Investigating Continual Learning Strategies in Nueral Networks

Christopher Tam
Western University
ctam86@uwo.ca

Abstract

This paper explores the role of continual learning strategies when neural networks are confronted with learning tasks sequentially. Our investigation measures forgetting with three factors in mind: the type of network architecture used, the continual learning scenario defined and the continual learning strategy implemented. This work is motivated by the theory that complementary learning systems are an essential ingredient in order to achieve lifelong learning, a theory put forward by McClelland et al. in 1998 [1]. Our results show that complementary learning systems provide significant improvements for memory retrieval and consolidation in neural networks, and provide the groundwork for further investigation in to the stability-plasticity dilemma.

1 Introduction

As humans, we have the distinct ability to continually acquire knowledge throughout our lifetime. This ability, aptly referred to as lifelong learning, is enabled by a rich set of neurocognitive mechanisms that allow us to consolidate new information without forgetting previously learnt concepts. This is to say that the process of learning new information does not significantly interfere with our ability to recall old information. In contrast, current machine learning algorithms are unable to process novel streams of data without forgetting previously learnt patterns. This is referred to as *catastrophic forgetting*. The problem of artificial intelligence (AI) systems learning over time by accommodating new knowledge and retaining previously learned patterns is referred to as *continual learning*, and has been a long-standing challenge for machine learning and neural networks [2]. In this project, we investigate continual learning strategies to overcome catastrophic forgetting in neural networks with three factors in mind: the type of network architecture used, the continual learning scenario defined and the continual learning strategy implemented. Specifically, we will be comparing multilayer perceptron to convolutional neural network architectures in the task-IL, domain-IL and class-IL continual learning scenarios using regularization, replay and hybrid continual learning strategies.

2 Background Work

2.1 Catastrophic Forgetting and Continual Learning

A well known constraint for artificial and biological neural systems is the stability-plasticity dilemma [3]. This dilemma expresses the tradeoff between the integration of new knowledge and the stability required in order to prevent forgetting previously acquired knowledge in a neural system. On one hand, excessive plasticity will result in previously encoded information being constantly overwritten as learning takes place, whereas excessive stability will prevent the uptake of learning new data in the system. Between the two ends of the spectrum, McCloskey and Cohen [4] have found that

artificial neural networks (henceforth referred to as neural networks) lean heavily towards plasticity, producing a phenomenon known as catastrophic forgetting [4]. Catastrophic forgetting is defined as the complete or significant forgetting of previously learned information by a neural network when exposed to new information, and has been demonstrated to be a general problem affecting many types of neural networks. For example, Richardson and Thomas [5] showed catastrophic forgetting to be present in a variety of neural networks, from standard back-propagation networks to unsupervised self-organizing map networks to connectionist models of sentence acquisition. In artificial neural networks, catastrophic forgetting occurs when learning is prompted using a different distribution of data than what the network had previously been trained on. In this case, new data instances differ significantly from examples the network had previously observed. The new information causes the network to overwrite the shared representational resources holding the previously learned knowledge, producing a 'catastrophic forgetting' effect of those previously learned patterns. This problem has stood in the way of building lifelong learning systems, capable of learning from a continuous stream of information with information becoming progressively available over time and where the number of tasks to be learned are not predefined [6]. In these systems, it is critical that the accommodation of new information should not be inhibited by the problem of catastrophic forgetting.

2.2 Three Continual Learning Scenarios

Research efforts towards continual learning has garnered plenty of attention and has resulted in a wide variety of experimental protocols being used. This has led to confusion, as some methods are shown to perform well in some certain experimental settings but dramatically fail in others. For example, the elastic weight consolidation algorithm presented by Kirkpatrick et al. [7] claims state-of-the-art performance in it's paper, but showed significant performance issues in the brain-inspired replay approach of ven de Ven et al. [8]. In order to better compare methods for reducing catastrophic forgetting, this report will follow the three distinct scenarios for continual learning proposed by van de Ven and Tolias [9] (Figure 1). This categorization focuses on the problem in which a single neural network is required to sequentially learn a series of tasks. Each scenario is distinguished by the task required and data available at test time, and will be presented in order of increasing difficulty in the following subsections.

Scenario	***Required at test time***
Task-IL	Solve tasks so far, task-ID provided
Domain-IL	Solve tasks so far, task-ID not provided
Class-IL	Solve tasks so far *and* infer task-ID

Figure 1: Overview of the three continual learing scenarios proposed by [9].

2.2.1 Task-Incremental Learning (Task-IL)

In the task-incremental learning scenario, the network is always informed about which task needs to be performed at test time. Given the availability of task identifiers at test time, it is possible to train a network with task-specific components. This enables architectures using a "multi-headed" output layer to be used, allowing the network to share its hidden resources but using task specific output units.

2.2.2 Domain-Incremental Learning (Domain-IL)

In the domain-incremental learning scenario, the network is not informed about which task needs to be performed at test time. However, the network only needs to solve the task at hand and does not need to be able to infer which task group the task belongs to. This scenario is representative of problems where the structure of the tasks is always the same, but the input distribution is changing.

2.2.3 Class-Incremental Learning (Class-IL)

In the class-incremental learning scenario, the network must be able to solve a task as well as infer which class the task belongs to at test time. This scenario represents the most difficult problem in the categorization, and reflects the most common real-world problem of incrementally acquiring new knowledge.

2.3 Strategies for Continual Learning

In 1995, McClelland et al. proposed the Complementary Learning Systems theory [1], which provided a basis for a computational framework modelling memory consolidation and retrieval. At the heart of this framework lies the interplay of episodic memory and semantic memory, which provided important insights into the mechanisms of memory consolidation in the absence of sensory input. Since then, many learning systems have taken inspiration from this framework in order to address the problem of catastrophic forgetting. As a result, there are several general categories which can be used to classify continual learning strategies. We may look to the Venn diagram of some of the most popular continual learning scenarios presented by Lesort et al. [10], for example. For the purpose of this paper, we will limit the discussion of strategies to those categories which are relevant to this project.

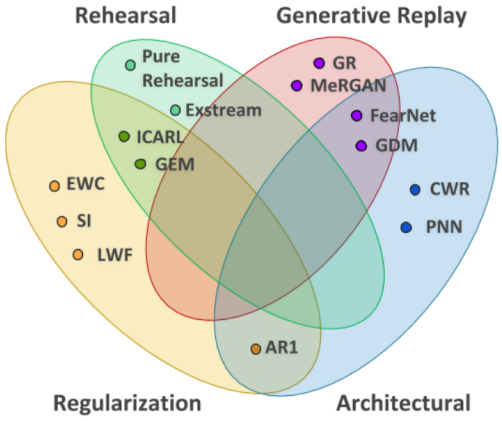


Figure 2: Venn diagram of four of the most popular continual learning scenarios [10].

2.3.1 Regularization Approaches

Several approaches introduce a regularization term into the loss function in order to mitigate the effect of catastrophic forgetting. One way of doing this has been to regularize the network's parameters during training on each new task, so as to constrain the movement of weights in a way that minimizes the amount of "forgetting". This strategy has been implemented in the Elastic Weight Consolidation (EWC) [7] and Synaptic Intelligence [11] algorithms. In both of these methods, estimates for the importance of parameters relevant to previously learned tasks are calculated and parameters are penalized proportional to their relative importance. This has the effect of slowing down learning for parts of the network which are important for previous tasks. Another class of regularization techniques is aimed at preventing activation drift primarily through means of knowledge distillation. One instance of this strategy is the Learning without Forgetting [12] algorithm, which generates pseudo-data to train on using the input data of the current task and labelling it using the model trained on the previous tasks. The goal here is to prevent the representations of previous data from drifting too far away while learning new tasks.

2.3.2 Replay Approaches

Another approach for alleviating catastrophic forgetting is to store data from previous tasks. Replay methods keep a small number of "exemplars" or generate synthetic representations of the data previously encountered in order to prevent the forgetting of previous tasks. This approach has largely drawn on inspiration from the generative role of the hippocampus for the replay of previously encoded experiences. In 2017, Shin et al. [13] proposed a dual-model architecture consisting of a deep generative model and a task solver. Modelling the replay process of the hippocampus, their architecture sampled training data from previously learned experiences to generate pseudo-data to be interleaved with the data from new tasks. In this way, there was no requirement to explicitly revisit old training samples for experience replay and therefore reduced the cost requirements of working memory. More recently, Lorez-Paz and Ranzato [14] proposed Gradient Episodic Memory (GEM), a replay method which yields positive transfer of knowledge to previous tasks. GEM features an episodic memory storing a subset of previously seen examples from a given task. GEM requires far more memory than typical regularization approaches but has produced better results in single pass settings.

3 Methodology

3.1 Task Protocol

Since the objective of this paper is to explore the differences between the three continual learning scenarios and to comprehensively compare the performances of the continual learning strategies discussed above, we selected a widely used task protocol for a basis of comparison. The permuted MNIST is a variation of the MNIST dataset, consisting of 70,000 handwritten digits from 0 to 9. 60,000 images are used for training, and 10,000 images are reserved for testing. Permuted MNIST differs from MNIST in that new tasks may be generated by permuting the pixels of each digit randomly. To generate the permuted images, the original 28x28 grey-scale images were randomly permuted so that the 784 pixels were randomly rearranged. The pixel values were transformed into a tensor and then normalized to a mean of 0.1307 and standard deviation of 0.3081. As a result, each new task can be considered a ten-way classification problem.

3.2 Model Architectures

To ensure a fair comparison, the same networks were used for all methods. Hyperparameters were tuned using a grid-search for each of the networks and the models yielding the highest average accuracy across all tasks were reported in the final comparison.

3.2.1 Multilayer Perceptron Network

A multilayer perceptron network with one hidden layer of 512 nodes was used. A ReLU nonlinearity was used as well as a dropout layer with a probability of 0.5. The network was optimized using stochastic gradient descent and minimized the multi-class cross entropy loss function. A summary of the multilayer perceptron network used is given in Figure 3.

3.2.2 Convolutional Neural Network

A convolutional neural network with two convolutional layers and two fully connected layers was used. For each convolutional layer, a kernel of size 5 and a ReLU nonlinearity was used. The two convolutional layers produced 6 and 16 output channels, respectively. Each convolutional layer was followed by a max pooling layer. In the two fully connected layers, 120 and 105 hidden units were used. The network was optimized using stochastic gradient descent and minimized the multi-class cross entropy loss function. A summary of the convolutional neural network is given in Figure 4.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 512]         401,920
              ReLU-2                  [-1, 512]               0
           Dropout-3                  [-1, 512]               0
            Linear-4                   [-1, 10]           5,130
================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 1.55
Estimated Total Size (MB): 1.57
----------------------------------------------------------------
```

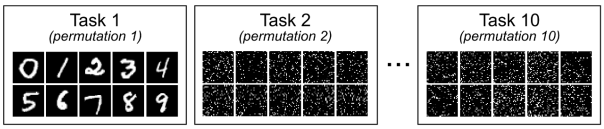
Figure 3: Model summary of multilayer perceptron network.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 6, 24, 24]             156
            Conv2d-2          [-1, 16, 19, 19]           2,416
            Linear-3                 [-1, 120]         155,640
            Linear-4                 [-1, 105]          12,705
            Linear-5                  [-1, 30]           3,180
================================================================
Total params: 174,097
Trainable params: 174,097
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.07
Params size (MB): 0.66
Estimated Total Size (MB): 0.74
----------------------------------------------------------------
```

Figure 4: Model summary of convolutional neural network.

3.3 Continual Learning Scenarios

As discussed in Section 2.2, we are interested in three continual learning scenarios. In order to provide a rigorous comparison, we will evaluate each scenario in a 3- and 10-task scenario. This should give us a better view of how each strategy generalizes to more tasks and will hopefully shine more light on stronger candidates. In this section we clearly define what each of these scenarios look like in the context of permuted MNIST. An overview of the scenarios under permuted MNIST is provided in Figure 5.



Task-IL	Given permutation *X*, which digit?
Domain-IL	With permutation unknown, which digit?
Class-IL	Which digit *and* which permutation?

Figure 5: Permuted MNIST according to Continual Learning scenarios [9].

3.3.1 Task-Incremental Learning (Task-IL)

Under Task-IL, task identities are always known and shown at test time. Given an image and a task identity, the network must be able to correctly identify which digit the image represents. Setting up Task-IL involved assigning a progressive task identity to each of the permutations starting from "0". A network was then consecutively trained on all tasks starting from task 0. At each step of the training loop (after training on each task) an evaluation loop would run to test the network's ability to recall all previously seen tasks. Since the output of the network was always an integer between 0 and 9 inclusive, 10 units were used as output layer for architectures in the Task-IL scenario.

3.3.2 Aside: Tasks vs. Experiences

Before discussing the methodology behind implementing the Domain-IL and Class-IL scenarios, it will be helpful to distinguish between a task and an experience. These notions will make the intuition behind the implementation of these two scenarios much more clear. As shown in Figure 5, task identities are not available at test time in Domain-IL and Class-IL. This is to say that task identities exist in both scenarios, but are merely not presented at test time. Tasks and experiences enable us to train a network on different task identities without exposing those identities at test time. Their primary difference is that networks are trained on experiences and are evaluated on tasks. Consider the following table for Class-IL with three permutations (identified as 0, 1 and 2):

Table 1: Tasks vs. Experiences

Class-IL	
Task	Experience
0	0
0	1
0	2

3.3.3 Domain-Incremental Learning (Domain-IL)

In Domain-IL, task identities are irrelevant at test time. Task identities are neither given nor expected to be inferred by the network. Given only an image, the network must be able to correctly identify which digit the image represents. We've already discussed how tasks and experiences allow us to train a network with different permutations without exposing their identities at test time. Under the permuted MNIST task protocol, a network will alway predict an integer between 0 and 9 (inclusive) for this scenario. The challenge is that for each evaluation step, a random bag of digits from all previously seen permutations will be presented. As such 10 units were used as the output layer for architectures under this scenario.

3.3.4 Class-Incremental Learning (Class-IL)

In Class-IL, task identities exist but are not provided at test time. Given an image, the network must be able to correctly identify which digit the image represents as well as correctly infer which permutation the image belongs to. This additional requirement prompted an adjustment to the labelling process of input and output classes. For each permutation, class labels were generated starting from the integer where the previous permutation ended. For instance, permutation 1 classes were labelled [0...9], permutations 2 classes were labelled [10...19], etc. At test time, these labels can be viewed as representing the m-th digit (ones column) from the n-th permutation (tens column). As a result, 3- and 10-times the number of units were used in the output layer for 3- and 10-task scenarios compared to the previous two scenarios.

3.4 Strategies

3.4.1 Naive

In the naive strategy, a network is sequentially trained on all tasks. With each new task it encounters, the network fine-tunes it's weights to fit the new data. This is where the most amount of catastrophic forgetting occurs, and will be used as a lower bound in our comparison.

3.4.2 EWC

The regularization approach selected for this project was Elastic Weight Consolidation (EWC) [7]. In this strategy, a regularization term is added to the loss function which penalizes changes to parameters estimated to be important to previously learned tasks. The regularization strength of the loss function is controlled by a hyperparameter λ, where

$$L_{total} = L_{current} + \lambda L_{regularization}$$

We used grid search to find the optimal value of λ.

3.4.3 AGEM

For the replay approach, we used the Average Gradient Episodic Memory (AGEM) [15] algorithm. AGEM is similar to the GEM algorithm discussed in 2.3.2, but demands fewer memory requirements by storing the average gradient vector computed from the individual gradients of task loss for all previously seen tasks at each weight update. This is in contrast to GEM, where each task specific gradient vector has to be stored. The AGEM algorithm was tuned using two hyperparameters which varied the number of patterns to store in memory per experience, and number of patterns from each memory sample to consider when computing the reference gradient. The optimal values of these hyperparameters were found using grid search as well.

All strategies were trained for 3 iterations per task using the multi-class classification cross entropy classification loss. Networks were trained for 3 iterations per task using the SGD-optimizer with a learning rate of 0.01 and momentum of 0.9. The hyperparameter search for each of the methods is given in Table 2. We compared the continual learning strategies discussed above using the average test accuracy over all previously seen tasks.

Table 2: Hyperparameter Values

EWC		AGEM	
λ	[0.001, 0.01, 0.1]	Patterns per experience	[10, 30]
		Sample size	[50, 250, 500]

4 Results and Analysis

A tabular summary of final results is given in Figure 6. Interestingly, the MLP networks displayed little forgetting across all scenarios for 3- and 10-class tasks compared to it's CNN counterpart. As shown in 8, the CNN networks suffered greatly when confronted with 10 classes, and performance in the Class-IL scenario dropped significantly for all strategies. Admittedly, the size of the CNN was much smaller than the MLP, containing 4x fewer parameters (100,000 vs 400,000). The degradation in CNN performance might therefore be attributed to its incapacity to learn more tasks due to the limited number of parameters. This is certainly worth investigating in future work. Overall, it seems that AGEM slightly outcompeted EWC. Unfortunately, a hybrid approach was unable to be implemented but it would be worth investigating whether such an approach would improve over both scores or yield results somewhere in between the two individual approaches. Without a doubt, we can see that continual learning strategies offer significant improvements over the naive approach of learning sequential tasks. This supports the work proposed by McClelland et al. [1] in the role of complementary learning systems as a basis for improving memory consolidation and retrieval, and provides groundwork to further investigate the stability-plasticity dilemma.

MLP							
		Task-IL		Domain-IL		Class-IL	
Approach	**Method**	**3-Class**	**10-Class**	**3-Class**	**10-Class**	**3-Class**	**10-Class**
Baselines	Naive (No CL Strategy)	*93.89%*	*85.16%*	*93.80%*	*91.30%*	*93.49%*	*91.30%*
Regularization	EWC	*97.24%*	*96.93%*	*97.22%*	*96.89%*	*96.62%*	*95.26%*
Replay	AGEM	*98.12%*	*97.36%*	*97.51%*	*97.33%*	*97.17%*	*91.15%*

CNN							
		Task-IL		Domain-IL		Class-IL	
Approach	**Method**	**3-Class**	**10-Class**	**3-Class**	**10-Class**	**3-Class**	**10-Class**
Baselines	Naive (No CL Strategy)	*55.74*	*28.90%*	*52.92%*	*31.29%*	*31.40%*	*9.38%*
Regularization	EWC	*89.07%*	*52.12%*	*89.74%*	*52.18%*	*82.86%*	*19.90%*
Replay	AGEM	*95.32%*	*52.45%*	*91.81%*	*56.40%*	*83.25%*	*37.18%*

Figure 6: Table of final results.

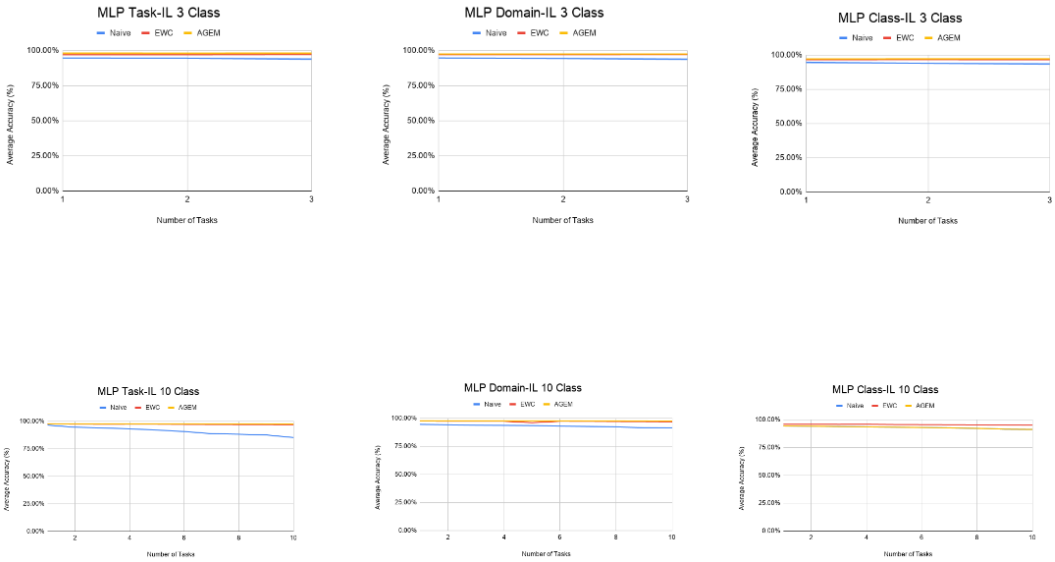


Figure 7: MLP: Average accuracy computer after training on each task for all tasks.

References

[1] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[2] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[3] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.

[4] M McCloskey and NJ Cohen. The psychology of learning and motivation. 1989.

[5] Fiona M Richardson and Michael SC Thomas. Critical periods and catastrophic interference effects in the development of self-organizing feature maps. *Developmental science*, 11(3): 371–389, 2008.

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

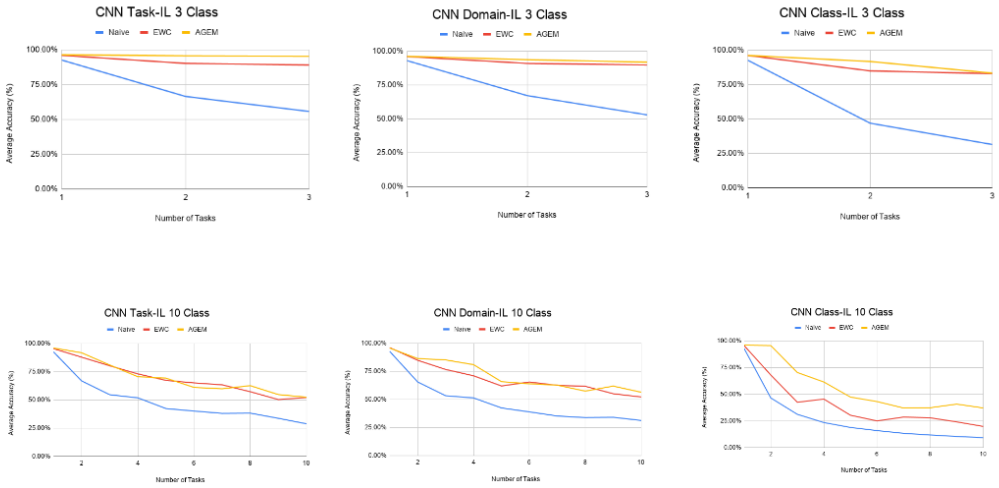


Figure 8: CNN: Average accuracy computer after training on each task for all tasks.

[7] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[8] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020.

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

[10] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

[11] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[13] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[14] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017.

[15] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.