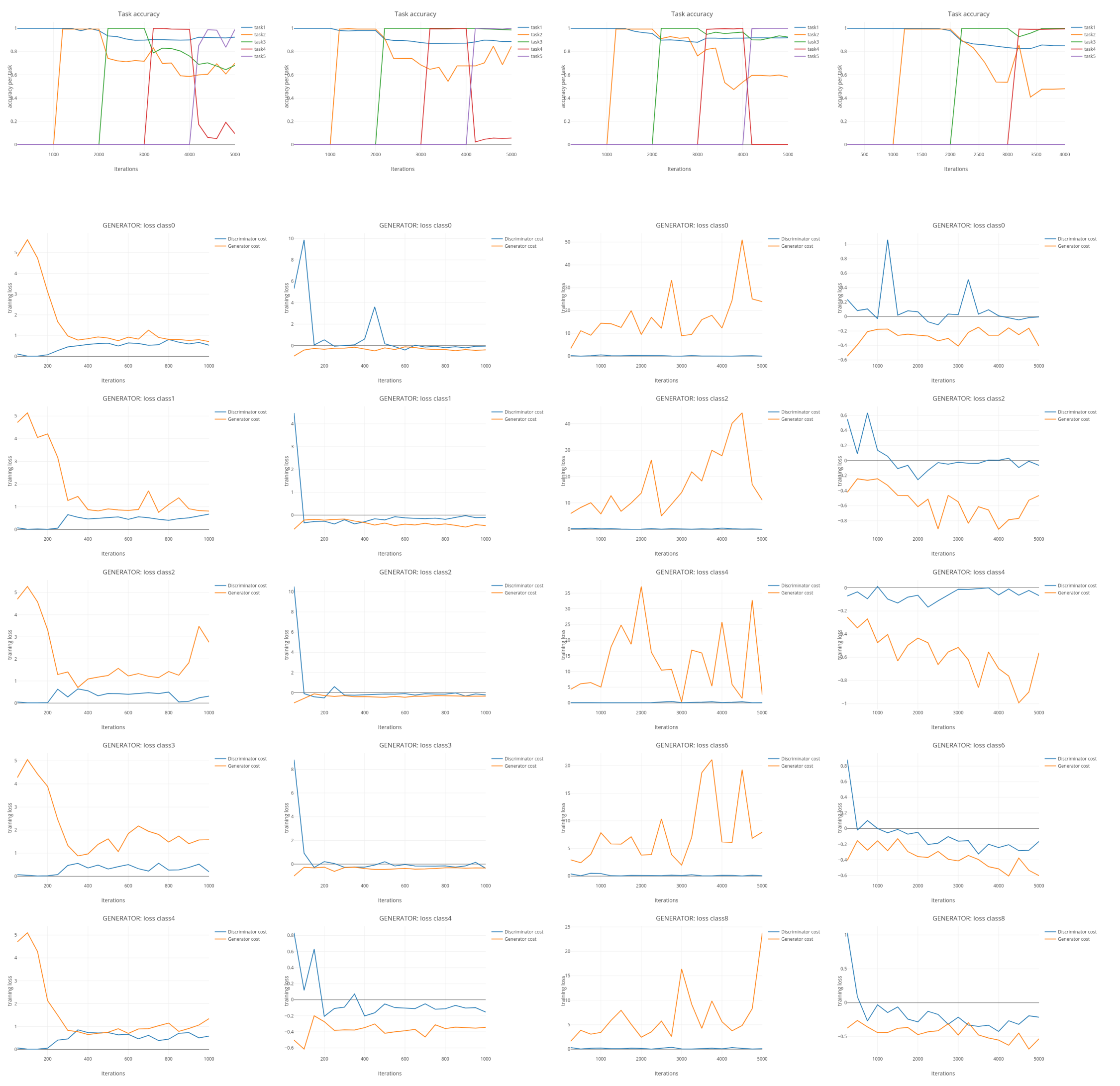


Figure 5.2: Line charts show an example of how model accuracy changes and how GAN's generator and discriminator loss change while they were training

To sum up, GANs can effectively capture the key properties of data. Even though the generated data behave differently to the actual data and only a certain group of data can be reproduced, these samples are capable of fooling the offline model resulting in an acceptable accuracy. They are conceivably sufficient to use as replayed data.

The experimental results suggest that two proposed architectures could reach the same level of accuracy but they both have their own advantages and disadvantages in different applications.

An obvious advantage of the multi-GANs models to the single-GAN is the faster training time. It will benefit as an online adaptive model. New classes could be trained shortly after it is introduced. One more advantage is flexibility. Because activity classes are learned individually in different GANs. Extend to this aspect, pre-trained GANs could be used as modules allowing users to select and combine certain classes of activity to build their own personal system. However, scalability is a considerable issue to this architecture. One can argue that adding a neural model does not require significant space but considering a platform for thousand of classes such as a model for recognising hundred of patients in a hospital, it might become a serious problem. Therefore, on a smaller scale such as a system in a household, the multi-GANs still is a better option.

Single-GAN, in contrast, is less efficient. However, given enough training iteration, it can be trained to reach the same level as the multi-GANs models. The unstable are more often observed when training these condition-GANs. The elaborate components are needed to regulate and support the stability of the model. Compensating with those disadvantages, it can be seen that the single-GAN requires significantly less space. The constant memory used could benefit to a limited memory device. The long training time can be avoided by offline training.

Lastly, instability is a concern in practice. There is no guarantee that whether GANs will converge. However, there also is no clear how the instability has negative impact on the performance of the end result but it certainly has a gap between exact replay and the generative replay. Higher quality sample is needed. The fine-tuning parameters might be a possible solution but, in practice, it is impossible to manually fine-tune every time is is being trained. The automated parameter selection might be a key component of the framework.

5.2 Evaluate the classifier

Similar to the previous section, 10 different orders of tasks were learned by the model. Hold-out validation where fixed, disjoint portions of datasets are used for training. Three metrics were calculated to represent how the model continuously learns while also maintain previous knowledge. They are defined by the following equations (with some modifications from the original paper [14]).

$$M_{base} = \frac{1}{N} \sum_{i=1}^{N} \frac{\textbf{base accuracy}\ \mathit{at\ training\ session}\ i}{\textbf{ideal accuracy}\ \mathit{of\ task}\ 1} \tag{5.3}$$

$$M_{new} = \frac{1}{N} \sum_{i=1}^{N} \frac{\textbf{new accuracy}\ \mathit{at\ training\ session}\ i}{\textbf{ideal accuracy}\ \mathit{of\ task}\ i} \tag{5.4}$$

$$M_{overall} = \frac{1}{N}\sum_{i=1}^{N}\frac{\textbf{accuracy}\ \mathit{of\ task\ i\ at\ the\ last\ training\ session}}{\textbf{ideal accuracy}\ \mathit{of\ task\ i}} \tag{5.5}$$

Given N as a number of learned tasks so far, M_{base} measures how the model retains the knowledge by measuring the average of the ratio between the accuracy of the first task *(base accuracy)* to the expected accuracy which is represented by the accuracy of the same task from the offline model *(ideal accuracy)*. M_{new}, in other hands, indicates how well the model acquire the new knowledge. The test accuracy of a task that was immediately learned at each session *(new accuracy)* was used and divided by the ideal accuracy of the same task. Lastly, $M_{overall}$ presents the overall result of the model determined by the average ratio between the accuracy of each task at the final training session and its expected accuracy. These three metrics represent stability, plasticity, and overall performance of the system. A clear explanation can be seen in the diagram in figure 5.3.

Four experiments were introduced including a comparison between continual learning ability in all four models, an experiment analysing how each learning parameters affects the overall results, an experiment investigating two strategies used to draw samples and lastly an analysis of how proposed additional components influences the framework's result.

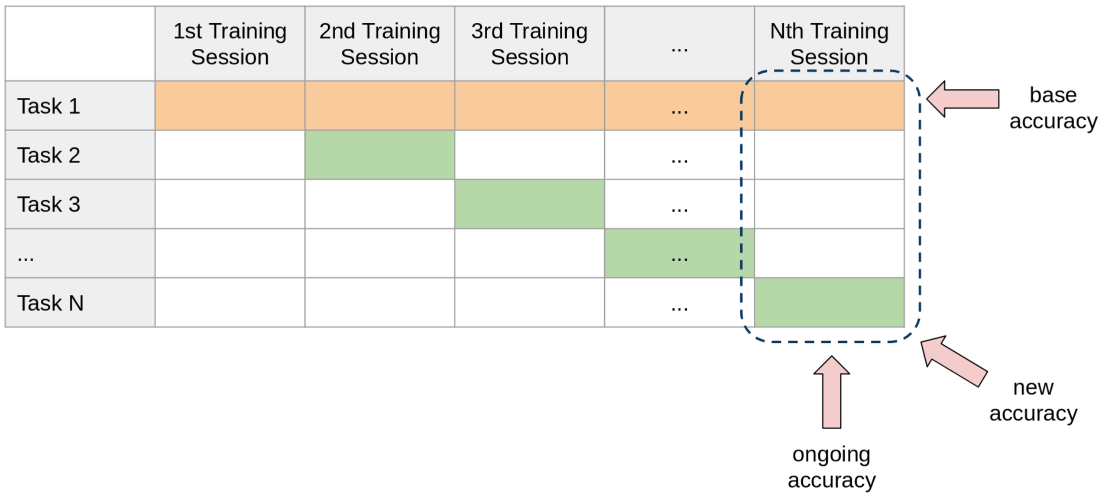


Figure 5.3: Visual representation of M_{base}, M_{new}, and M_{now}

5.2.1 Continual learning ability

Results are summarized in Table 5.2. As expected, the exact replay model surpasses the others. Moreover, its result values also had more than one in all metrics indicating that it is able to defeat the offline model. There are two likely explanations. It might be because of the simplicity of the training process. Training with only two classes at a time might be easier than training all classes together. This improved accuracy could also come from transfer learning from previously known knowledge.

However, the generative models still performed well in an acceptable range with only slightly different results. The M_{base} shows that both multi-GANs models worked effectively at retaining the known knowledge. The less accurate results were found in the single-GAN models. Since the difference has not been found elsewhere it is probably due to the number of GAN itself. With only one GAN used, they are more susceptible to the inference and instability problem. There is also no significant difference in term of plasticity as can be seen by about the same M_{new} for all models.

	M_{base}	M_{new}	$M_{overall}$
None	0.300 ± 0.046	1.222 ± 0.032	0.265 ± 0.028
Exact Replay	1.371 ± 0.170	1.201 ± 0.029	1.049 ± 0.010
Generative (multi-GANs)	0.903 ± 0.080	1.222 ± 0.031	0.776 ± 0.033
Generative (multi-WGANs)	0.924 ± 0.094	1.222 ± 0.032	0.801 ± 0.034
Generative (single-CGAN)	0.884 ± 0.111	1.224 ± 0.031	0.725 ± 0.041
Generative (single-CWGAN)	0.713 ± 0.047	1.223 ± 0.031	0.654 ± 0.037

Table 5.2: Continual learning performance for each model

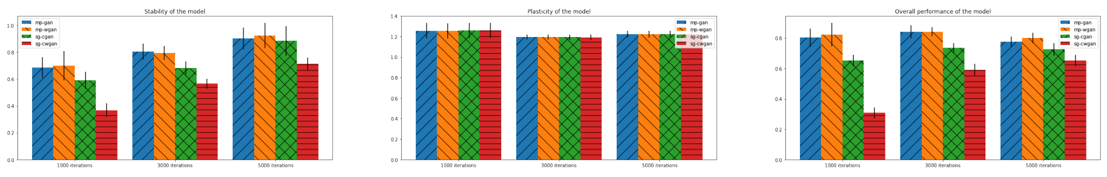
But in the overall performance, the generative model reached only about 65-80 % of the ideal accuracy. Our best model so far is the generative model with multi-WGANs architecture which achieved 80% of the offline performance. Comparing to the recent study by Kemker et al., my models still did not surpass their best model. One simple explanation might come from the different issues that only occur in the smart home domain but not in MNIST dataset that they used. The interclass similarity, class imbalance, and data variability have a more severe negative influence in this domain. The severity of these problems could be confirmed by the higher standard error in our experiments (ten times higher than the stand error in [14]). These significantly high standard errors also implied that the framework might not be consistency and robustness but it is still in an agreeable range. Interclass interference and other potential causes should be addressed seriously in future work.

5.2.2 Hyperparameters

Choosing a set of optimal hyperparameters for a model still is an open problem in machine learning. In my framework, three sets of neural networks are used in each model causing exploding the number of hyperparameters (at least three times higher than a typical learning model). Finding the optimal model from gridsearch is impractical in this circumstance. Therefore, only two crucial parameters were evaluated here (generator training iterator and the number of hidden units per layer) to give an overview of how these learning parameters affect the models.

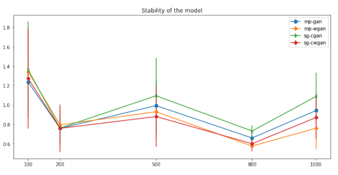
Figure 5.4 shows a comparison when the generators were trained with 1000, 3000 and 5000 iteration. The training iteration clearly has a positive effect on the stability of the classifier. However, at a certain point, the overall performance drops rather increases. This backward could be attributed to the overfitting. Only the WCGAN generator that had the opposite trend indicating that it is still not the optimum. It could be trained with more iteration. This also emphasises early argument when I decided to use more iterator for WCGAN.

Regarding the number of hidden units per layer, the frameworks were tested with 5 configurations including 100, 200, 500, 800 and 1000 hidden neurons per layer. As shown in 5.5, all four model shows similar trends. The plasticity and overall performance of the models increase but suddenly decrease at a certain point due to the over-complexity of the model. In ohter hand, there is no clear pattern to explain how stability change as the number of hidden per layer increases. The number of hidden units hypothetically should have a strong correlation to retainability and overall capability of the system because the more hidden neurons, the more learning space the models can have. Despite the results did not support the hypothesis but I strongly believe that because of the fixed number of training iteration, the larger models had not been trained properly. However, it is difficult to test due to time constraint.

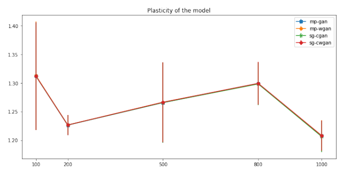


(a) M_{base} (b) M_{new} (c) $M_{overall}$

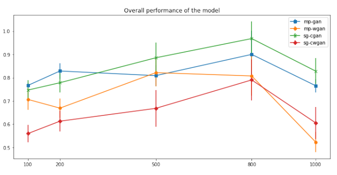
Figure 5.4: Bar charts show stability, plasticity and overall performance (M_{base}, M_{new}, and $M_{overall}$) in different generator training iteration



(a) M_{base}



(b) M_{new}



(c) $M_{overall}$

Figure 5.5: Line charts show stability, plasticity and overall performance (M_{base}, M_{new}, and $M_{overall}$) of the models with 100, 200, 500, 800 and 1000 hidden units per layer

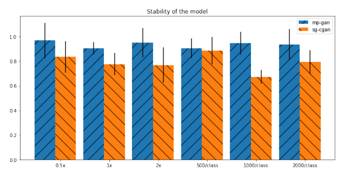
5.2.3 Important of replay samples

For completeness, another test was conducted to explore how important of the replayed samples, how many samples the system needed to maintain the overall accuracy and how the system improves from completely forgetting model to the generative model. Two strategies used to draw samples were tested. The first is to fix the number of samples. Given P as the fixed parameter. The generator will draw $P \times |trainingdata|$ samples throughout all training session. The second strategy is to dynamically draw samples by fixed number of samples per known class. Given Q samples per class, the R^th training session. The generator will draw $2 \times Q \times R$ samples in that session (assume that there are 2 new classes per session).

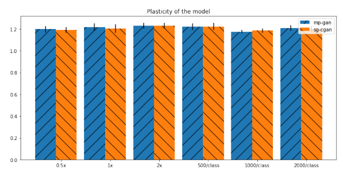
With $P = 0.5, 1, 2$ and $Q = 500, 1000, 2000$, Figure 5.6 shows that the number of samples did not significantly affect the performance of the multi-GANs model. In contrast, there are vast differences for the single-GAN. One reason behind the differences in the single-GAN but multi-GAN might due to how the single-GAN's generator is trained. Because the generator in single-GAN is trained by both replayed and actual data but the generators in multi-GANs are trained only with actual data.The quality and the number of replayed samples, therefore, have a double impact on the single-GAN models from both the classifier and the generator.

5.2.4 Component Sensitivity

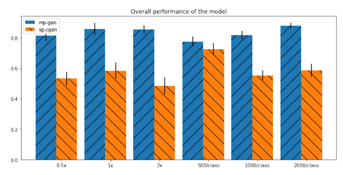
This experiment aims to validate that my framework requires all its key design components. Each component will be enabled one-by-one showing how it affects the final results. There are four components considered here; self-verifying, oversampling, classifier regularisation, and instance noise in GANs. Without loss of generality, the only multi-GANs model was evaluated.



(a) M_{base}



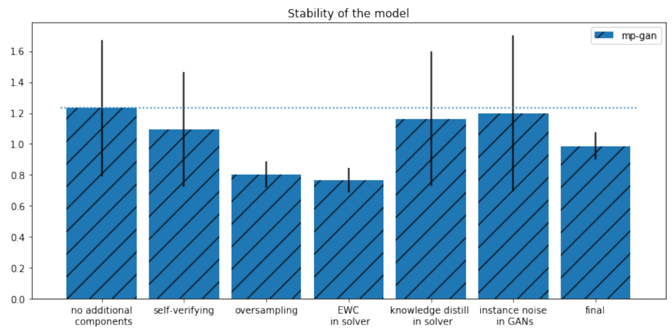
(b) M_{new}



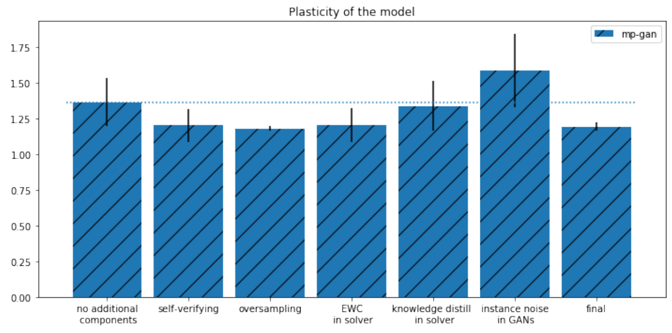
(c) $M_{overall}$

Figure 5.6: Bar charts show stability, plasticity and overall performance (M_{base}, M_{new}, and $M_{overall}$) when the models were trained with fixed number of samples throughout all training session (0.5, 1 and 2 times to number of training data) and with very number of samples depending on the number of samples per known class (500, 1000 and 2000 per class)

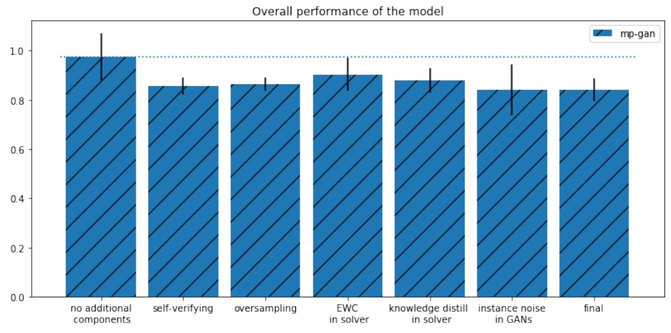
The results unveil a number of interesting effects on my modelling approach (see figure 5.7). First, the models reached remarkably high stability, plasticity and overall performance scores at 1.23, 1.36 and 0.97 even without any additional components. Its performance was close to the exact replay model which is seen as the upper limit of the model. However, the standard error indicates that it also has high inconsistency. Second, oversampling has the greatest impact on reducing this variability. In the same time, it also has a high negative impact on the ability to retain the previously known knowledge of the model. In contrast, self-verifying did not work well in term of reducing the variability of the stability but it worked well regarding the overall performance with minor degrade on the stability. Third, two regularisation approaches were tested; EWC and knowledge distillation. The results clearly show that the latter is more effective. Interestingly, any positive effect on all metrics when the regularisations were used. They both have failed to gain higher accuracy opposing to the results by the previous study [34]. These raise doubts about adding these components to the system. Lastly, it can be seen that the instance noise is the only part that improves the plasticity of the model. The data reported here appear to support the assumption that the instance noise improves the stability of the GANs and also increase the quality of the generated samples. Consequently, less noisy samples may cause lesser interference while the model is learning. These have an indirect impact on how the model learns a new task. However, putting all these components together is non-trivial. The intervention might occur as can be observed. Finding the best combination could be done in future work.



(a) M_{base}



(b) M_{new}



(c) $M_{overall}$

Figure 5.7: Bar charts show stability, plasticity and overall performance (M_{base}, M_{new}, and $M_{overall}$) when only one component was enabled at a time while the models were being trained

5.3 Evaluate generality

Lasty, to demonstrate the generalisability of the framework, it was evaluated on another different dataset – PAMAP2 [26]. Differently, from sensor event dataset from CASAS, PAMAP2 datasets contain continuous sensor readings from accelerometers, gyroscopes, magnetometers that attached to the hand, chest and ankle of 9 subjects. However, due to the difficulty of feature extraction procedure from the raw data, extracted features from Wang et al. were used instead [36]. It consists of 7312 records of 245 features. Data scaling was applied before running the training procedure. In addition, the neural networks were enlarged up to 1000 hidden units per layer due to a significant large number of features comparing to 71 features in CASAS data.

Table 5.3 shows that the single-GAN model could reach up to 78% and 68% of the ideal accuracy. Even though the framework has unsatisfactory results but considering that this is just an initial observation without fine-tuning. Due to the fact that this dataset has three times more features than the baseline. The original parameters such as the number of hidden layers, the number of training iterator and number of replayed samples could not be suitable here. Fine-tuning techniques should also be done. This project provided a concept applying continual learning to the smart home domain. Due to the small size of the CASAS dataset. Simple fully connected neural networks are sufficient to the problem but for the larger dataset such as PAMAP2, deep learning techniques could also be applied. It could further gain better performance to the model.

	M_{base}	M_{new}	$M_{overall}$
Exact Replay	1.227 ± 0.061	1.135 ± 0.014	1.081 ± 0.015
Generative (multi-GANs)	0.663 ± 0.061	1.153 ± 0.014	0.537 ± 0.025
Generative (single-CGAN)	0.780 ± 0.064	1.158 ± 0.014	0.681 ± 0.031

Table 5.3: Continual learning performance trained with PAMAP2 dataset

Chapter 6

Conclusion

In this chapter, it aims to summarize the learning outcomes of this three-month project. A brief overview objective, architecture and the strengths and weaknesses of the project are presented. And the possible future work will also be discussed.

The smart home is proposed to improve people's quality of daily life, especially for elderly and disabled people. However, the smartness behind this system is still limited. Despite, there are several breakthroughs in human activity recognition but one major problem is still remain; those solutions cannot adapt when changes occur. Continual learning is trying to tackle this problem by adding a new ability to the models allowing them to be able to incrementally and continuously learn a new task.

This project is to design and implement a continual model for human activity recognition. The generative replay technique was implemented. The model consists of two agents. One is called "generator " trained to reproduce the previously learned datasets. Its samples will be drawn to replay those pieces of data while another agent called "classifier" is being trained to classify activities. The previously learned knowledge will be captured by the classifier though the actual data and the replayed samples. In this project, the classifier and the generator was implemented by a fully-connected neural network and GANs respectively.

The core contribution of this work is summarised as follows.

1. To the best of my knowledge, this is the first time that the continual learning approaches are integrated into the smart home domain. Four additional components; self-verifying, oversampling, classifier's regularisation and instance noise in GANs were introduced to improve the stability and plasticity of the system. The best model can reach up to 80.1% of the expected accuracy and can maintain 92.4% accuracy of the previously learned task. Even the result suggests that my framework did not significantly improve over the state-of-the-art comparing to other studies in image recognition. This demonstrates the practical value of the proposed method in a new domain – activity recognition.

2. This project provides a comprehensive evaluation of the framework. Two architectures along with four GAN models are developed; multi-GANs with vanilla GANs, multi-GANs with WGANs, single-GAN with CGAN, single-GAN with CWGAN. Seven experimental evaluations were undertaken to validate the design of the different levels of the framework. These are divided into three sections including an evaluation to assess the effectiveness and instability of the generator, a classifier's evaluation to analyse its continual learning ability and examine its hyperparameters, lastly, an evaluation demonstrating the generalisability of the framework. These can be used as a baseline for further study.

3. New challenges specifically to continual learning model in human activity recognition have been addressed. Apart from catastrophic forgetting, the results show that the model critically suffers from interclass interference and GAN instability. Due to the former, learning a new task could potentially interfere with other tasks. It is similar to catastrophic forgetting but it comes from the nature of the data, not by training procedure. In addition, the experiment shows that GAN instability occurs fairly often during training. Even it did not major decrease the overall performance but it prevented the model to reach higher accuracy as can be observed in the exact replay (a model representing the upper limit). In addition, specific to the generative replay technique, the exploding number of hyperparameters is also another concern. This increase the difficulty to find the optimal parameters of the model.

Furthermore, common difficulties in the smart home domain are needed to concern such as activity variability where the same activity may be performed differently by different individuals, class imbalance where only a few activities occur often while most activities occur rather infrequently and noisy data where several sources of noise could be introduced such as by sensor failure and activity ambiguity.

In general, my proposed approach can learn incrementally to recognise new activities without forgetting in high accuracy. The accuracy and robustness of the system, however, can be further improved.

6.1 Future Extensions

As the results of this project have shown new challenges in the field. Each issue should be thoroughly investigated. Interclass interference could be solved more efficiently by XdG or other regularisation technique [20]. Automated parameter selection could be employed to leverage the present structure. Various GANs could be explored to provide the best fitting to the problem.

At this point, several limitations exist in the current design. Considering to deploy the model as a lifelong model, a fixed-structure model might have a limitation on the number of tasks it can learn. Perhaps, self-organizing neural networks or neural network with the ability to transfer knowledge could be a key additional component. Moreover, as a lifelong model, managing a large number of GANs in the multi-GANs model could be a serious issue. In contrast, the single-GAN model might quickly become unstable and only producing a low quality of samples. It might have a cleaver way to over these disadvantage by combining these two architectures as a hierarchy model. It could benefit from higher accuracy/stability from multi-GANs model and less storage required from multi-GANs. Lastly, one limitation could still come from changes. Even this approach could incrementally learn by it is still unclear whether the proposed model can manage when the learned activity is gradually changing. One assumption I made so far is all tasks are unchanged, clear and well-defined. The proposed model here might no longer apply. A new training procedure that allows a certain class to evolve is needed.

Bibliography

[1] Martin Arjovsky and Leon Bottou. “Towards Principled Methods for Training Generative Adversarial Networks”. In: *5th International Conference on Learning Representations*. 2017.

[2] Martin Arjovsky and Leon Bottou. “Towards Principled Methods for Training Generative Adversarial Networks”. In: (Jan. 2017). arXiv: `1701.04862 [stat.ML]`.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks”. In: *International conference on machine learning*. 2017, pp. 214–223.

[4] Ali Borji. “Pros and cons of gan evaluation measures”. In: *Computer Vision and Image Understanding* 179 (2019), pp. 41–65.

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

[6] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[7] Liming Chen and Ismail Khalil. “Activity recognition: Approaches, practices and trends”. In: *Activity Recognition in Pervasive Intelligent Environments*. Springer, 2011, pp. 1–31.

[8] Liming Chen et al. “Sensor-based activity recognition”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.6 (2012), pp. 790–808.

[9] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[10] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[11] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems*. 2017, pp. 5767–5777.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[13] Wenpeng Hu et al. "Overcoming Catastrophic Forgetting for Continual Learning via Model Adaptation". In: (2018).

[14] Ronald Kemker et al. "Measuring catastrophic forgetting in neural networks". In: *Thirty-second AAAI conference on artificial intelligence.* 2018.

[15] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[16] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[17] Dharshan Kumaran, Demis Hassabis, and James L McClelland. "What learning systems do intelligent agents need? Complementary learning systems theory updated". In: *Trends in cognitive sciences* 20.7 (2016), pp. 512–534.

[18] Zhizhong Li and Derek Hoiem. "Learning without forgetting". In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[19] David Lopez-Paz et al. "Gradient episodic memory for continual learning". In: *Advances in Neural Information Processing Systems.* 2017, pp. 6467–6476.

[20] Nicolas Y Masse, Gregory D Grant, and David J Freedman. "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization". In: *Proceedings of the National Academy of Sciences* 115.44 (2018), E10467–E10475.

[21] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *arXiv preprint arXiv:1801.04406* (2018).

[22] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[23] Neural Information Processing Systems Foundation. *Continual learning Workshop NeurIPS 2018.* `https://sites.google.com/view/continual2018`. Accessed: 2019-6-14.

[24] German I Parisi et al. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* (2019).

[25] Sylvestre-Alvise Rebuffi et al. "icarl: Incremental classifier and representation learning". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 2017, pp. 2001–2010.

[26] Attila Reiss and Didier Stricker. "Introducing a new benchmarked dataset for activity monitoring". In: *2012 16th International Symposium on Wearable Computers*. IEEE. 2012, pp. 108–109.

[27] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[28] Andrei A Rusu et al. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[29] Mehdi SM Sajjadi et al. "Tempered adversarial networks". In: *arXiv preprint arXiv:1802.04374* (2018).

[30] Hanul Shin et al. "Continual learning with deep generative replay". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2990–2999.

[31] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[32] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *arXiv preprint arXiv:1511.01844* (2015).

[33] Niall Twomey et al. "Unsupervised learning of sensor topologies for improving activity recognition in smart environments". In: *Neurocomputing* 234 (2017), pp. 93–106.

[34] Gido M van de Ven and Andreas S Tolias. "Three scenarios for continual learning". In: *arXiv preprint arXiv:1904.07734* (2019).

[35] Jindong Wang et al. "Deep learning for sensor-based activity recognition: A survey". In: *Pattern Recognition Letters* 119 (2019), pp. 3–11.

[36] Jindong Wang et al. "Stratified transfer learning for cross-domain activity recognition". In: *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. 2018, pp. 1–10.

[37] Zhengwei Wang, Qi She, and Tomas E Ward. "Generative Adversarial Networks: A Survey and Taxonomy". In: *arXiv preprint arXiv:1906.01529* (2019).

[38] Qiantong Xu et al. "An empirical study on evaluation metrics of generative adversarial networks". In: *arXiv preprint arXiv:1806.07755* (2018).

[39] Juan Ye. "Lifelong Learning in Sensor-Based Human Activity Recognition". In: *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE. 2019, pp. 2–2.

[40] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3987–3995.