

https://dspace.org/

Evaluating elastic weight consolidation across a taxonomy of continual learning scenarios

Laxton, Gregory Stephen

2025-11
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/50305

Laxton, Gregory Stephen. (2025). Evaluating elastic weight consolidation across a taxonomy of continual learning scenarios. [Master's dissertation, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/50305

Downloaded from a DSpace repository. https://dspace.org/

Evaluating elastic weight consolidation across a taxonomy of continual learning scenarios

---

Greg Laxton
542087

Supervisor(s):
Mr. Devon Jarvis



A research report submitted in fulfilment of the requirements for the degree of MSc Data Science

in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

13 November 2025

Declaration

I, Gregory Stephen Laxton, declare that this proposal is my own, unaided work. It is being submitted for the degree of MSc Data Science at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.



Greg Laxton
542087
13 November 2025

Abstract

The dynamic nature of real-world datasets poses significant challenges for machine learning, especially in continual learning, where catastrophic forgetting occurs as neural networks forget prior knowledge when acquiring new information, known as catastrophic forgetting. Various methods have been investigated in order to overcome catastrophic forgetting, such as Elastic Weight Consolidation (EWC). This study aims to determine the effectiveness of EWC in diverse real-world datasets characterised by different continual learning scenarios. We compare EWC against a base model with no regularisation, models with different continual learning approaches, and a method that incorporates data rehearsal from the previous task. Through experiments and performance comparisons using datasets specifically designed to represent three key iterated learning scenarios—class-iterated learning, domain-iterated learning, and task-iterated learning—this research aims to provide a comprehensive assessment of the capacity of EWC to facilitate continual learning. To ensure a thorough evaluation, these scenarios were carefully selected to include a toy domain, an MNIST-based domain, and a practical real-world domain, reflecting a progression from abstract conceptual tasks to more complex and realistic challenges. This meticulous approach highlights the core contribution of the research, emphasising its relevance and applicability across varying levels of complexity in continual learning. Our findings show that while EWC consistently delivers stable performance across diverse continual learning scenarios, it struggles with adaptability in tasks involving high feature overlap or dynamic shifts. In contrast, Synaptic Intelligence offers greater flexibility but exhibits higher variability, especially when task boundaries are ambiguous or overlap occurs at deeper representation levels.

Acknowledgements

I would like to thank Ronald Laxton for the continued financial support and to my supervisor, Mr Devon Jarvis, for the extensive guidance.

Contents

Declaration i

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Background and Related Work 6
- 2.1 Regularisation Methods . . . 6
  - 2.1.1 Elastic Weight Consolidation . . . 7
  - 2.1.2 Synaptic Intelligence . . . 10
- 2.2 Architectural Method . . . 12
- 2.3 Rehearsal Method . . . 14

3 Research Methodology 16
- 3.1 Task Iterated Learning . . . 19
  - 3.1.1 Toy Dataset . . . 19
  - 3.1.2 Benchmark Dataset . . . 21
  - 3.1.3 Real-World Dataset . . . 22
- 3.2 Domain Iterated Learning . . . 23
  - 3.2.1 Toy Dataset . . . 24
  - 3.2.2 Benchmark Dataset . . . 25
  - 3.2.3 Real-World Dataset . . . 26
- 3.3 Class Iterated Learning . . . 27
  - 3.3.1 Toy Dataset . . . 27
  - 3.3.2 Benchmark Datasets . . . 28
  - 3.3.3 Real-World Datasets . . . 29

4 Results **32**
4.1 Task Iterated Learning . . . 32
4.1.1 Toy Dataset . . . 32
4.1.2 Benchmark Dataset . . . 34
4.1.3 Real-World Dataset . . . 36
4.2 Domain Iterated Learning . . . 37
4.2.1 Toy Dataset . . . 37
4.2.2 Benchmark Dataset . . . 39
4.2.3 Real-World Dataset . . . 40
4.3 Class Iterated Learning . . . 43
4.3.1 Toy Dataset . . . 43
4.3.2 Benchmark Dataset . . . 46
4.3.3 Real-World Dataset . . . 48

5 Discussion **51**
5.1 Performance Evaluation Analysis . . . 51
5.2 Performance and Variability of the Regularisation Methods . . . 53
5.3 Dataset Characteristics vs Learning Scenario . . . 55

6 Conclusion **57**
6.1 Concluding Remarks . . . 57
6.2 Ethical Considerations . . . 59
6.3 Limitations . . . 60
6.4 Further Research . . . 61

A Implementation Details and Technical Contributions **66**

B Hyperparameter Settings **67**

C Results **68**
C.1 Task Iterated Learning . . . 68
C.1.1 Toy Dataset . . . 68
C.1.2 Benchmark Dataset . . . 69
C.1.3 Real-World Dataset . . . 69
C.2 Domain Iterated Learning . . . 70
C.2.1 Toy Dataset . . . 70

C.2.2 Benchmark Dataset 71
C.2.3 Real-World Dataset 71
C.3 Class Iterated Learning 72
C.3.1 Toy Dataset 72
C.3.2 Benchmark Dataset 73
C.3.3 Real-World Dataset 74

Chapter 1

Introduction

Continual learning is the process where a model is incrementally trained on new tasks or data without any explicit retraining of prior tasks [1, 2, 3]. Ideally, when a model is trained on new data, it is able to retain old knowledge. However, any degradation in performance on the initial task or data is known as catastrophic forgetting [4, 5]. This challenge is particularly pronounced in machine learning applications that require models to operate in sequential learning settings, where old and new knowledge must coexist within the same system. This is more so relevant when tasks are correlated, as shared and overlapping features can exacerbate the difficulty in retaining old knowledge while adapting to new information [6].

The primary reason for catastrophic forgetting is due to the shared parameters that are within the architecture of the neural network. Initially, upon training, the model will establish a learned representation for the task on which it was trained. When new data or a new task is introduced to the learning process of the machine learning model, the parameters are updated in order to minimise the loss for the new task, however these updates can lead to unintentional changes to the initial learned representation and thus resulting in a poorer performance and catastrophic forgetting [4, 5]. Whilst machine learning models and neural network models have seen significant technological advancements over recent years, catastrophic forgetting still presents as a significant hurdle, particularly in machine learning tasks which involve continual learning [1, 2].

To address this issue, several methods have been developed to overcome catastrophic forgetting, such as Elastic Weight Consolidation (EWC) [1] and Synaptic

Intelligence (SI) [7]. As both are regularisation-based techniques that constrain updates to important parameters using task-specific importance scores, SI provides a natural and directly comparable counterpart to EWC for evaluating trade-offs in regularisation-based continual learning. EWC works by selectively constraining the updates to the model in order to preserve performance on the initial task. In other words, the parameters deemed most important to the initial task will be allowed to change very little or not at all. Kirkpatrick et al. [8] show that EWC allows a model to preserve old knowledge whilst new knowledge is being acquired. However, these findings are limited in two key aspects. First, the efficacy of EWC in complex real-world datasets has not been extensively evaluated, with most prior research focusing primarily on benchmark tasks and datasets. Second, the three distinct continual learning environments—task-iterated, domain-iterated, and class-iterated learning, as identified by [9]—have not been rigorously tested in a unified framework. This study addresses both omissions by not only evaluating EWC across these three scenarios but also extending the evaluation to real-world datasets, thereby providing a comprehensive analysis of its applicability across diverse and complex setting.

In task-iterated learning the focus is on retaining the performance of a model on initially trained tasks while learning new tasks sequentially, for example, digit classification as the initial task and even-odd classification as the sequential task. Regarding learned parameters, the primary obstacle for EWC is identifying those that are specific and crucial to each task and ensuring that they are protected [10]. Domain iterated learning requires the model to perform the same task but on different variations of input data, for example, topic classification of news articles from 2 different sources of data. In this scenario, the success of EWC is determined by how it is able to generalise across the different domains while preserving the knowledge required to perform the task [11]. Finally, class iterated learning sequentially introduces new classes to an existing classification task, for example progressively adding digits to a digit classification task. Here, the model is required to retain prior information learned regarding the initial classes whilst learning new classes [7]. Evaluating the performance of EWC in the 3 different learning scenarios is a focal point of this report and imperative to produce a comprehensive analysis. Table 1.1 provides a brief summary of the three different continual learning scenarios

[9].

The distinction between these scenarios is closely tied to the concept of task or domain boundaries, which exists on a spectrum rather than as a binary distinction. Task-iterated learning typically has the clearest boundaries due to explicit task identifiers, allowing the model to distinguish between tasks effectively. In domain iterated learning, task boundaries are moderately clear as tasks share objectives but differ in input distributions, such as classifying news topics from different sources. In contrast, class iterated learning presents the least distinct boundaries due to overlapping features and the incremental nature of class additions, such as progressively adding digits in MNIST or topic groups in real-world datasets. However, it is worth noting that the characteristics of the dataset can also influence how clear the task boundaries are, and, as such, the learning scenario must be considered in conjunction with the dataset. Table 1.1 provides a brief summary of these three continual learning scenarios [9].

TABLE 1.1: Summary of Continual Learning Scenarios with Test-Time Requirements [9]

Scenario	**Description**	**Challenges**	**Test-Time Requirement**
Task Iterated Learning	Focuses on retaining performance on initially trained tasks while learning new tasks sequentially.	Identifying and protecting crucial parameters specific to each task.	Task-ID provided
Domain Iterated Learning	Requires the model to perform the same task on different variations of input data.	Generalizing across different domains while preserving necessary task-specific knowledge.	Task-ID not provided
Class Iterated Learning	Involves sequentially introducing new classes to an existing classification task.	Retaining information about initial classes while learning about new ones.	Infer task-ID

This study employs three types of datasets to evaluate continual learning methods

across a spectrum of complexity and realism. Toy datasets facilitate controlled experiments by enabling direct manipulation of task structure. While they lack noise and ambiguity, they pose a meaningful challenge due to their regression-based formulation, allowing for focused analysis of learning dynamics. Benchmark datasets, such as MNIST [12], offer a standardised image classification task commonly used in continual learning research. Despite being more constrained than real-world data, MNIST introduces moderate variability through different handwriting styles, making it more complex than purely synthetic alternatives. Real-world datasets, which in this study involve language-based tasks with domain shifts and noisy labels, represent applied machine learning challenges such as variability, ambiguity, and scalability. These dataset categories were selected to provide complementary perspectives, progressing from controlled environments to realistic conditions. Table 1.2 summarises the purpose and contribution of each dataset type.

EWC has shown some promising results, however there are questions regarding its performance and its application breadth and depth [13]. An assessment of the performance of EWC across the different learning scenarios as defined by de Ven and Tolias [9] has not been done and as such the aim of the research report is to scrutinise the effectiveness of EWC in overcoming catastrophic forgetting across a variety of different benchmark datasets, toy datasets, and real-world datasets in each of the different continual learning scenarios as described above. By exploring EWC's strengths and limitations across a variety of scenarios, this study provides new insights into its applicability and potential for improvement in continual learning.

This report is structured as follows: In **Chapter 2**, a comprehensive review of the common methods for overcoming catastrophic forgetting is presented, with a particular focus on Elastic Weight Consolidation (EWC). The chapter classifies these methods into three main types: regularisation-based methods, architectural approaches, and rehearsal strategies. In **Chapter 3**, the research methodology is described, including the motivation for selecting datasets that align with the three continual learning scenarios. **Chapter 4** presents the results of the experiments conducted across these scenarios, highlighting the performance of EWC and other methods across the different datasets. The findings are critically analysed in **Chapter 5**, where the implications for continual learning methods are discussed, along

TABLE 1.2: Overview of Dataset Types Used for Evaluating EWC

Dataset Type	**Purpose**	**Contribution**
Toy (Synethic) Datasets	Fully controlled and custom-designed to isolate specific learning challenges.	Enables analysis of retention and forgetting dynamics in a regression setting without data ambiguity or noise.
Benchmark Datasets	Widely used in continual learning literature (e.g. MNIST).	Provide a standardised setting for validating reproducibility and comparability with existing methods.
Applied (Real-World) Datasets	Represent higher complexity and practical constraints (e.g. language tasks with noise or variability).	Evaluate robustness and generalisation of different methods under realistic and diverse learning conditions.

with comparisons across datasets and scenarios. Finally, **Chapter 6** summarises the contributions of this research, outlines its limitations, and proposes avenues for future work, including potential hybrid approaches to address the trade-offs observed in retention and adaptability.

Chapter 2

Background and Related Work

Numerous methods have been developed to overcome catastrophic forgetting, with the purpose of preserving as much knowledge as possible across different tasks and different sets of data while still acquiring new knowledge. These can be typically categorised into three broad categories: architectural methods, rehearsal methods, and regularisation methods [1, 7]. Architectural approaches are those that result in a change to the architecture of the neural network when new information is introduced. Here, the goal is to overcome catastrophic forgetting by altering the network architecture when new information is learned in such a way as to ensure that previously learned information is not forgotten [1, 14] or additional capacity is added to the network to accommodate new information more easily. Rehearsal methods require the use of a subset of the data from the previous task, which is used in conjunction with the data for the new task during training [1]. These subsets of prior data act as a refresher for the model on previously acquired knowledge. Architectural [15] and rehearsal methods [9, 10] tend to struggle with regard to scalability and practicality, particularly where there are limitations on the size of the model or the data used is confidential. On the other hand, regularisation methods attempt to overcome forgetting by placing restrictions on the parameter updates while the model is being trained on new data [8, 7].

2.1 Regularisation Methods

Two prominent examples of regularisation methods are EWC and Synaptic Intelligence (SI), both of which focus on penalising updates to parameters critical to earlier tasks. EWC is an approach first introduced by Kirkpatrick et al. [8] that

aims to identify the parameters crucial to earlier tasks and selectively penalise any changes to them during subsequent training, preserving prior knowledge [8]. SI, on the other hand, complements and aims to hone the EWC approach to handle the challenges faced by continual learning [7]. SI attempts to improve the EWC process of adding a penalty term to important parameters by dynamically determining the importance of synapses or nodes during the learning process instead of only once the task is complete. This leads to a more refined continuous cost of forgetting, allowing for more fluid adaptations to the model.

2.1.1 Elastic Weight Consolidation

The aim of EWC is to limit changes model parameters that are deemed important for the first task when training on subsequent tasks. This is done by penalising changes to those parameters which are deemed important to prior tasks. The formulation of EWC is motivated by the Bayesian principles in its use of prior and posterior distributions [8]. The posterior distribution of the parameters for the neural network following training is approximated and then this distribution is used as a prior when the model is being trained on a new task. In order to identify which parameters are deemed important or crucial, their impact on the loss function is measured. Those which have a large impact are deemed more important and thus a larger penalty is applied to them when changes are attempted when the model is being trained on new data. The calculation of EWC requires computing a penalty term which limits the update of each parameter depending on its importance for previously learned knowledge. The importance is quantified by the Fisher Information Matrix (FIM) which measures the sensitivity of the output of a neural network to changes in its parameters. The FIM is computed by evaluating the second-order partial derivatives of the networks's loss function with respect to its parameters, typically averaged over the dataset associated with a given task [8, 7]. It captures how changes to individual parameters affect the network's performance, with higher values indicating that those parameters are critical for maintaining task-specific knowledge.

Mathematically, the FIM is defined as:

$$F_{ij} = \mathbb{E}\left[\frac{\partial^2 \log p(y|x,\theta)}{\partial\theta_i \partial\theta_j}\right] \tag{2.1}$$

In the diagonal approximation used by EWC, this simplifies to:

$$F_i = \mathbb{E}\left[\left(\frac{\partial \log p(y|x,\theta)}{\partial\theta_i}\right)^2\right] \tag{2.2}$$

where $p(y|x,\theta)$ represents the likelihood function of the model parameters given the data.

In practice, EWC uses only the diagonal of the FIM, simplifying its representation and computation [8]. This approach assumes that the parameters are independent, thus reducing computational overhead and memory requirements compared with using the full matrix. The diagonal approximation offers significant advantages in terms of scalability, particularly for large neural networks with numerous parameters, since storing and computing the full FIM would be infeasible. However, this simplification comes with limitations. By ignoring the off-diagonal terms, it disregards potential correlations between parameter updates, which may lead to suboptimal preservation of knowledge, particularly in complex tasks or when the neural network's architecture exhibits high interdependencies. Despite these limitations, the diagonal FIM strikes a practical balance between computational efficiency and the ability to mitigate catastrophic forgetting.

The EWC loss function augments the standard task loss with a quadratic penalty term to preserve important parameters from previous tasks:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\text{task}}(\theta) + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta^*_{A,i})^2 \tag{2.3}$$

Model parameters are then updated using gradient descent on this combined loss:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta\nabla_\theta \mathcal{L}_{\text{EWC}}(\theta) \tag{2.4}$$

Where:

- θ_{new} and θ_{old} are the updated and current parameter values, respectively.
- η is the learning rate.
- $\mathcal{L}_{\text{task}}(\theta)$ is the loss for the current task.
- λ is a hyperparameter that controls the strength of the EWC penalty.
- F_i is the i-th diagonal entry of the Fisher Information Matrix (FIM), approximating the importance of parameter θ_i to the previous task.
- $\theta^*_{A,i}$ is the value of parameter θ_i after training on Task A.

In the above equation, the term $\nabla L(\theta)$ is the gradient of the loss function with regards to the parameters for the new task which encourages the model to learn the new task. The EWC penalty is the second term and this discourages the parameters from changing too much from their ideal value for the prior task weighted by their importance as calculated by the FIM [8].

The inclusion of the penalty term results in EWC making a trade-off between the plasticity to acquire new knowledge or learn new tasks and retaining prior knowledge or maintaining performance on prior tasks [14, 16, 7, 2].

For EWC training, there were two training approaches this report implemented which differed in how they computed and applied the penalty to regularise model updates and prevent forgetting. The first version employed an incremental penalty EWC. Here, the penalty term was incorporated dynamically during training. For each batch, the model calculated a task-specific loss, which was then augmented by the EWC penalty. This penalty was computed based on the FIM, which quantified the importance of model parameters for the current task. The FIM was updated iteratively as training progresses, with gradients from each batch contributing to parameter importance. After training on a task, the model stored the current parameters as optimal values for later penalty calculations. The penalty discourages significant deviations from these stored values during subsequent tasks, effectively retaining knowledge.

The second EWC version was a task-defined importance EWC method, which was better suited for environments with clear task boundaries. In this approach, the

FIM was precomputed after training on each task, using the gradients obtained from the model's outputs, and scaled by an importance hyperparameter to determine the contribution of the penalty to the overall loss. This matrix, alongside the stored parameters from the previous task, was then used to calculate a penalty during training on subsequent tasks. The penalty discourages changes to parameters identified as critical to the prior task. Unlike the above mentioned version, this method computes the FIM once per task, making it better suited for scenarios where task transitions are clearly defined. This explicit task-based structure simplifies the application of EWC but requires distinct separation of tasks during the training process. Both methods aimed to balance learning new tasks while preserving prior knowledge.

2.1.2 Synaptic Intelligence

SI, introduced by Zenke et al. [7], builds upon and refines the principles of EWC by adopting a dynamic approach to parameter importance. Instead of calculating parameter significance only after a task is complete, SI continuously tracks the importance of each parameter throughout the learning process. This is achieved by monitoring changes in the loss function and associating these changes with specific parameters, thus providing more flexibility and improved adaptation. This continuous evaluation of the "cost of forgetting" allows the model to make more fluid adaptations, particularly in scenarios where tasks overlap or evolve incrementally. This makes SI a natural point of comparison when evaluating the strengths and limitations of regularisation-based continual learning methods. Although SI shares conceptual similarities with EWC, its dynamic nature makes it more suitable for environments requiring frequent updates or tasks with shared feature spaces, as noted by Schwarz et al. [17]. However, its reliance on dynamic updates can introduce variability, particularly in tasks with overlapping input features or shared parameter spaces, i.e., where tasks are not well separated, requiring parameters to encode multiple objectives simultaneously [7], and as discussed by Lee et al. [6], who examine node reuse and interference in catastrophic forgetting. In these scenarios, the reassignment of importance values to parameters may conflict with earlier task requirements, resulting in poorer knowledge retention or leading to interference between tasks.

The SI regularisation term can be expressed mathematically as:

$$\mathcal{L}_{\text{SI}} = \sum_i \Omega_i (\theta_i - \theta_i^*)^2 \tag{2.5}$$

Where:

- $\mathcal{L}_{\text{SI}}$ is the regularisation term added to the loss function.
- θ_i represents the current value of the i-th parameter.
- θ_i^* is the optimal value of the i-th parameter from previous tasks.
- Ω_i is the importance measure of the i-th parameter.

The importance measure Ω_i is dynamically computed as:

$$\Omega_i = \frac{\sum_t \Delta\theta_{i,t} \cdot \left(-\frac{\partial \mathcal{L}_t}{\partial \theta_i}\right)}{\sum_t \Delta t + \xi} \tag{2.6}$$

Where:

- $\Delta\theta_i$ is the change in the i-th parameter during training on the current task. These changes are accumulated throughout the entire task until the task switch.
- $-\frac{\partial \mathcal{L}}{\partial \theta_i}$ is the gradient of the loss function with respect to the i-th parameter indicating the contribution of the parameter to the current task's performance.
- Δt is the time interval (or step count) over which the contributions are accumulated during the training of the current task.
- ξ is a small constant to stabilise the computation and avoid division by zero.
- t represents the training step index, summing over all steps within a task.

By continuously updating Ω_i throughout training, SI allows the model to make fluid adjustments, dynamically penalising updates to parameters deemed critical for retaining prior knowledge.

While the iterative approach in the first EWC method might appear to resemble the dynamic updates of SI, the conceptual foundation and implementation is different.

EWC used the FIM to calculate parameter importance which is grounded in probabilities and second-order derivatives of the loss, while SI accumulates contributions to the loss reduction over time, which is unrelated to the probabilistic framework of the FIM.

2.2 Architectural Method

Progressive Neural Networks (PNNs) offer a structural approach to mitigating catastrophic forgetting by dynamically expanding the network architecture for new tasks. Instead of reusing the same parameters, PNNs introduce a new set of neurons at every layer for each new task while freezing the parameters of previously trained tasks. Knowledge transfer between tasks is achieved through lateral connections which link each layer in the newly added task-specific network to the corresponding layers of previously trained networks. These lateral connections enable the sharing of representations across tasks, allowing the new task to benefit from prior knowledge without overwriting it. For example, in vision-based tasks, lower-layer features such as edges can be reused by the new task, while task-specific features are learned in the added layers. The inclusion of task identifiers ensures that task-specific pathways are explicitly engaged, enhancing the retention of prior knowledge and preventing interference across tasks. This architecture ensures knowledge retention while supporting learning for diverse tasks. Rusu et al. [18] visually illustrate the role of lateral connections in enabling knowledge transfer between tasks in Figure 2. This approach is particularly effective in scenarios requiring strong task retention, though the continual growth of the network can pose challenges in scalability and computational resource management [18].

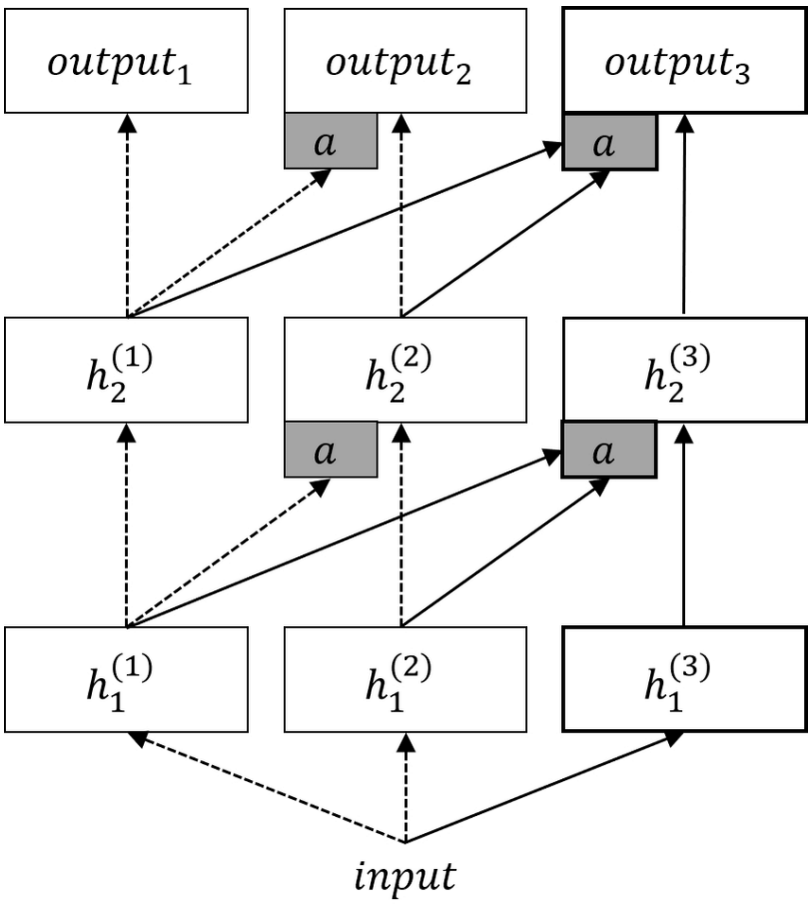


FIGURE 2.1: The figure provides a visual representation of the lateral connections in a three-column PNN, showing how earlier layers transfer knowledge to new task-specific layers while maintaining task isolation. The first two columns correspond to a network trained on tasks 1 and 2 with their parameters frozen. Lateral connections enable knowledge transfer between corresponding layers of previously trained tasks and the new task-specific column. The grey boxes represent adapter layers that allow the new task (Task 3) to access representations from earlier tasks. The dashed lines have frozen weights which correspond to previously learned representations while the solid lines, which correspond to new parameters, are updated during training. Adapted from Rusu et al. [18].

The Progressive Neural Network (PNN) training function used in this report was designed to accommodate the unique architectural structure of PNNs. For each batch of data, the function passed the inputs and the corresponding task identifier through the model, ensuring that the correct task-specific branch of the architecture was used. The model computed the outputs based on this task-specific pathway, and the loss was calculated between the model's predictions and the true targets. The calculated loss was then backpropagated to update the weights of the task-specific layers without interfering with the parameters of previous tasks. This process ensured that knowledge from prior tasks was preserved in separate pathways while enabling learning for the current task. The PNN training function is particularly suited for continual learning scenarios where task boundaries are explicit and where the goal is to avoid overwriting previous knowledge by design.

2.3 Rehearsal Method

Rehearsal methods are a foundational approach in continual learning, designed to address catastrophic forgetting by reintroducing previously encountered data during training on new tasks. These methods rely on storing a subset of prior data, often in a memory buffer, which is replayed alongside new task data to reinforce learned representations and prevent the overwriting of old knowledge [16]. The replay process ensures that the model retains critical information from earlier tasks while adapting to new data. Memory buffers are often carefully curated to ensure class balance and diversity while limiting storage requirements [19].

Recent advancements have explored variations of rehearsal methods that optimise the trade-off between retention and adaptability, such as employing replay ratios instead of relying solely on fixed memory buffers. One notable development is the use of replay ratios which determine the frequency and proportion of prior data included in training, offering greater flexibility in managing computational resources and mitigating memory constraints [20]. This approach can balance the retention of prior knowledge with the efficient incorporation of new information, making it particularly useful in scenarios with limited storage or high-dimensional data. Furthermore, some studies have proposed adaptive replay strategies that dynamically

adjust replay ratios based on task similarity, data difficulty, or model performance, further improving the efficiency of rehearsal methods [21, 22].

The simplicity and adaptability of these methods have made them a cornerstone in continual learning research, often used in conjunction with other techniques to further enhance performance across diverse learning scenarios. These combinations leverage the strengths of each method, with rehearsal ensuring retention of prior knowledge and other approaches addressing task-specific challenges.
The rehearsal training methods used in this paper employed two distinct approaches to integrate prior knowledge from previous tasks. The first is the buffer-based training function, which maintains a fixed-size buffer containing a subset of past task data. During training, batches are augmented with data from this buffer, and the combined data is used to calculate the task-specific loss. This approach ensures retention of previous knowledge while learning new tasks and is often used in toy dataset experiments. The second version is the replay ratio-based training function, where the model alternates between training on new task data and revisiting old task data using a predefined replay ratio. The replay ratio scales the contributions of losses from old and new data, ensuring a balance between retaining past knowledge and learning the current task.

Chapter 3

Research Methodology

The following sections explain the different experiments, detailing aspects such as data generation and setup, neural network architectural design, training and evaluation processes, and analysis. Each method was evaluated across a diverse array of datasets which meet the nuanced definitions of the different continual learning scenarios. The purpose of the toy dataset is to assess the performance of the methods in an uncomplicated setting. The purpose of the benchmark experiment was to evaluate the performance on a widely used dataset. Lastly, the purpose of the real-world dataset was to assess the methods in an environment which introduces the natural and dynamic complications that come with real-world data.

The incremental penalty EWC method was applied in the task iterated benchmark experiment (task A: digit classification; task B: even-odd classification), two of the domain iterated experiments, and all class iterated learning experiments. Specifically, for domain iterated learning, this method was used in the benchmark datasets (e.g., SPLIT MNIST) and real-world datasets (e.g., topic classification across 20 Newsgroups and Reuters), where domains were variations of the same task rather than distinct, independent tasks. In class iterated learning, it was applied to toy datasets (e.g., sinewave variations), benchmark datasets (e.g., incremental MNIST class introduction), and real-world datasets (e.g., incremental topic addition in 20 Newsgroups), where new information was introduced gradually, and previously learned knowledge remained relevant to later stages of training.

These datasets have less distinct task boundaries because they involve progressive

modifications to existing structures rather than entirely new and independent objectives. For example, in class iterated learning, new classes are introduced incrementally rather than as separate tasks, meaning earlier representations remain relevant as new categories are learned. Similarly, in domain iterated learning, shifts between domains (e.g., topic classification across 20 Newsgroups and Reuters) involve variations in data distribution rather than fundamentally different classification tasks. The incremental penalty approach works well in these cases because it continuously updates parameter importance, allowing the model to adjust to gradual changes without rigid task transitions.

In contrast, datasets such as topic classification followed by place identification (task iterated benchmark dataset), or sinewave output followed by phase-shifted sinewave output (task iterated toy dataset), have clearer task boundaries. These involve different underlying objectives that do not share as much representational overlap, making a precomputed Fisher Information Matrix at each task transition more effective. Similarly, in domain iterated learning with sinewave outputs split across x-axis segments, each domain represents a structurally distinct portion of the dataset, requiring adaptation to new input distributions rather than incremental modifications to prior knowledge. In such cases, task-defined importance EWC is more appropriate because the parameter importance can be determined at distinct transition points rather than needing continuous refinement.

A standard evaluation function was used for all domain and task iterated experiments, however, the class iterated experiments required an enhanced function to properly evaluate the dynamics of class iterated learning. Its primary focus of the class iterated function was to measure the model's performance on the subset of classes relevant to the current task while ensuring that it does not rely on task identifiers, which would otherwise classify it as task-iterated learning. To achieve this, a dynamic masking mechanism restricted the model's output logits to the currently active classes, ensuring that evaluations remained task-specific. Additionally, ground-truth labels were remapped to align with the current subset of classes, maintaining consistency between predictions and labels. By selectively masking outputs and remapping targets, the function ensured valid loss and accuracy computations while preserving the integrity of the class iterated learning framework. This targeted approach provided a robust measure of the model's ability to learn

new classes while retaining knowledge of previously encountered ones.

To find the appropriate importance values and learning hyperparameters (epochs and learning rate), a systematic approach was followed to balance retention and adaptation while simultaneously avoiding over-or-under training. An incremental tuning strategy was used for the importance values for SI and EWC. This involved beginning with a low importance value and gradually increasing it while monitoring knowledge retention and adaptability. Rather than identifying a single global optimum, the goal was to locate a practical point on the retention–adaptation trade-off curve (i.e., a Pareto frontier) that offered strong retention without severely compromising adaptability. For learning rates and epochs, initial values were set conservatively to avoid instability. Loss trends and accuracy metrics were closely monitored across epochs, and adjustments were made iteratively to ensure a combination that enabled effective learning without causing overfitting or underfitting. The final learning rate and epoch settings were selected based on this optimisation process.

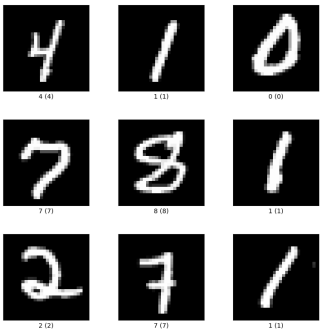


FIGURE 3.1: Example image from the MNIST dataset [23].

The hyperparameter values used in the experiments are summarised in Table B.1 (see Appendix).

3.1 Task Iterated Learning

The task-iterated learning scenario was set up such that a neural network model was trained on sequential tasks in order to assess the ability of EWC and other regularisation methods in overcoming catastrophic forgetting when learning new tasks. These experiments specifically assess how well the model retains performance on previously learned tasks while learning new ones, where the input structure remains consistent across tasks.

3.1.1 Toy Dataset

The following experiment was designed and set up using the sine wave function, where the goal was to predict the output first without a phase shift and then with a phase shift, where the introduction of the phase shift represents a separate task. This experimental setup fits the definition of task-iterated learning since the phase shift introduces a fundamentally new pattern which can be seen as a new task as it requires task-specific adaptations. It therefore assesses the ability of the model to retain task A specific knowledge whilst learning task B. The purpose of the toy dataset is to evaluate EWC in an uncomplicated and straightforward environment.

Figure 3.2 illustrates the sinewave functions used for Task A and Task B. Task A is defined as predicting the output of $y = \sin(x)$, and Task B as predicting the output of $y = \sin(x + \frac{\pi}{2})$. For task identification, an indicator variable is appended to both inputs and outputs: $[1, 0]$ for Task A and $[0, 1]$ for Task B. This indicator enables the model to distinguish between the two tasks during training and evaluation. The MSE loss function was used in this experiment.

The neural network used in this experiment was a basic network with three fully connected layers, designed to handle a simple regression task. The architecture is shown in Figure 3.3.

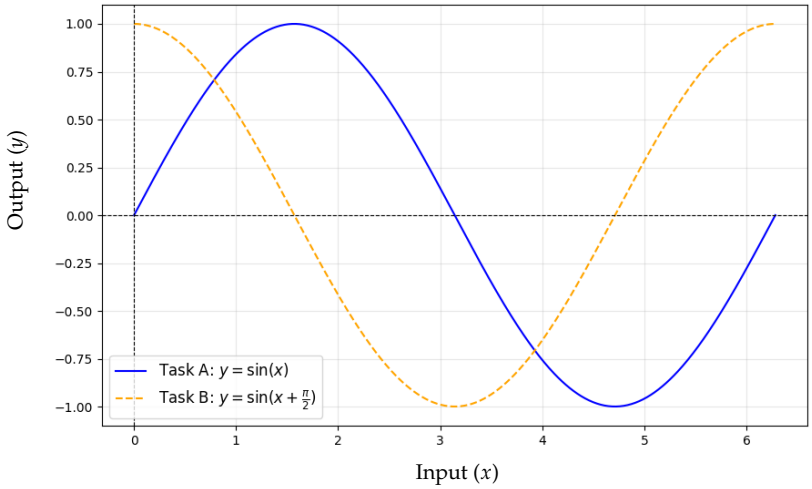


FIGURE 3.2: Task A corresponds to $y = \sin(x)$ (no phase shift), and Task B corresponds to $y = \sin(x + \frac{\pi}{2})$ (half-phase shift). An indicator variable is appended to the inputs and outputs for task identification, with $[1, 0]$ for Task A and $[0, 1]$ for Task B.

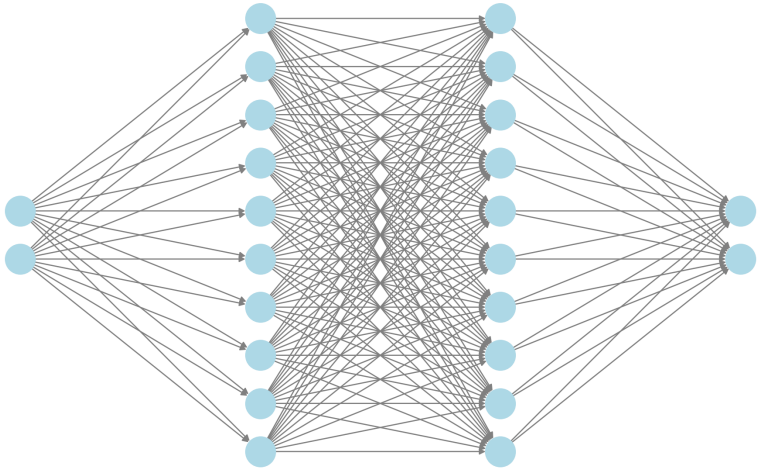


FIGURE 3.3: A simple neural network designed with a single input node plus the task identifier, 2 hidden layers of 10 nodes each, and 2 separate output nodes, one for each task.

3.1.2 Benchmark Dataset

Here, the MNIST dataset [23], a widely used dataset in image classification tasks, was used to create the two different tasks. The first task involved digit classification, whilst the second task was to classify a digit as even or odd. The sequential nature of these tasks requires the model to retain granular multi-class knowledge (digit classification) before adapting to a higher-level binary classification task (even vs. odd). The tasks differ in objectives and outputs but share the same input space, with task identifiers ensuring clear boundaries. This setup, which serves as a middle ground between the simplicity of toy datasets and the complexity of real-world datasets, evaluates EWC's ability to preserve task-specific knowledge and mitigate catastrophic forgetting during transitions between distinct but related tasks. While an alternative setup using more closely matched output structures (e.g., ≥ 5 vs. < 5 followed by even/odd) could offer tighter experimental control, the current design was chosen to introduce a meaningful task shift and assess EWC's capacity to preserve detailed multi-class knowledge during transitions to higher-level abstractions. This experiment made use of the cross entropy loss function.

The neural network was designed for multi-task learning, where tasks are handled using a shared convolutional backbone and task-specific fully connected layers. The model makes use of a task_id as an identifier to determine which task-specific output to activate. The network architecture can be seen in Figure 3.4:

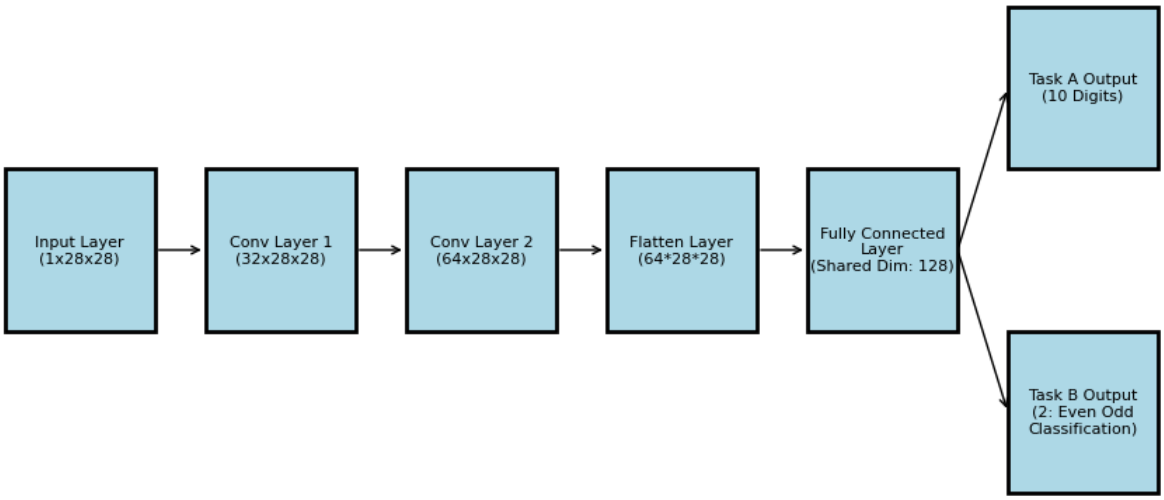


FIGURE 3.4: The model features an input layer processing 28x28 greyscale images, followed by two convolutional layers (32 and 64 filters, respectively), a flattening layer, and a shared fully connected layer with 128 dimensions.

3.1.3 Real-World Dataset

Here a dataset of Reuters news text strings was used where task A required the model to classify the topic, and task B to classify the place or location. The dataset was downloaded from Kaggle [24]. The goal here is to evaluate EWC in a more complicated real-world environment. The two different tasks requires the model to focus on different linguistic features of the same dataset and assesses the ability of EWC to transition between tasks that emphasise different semantic aspects of the input data.

To properly set up the experiment, the train and test datasets underwent several preprocessing steps. The data was cleaned and divided into distinct tasks, filtered to include only entries with a single topic and place label, and then the labels were encoded. The text data was vectorised, and the encoded labels were extracted to serve as the y-values. Task identifiers were appended to both train and test datasets, and data loaders were prepared for each task. Finally, the datasets were split into training, validation, and testing subsets to ensure a comprehensive evaluation framework. Here the cross entropy loss function was used.

The neural network used was designed to handle two separate tasks through the

use of a task identifier to direct data through the task-specific output layer. The model consists of shared hidden layers and separate output layers allowing the model to generate different outputs based on the task at hand. The key components of the network can be seen in Figure 3.5:

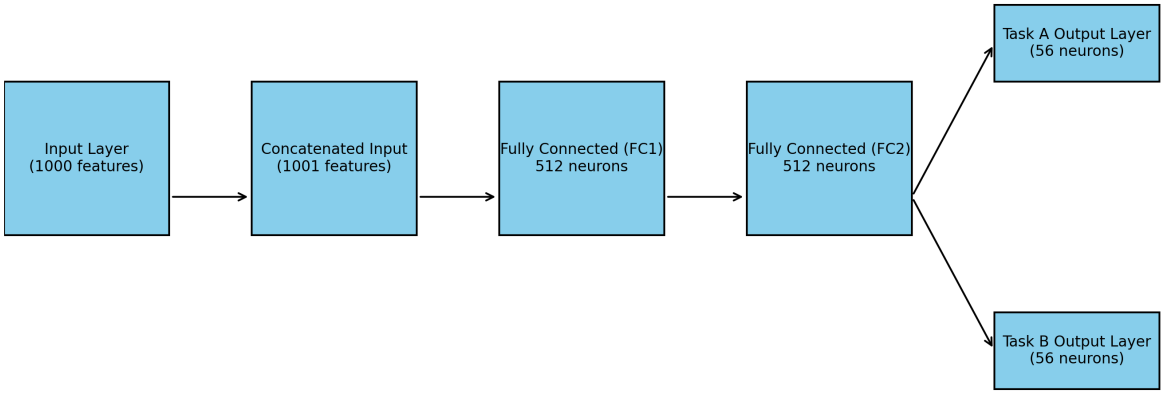


FIGURE 3.5: Input layer consists of 1000 features plus the task identifier. There are two fully connected layers, each with 512 neurons which are shared across both tasks. Separate output nodes for each task, each with 56 neurons.

3.2 Domain Iterated Learning

The domain iterated learning scenario was set up such that a neural network model was trained on a specific task in sequential domains in order to assess the ability of EWC and other regularisation methods in overcoming catastrophic forgetting when different domains are introduced. In other words, assessing the ability of a model to perform a task with the same input-output relationship but presented in a new domain. The purpose is to evaluate how well EWC handles shifts in data distribution whilst ensuring the model maintains generalisation ability across domains. This tests how well EWC can adapt to new domain-specific features. Since domain iterated learning requires that regularisation methods make no use of task identifiers, and that the PNN method makes use of task identifiers within its architecture, the PNN method was not analysed in these experiments.

3.2.1 Toy Dataset

Here, the model was required to predict the output of the sinewave function where the different domains were represented by separate segments of the sinewave function. Domain A consists of segments from $[0\pi, 1\pi]$, $[2\pi, 3\pi]$, and $[4\pi, 5\pi]$, while domain B includes segments from $[1\pi, 2\pi]$, $[3\pi, 4\pi]$, and $[5\pi, 6\pi]$. This experiment setup is appropriate since it involves training a model sequentially on different domains of the same underlying task - in this case, learning from distinct segments of the sinewave function. Here the model is exposed to multiple domains sequentially, where each domain contains variations in the input data while keeping the same task objective. The aim is to assess the ability of the model to adapt to new domains and in its ability to generalise across domains. This experiment made use of the MSE loss function.

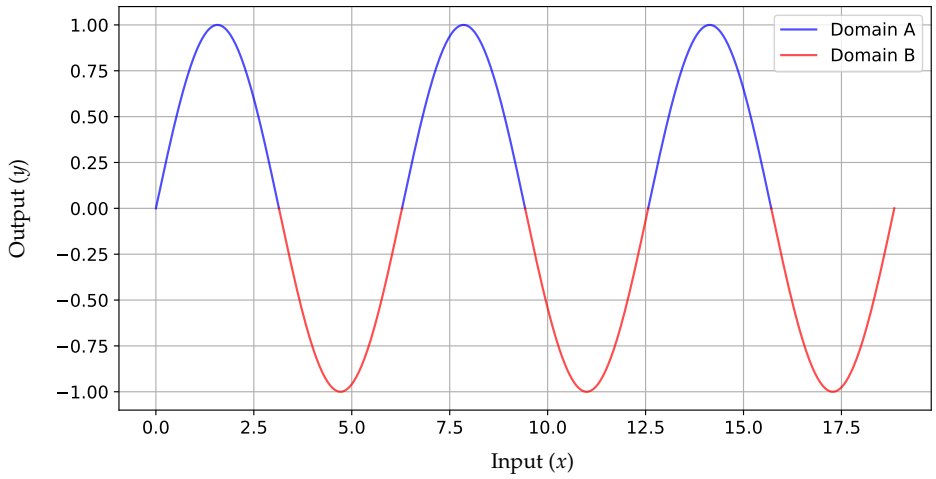


FIGURE 3.6: Sinewave functions used for domain A and domain B. This chart illustrates the segmentation of a sine wave into two distinct domains: Domain A (blue) and Domain B (red). Each domain consists of alternating segments along the sine wave, capturing different portions of the function.

The neural network used was a fully connected feedforward network designed to learn and predict the output of the sinewave function. Figure 3.7 shows the network architecture used for this experiment.

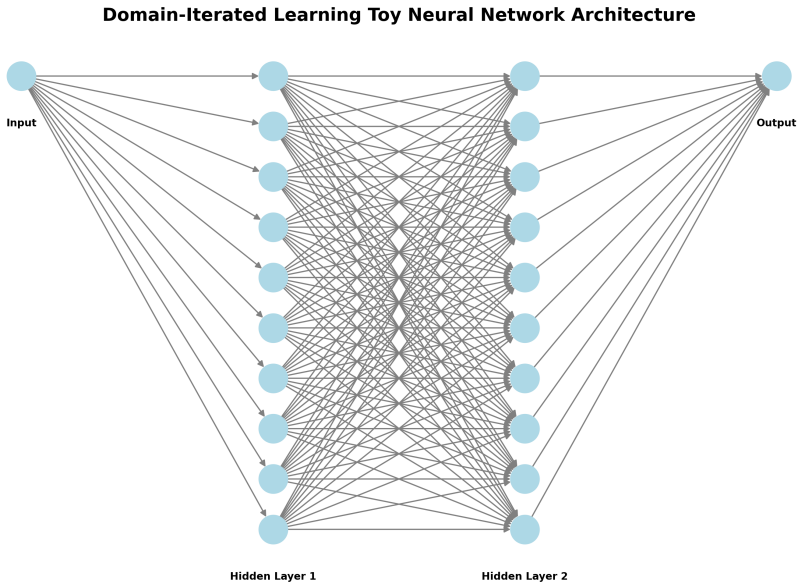


FIGURE 3.7: A simple neural network designed with one input and output with 2 hidden layers of 10 nodes each.

3.2.2 Benchmark Dataset

This experiment makes use of the Split MNIST dataset [9] where the task is binary classification to determine whether a digit is even (label 0) or odd (label 1). Domain A is represented by the even digits and domain B as the odd digits. These domains are treated as separate datasets to investigate how well a model trained on one domain generalises or retains performance when exposed to another domain. While this differs slightly from the setup described by de Ven and Tolias [9], where tasks involve pairs of digits (e.g. [0,1], [2,3], etc.) with class labels alternating between tasks, our setup was intentionally designed to isolate domain shift based on digit type (even vs odd) rather than digit identity. This provides a clearer evaluation of a model's ability to retain and generalise under domain-iterated scenarios with shared output semantics. This experiment made use of the BCE logits loss function.

The neural network architecture used for this experiment was designed as found in Figure 3.8:

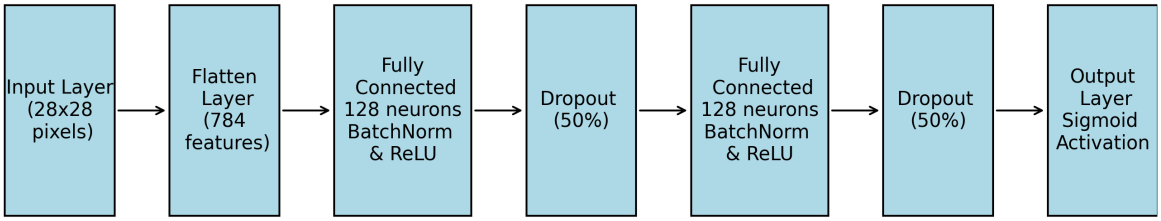


FIGURE 3.8: A neural network designed for binary classification.

Dropout layers were incorporated into the above network architecture to enhance the model's generalisation capabilities and mitigate the risk of overfitting. By randomly deactivating a fraction of neurons during training, dropout prevents the network from becoming overly reliant on specific neurons, encouraging more robust and distributed feature learning. This technique is particularly beneficial for deep neural networks, where the risk of overfitting increases due to the large number of trainable parameters. The chosen dropout rate of 0.5 ensures a balance between retaining sufficient information flow during training and introducing enough regularization to improve performance on unseen data.

3.2.3 Real-World Dataset

This experiment made use of two different datasets, being the 20 Newsgroup dataset which contains 20 distinct topics such as computing hardware, sport, politics amonst others, [25], and the Reuters-21578 data which originates from the Reuters Corpus and sourced via the Natural Language Toolkit [26]. The task here was topic classification and the two different domains were represented by the different news dataset sources which differ in terms of domain-specific language and context. The experiment assesses the ability of EWC to adapt to domain-specific nuances whilst retaining general topic classification capability, and the ability of EWC to manage knowledge retention and domain-specific adaptations simultaneously.

The network architecture used for this experiment can be seen in Figure 3.9:

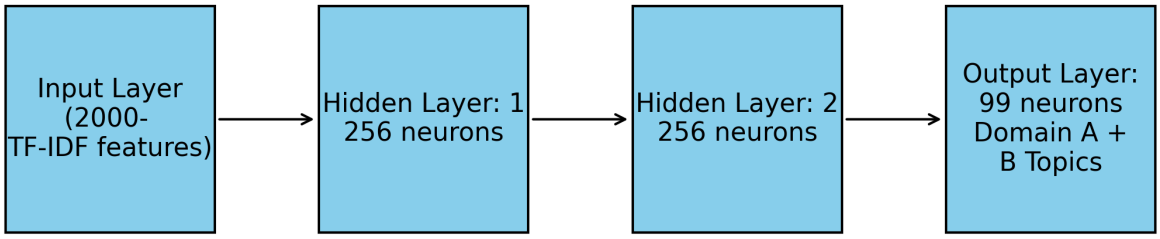


FIGURE 3.9: A network architecture designed for text classification. Input of 2000 representing TF-IDF features, 2 hidden layers of 256 neurons, and an output layer of 99 neurons representing the sum of the different topics across domain A and domain B.

3.3 Class Iterated Learning

Here, a model was trained on sequential classes to evaluate how well different continual learning algorithms retain knowledge of original classes while learning additional ones where the focus is on incrementally introducing subsets of classes from the same dataset. The purpose is to evaluate EWC in its ability to learn new class labels whilst preserving knowledge of previously learned classes. The goal is to evaluate EWC in its ability to overcome catastrophic forgetting in classification problems whilst maintaining accurate predictions across all classes. Similar to the domain iterated experiments, class iterated learning does not make use of any task identifiers, and as a result, the PNN method was not analysed in any of the class iterated experiments.

3.3.1 Toy Dataset

This experiment used the sinewave function to create different classes where the task was the same as those from the other toy dataset experiments - to predict the output of the function making it another regression task. Here, class A was the standard sinewave function, class B was the sinewave function with a higher frequency, and class C introduced a higher amplitude. This experiment is well-suited for class-iterated learning as it progressively introduces new variations of the function. Each sinewave function represents a unique class defined by its specific input-output behaviour, akin to distinct categories in classification but within a regression framework. The differences in frequency, amplitude, and overall complexity require the model to generalise across diverse patterns while retaining knowledge

of previously learned functions. The neural network architecture used for this experiment was the same as that used for the domain-iterated toy dataset experiment, see Figure 3.7. This experiment used the MSE loss function.

Figure 3.10 illustrates the sinewave functions used to represent class A, B, and C. Class A is the sinewave standard function, class B is a high frequency sinewave function, and class C is a modulated sinewave with a dynamically changing amplitude.

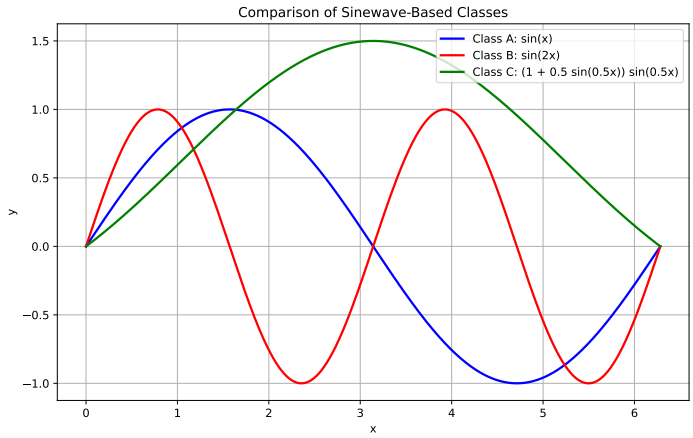


FIGURE 3.10: Sinewave functions used for Task A and Task B. Task A corresponds to $y = \sin(x)$ (no phase shift), and Task B corresponds to $y = \sin(x + \frac{\pi}{2})$ (half-phase shift). An indicator variable is appended to the inputs and outputs for task identification, with $[1, 0]$ for Task A and $[0, 1]$ for Task B.

3.3.2 Benchmark Datasets

This experiment made use of the MNIST dataset where the digits were subdivided into smaller groups or classes. The digits 0 to 3 were defined as class A, 4 to 6 as class B, and the remaining digits as class C. Since this was a class iterated experiment the model was trained on these classes sequentially and the performance of the regularisation methods were analysed based on how well the model was able to classify digits from the original classes after learning new ones. This setup highlighted how different algorithms handle knowledge retention and adaptability

when transitioning between tasks with overlapping input spaces. This experiment made use of the cross entropy loss function.

The network architecture used for this experiment can be seen in Figure 3.11:

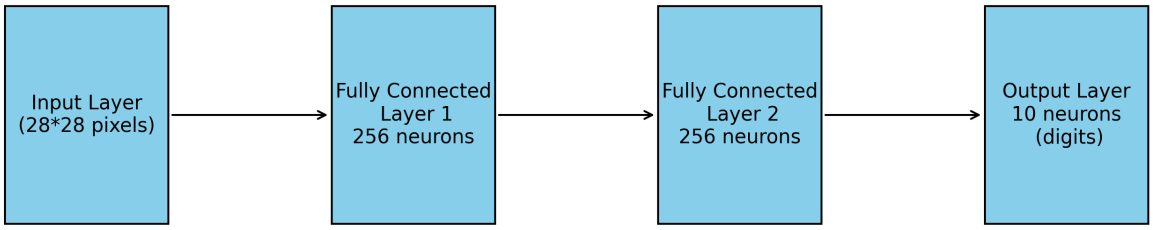


FIGURE 3.11: A Neural Network designed for image classification. Input of 28x28 pixels, 2 hidden layers of 256 neurons, and output layer of 10 neurons representing the 10 digits.

3.3.3 Real-World Datasets

This experiment made use of the 20 News group dataset [25], as described above, where the dataset was divided into four separate groups each containing five distinct labels. This dataset consists of textual data with variability on vocabulary, writing styles, and topic overlap, thus making it a much more challenging and realistic scenario compared with the MNIST dataset. Each group introduces a new set of classes, simulating real-world scenarios where data arrives incrementally. Text data is inherently complex with high dimensionality, which tests the ability of a model to manage catastrophic forgetting and adapt to new classes in a resource intensive domain. The purpose is to evaluate the robustness and applicability of continual learning techniques to a real-world setting involving unstructured data and where data arrives incrementally. In addition, to investigate the effect of increasing class complexity and potential inter-class similarity on continual learning performance. The model was required to correctly classify the news label of the input and the goal of the experiment was to evaluate how well EWC is able to retain knowledge of the initial labels whilst learning additional ones. This experiment made use of the cross entropy loss function.

Since this experiment involved the use of four groups of data, it required four different training sets, four testing, and four validation sets. This were created in the

same way as the other experiments. The breakdown of the topics of each class were as follows:

- Class A: alt:atheism, comp:graphics, comp:os-ms-windows-misc, comp:sys-ibm-pc-hardware, comp:sys-mac-hardware.
- Class B: comp:windows-x, misc:forsale, rec:autos, rec:motorcycles, rec:sport-baseball.
- Class C: rec:sport-hockey, sci:crypt, sci:electronics, sci:med, sci:space.
- Class D: Soc:religion-christian, talk:politics-guns, talk:politics-mideast, talk:politics-misc, talk:religion-misc.

The model architecture was similar in nature to that described in the domain iterated real-world experiment, see Figure 3.9, except since only the 20 Newsgroup dataset was used, the output of the network was only 20.

Table 3.1 provides a summary of the different datasets used for each experiment within each learning scenario.

The datasets were first divided into three subsets: The split of 64% for training, 16% for validation, and 20% for testing was chosen to ensure a balance between adequate training data and robust evaluation and is consistent with standard practices in machine learning. Then each training algorithm was run ten times to ensure reliable and reproducible results. For classification tasks, performance metrics such as accuracy, precision, and recall were recorded, while regression tasks were evaluated using mean absolute error (MAE) and mean squared error (MSE). Additionally, the mean, maximum, and minimum performance results were captured to provide a comprehensive view of the model's capabilities. The following steps were taken for training and evaluation: before any training commenced, the model's performance on task A was evaluated to establish a baseline for untrained performance; The model was then trained on task A for a specified number of epochs, during which its performance on both task A and task B was measured at each epoch, providing insights into learning progress and task generalisation; after completing training on task A, the model's performance on both task A and task B was again measured to capture any residual knowledge transfer or forgetting; subsequently, the model

TABLE 3.1: Summary of Datasets for Continual Learning Scenarios

Scenario Type	**Toy Datasets**	**Benchmark Datasets**	**Real-World Datasets**
Task Iterated	Output of y in $y = \sin(x)$, and Task 2: Output of y with phase shift $(y = \sin(x + \alpha))$	MNIST, task A = classify digit, task B = classify digit as even or odd	Reuters News collection, task A = topic classification, task B = place label
Domain Iterated	Output of y in $y = \sin(x)$ with different segments of the function representing different domains	Split MNIST with even digits as Domain A, and odd as Domain B	Topic classification of different news datasets, 20NewsGroup as Domain A, Reuters News as Domain B
Class Iterated	Class A: standard sinewave output, Class B: sinewave with higher frequency, Class C: sinewave with higher amplitude	MNIST, introduce classes incrementally with increasing complexity	Sklearn's 20 NewsGroups dataset, start with small subset and introduce additional subsets containing different topics

was trained on task B, with evaluations performed on both task A and task B at each epoch to monitor changes in performance during sequential task learning. Task B training was conducted separately for each continual learning algorithm—EWC, SI, PNN (for task-iterated only), and Rehearsal—as well as for a baseline experiment without any algorithm applied where each algorithm represented a distinct setup. After training on task B, the model was once more evaluated on both tasks to assess its ability to retain knowledge of task A while learning task B. This entire process was repeated across ten runs to ensure robustness and consistency in the findings, and to establish variance between individual runs. Following this, the results were then used to produce 2 different graphs. The first being the performance of task, domain, or class A during the initial task, domain, or class and then during the subsequent task, domain, or class(es). This procedure was then repeated but instead tracked the performance of the subsequent task, domain, or class(es).

Chapter 4

Results

All performance metrics presented in this chapter are evaluated on the test set, unless explicitly noted otherwise. Two standard deviations were chosen to represent the shaded regions around the mean performance in the results figures as a standard practice to capture approximately 95% of the variability in the data, assuming a roughly normal distribution.

4.1 Task Iterated Learning

4.1.1 Toy Dataset

The first chart in Figure 4.1 shows the mean absolute error performance of task A during training on task A and during training on task B. The second chart shows the error of task B during training on task A and during training on task B. Due to the unique architectural setup of the PNN method, it requires the initialisation and setup of its own class and network, hence the performance of Task A during training on Task A can differ to those of EWC, baseline, SI, and rehearsal.

Figure 4.1 highlights the impact that these continual learning methods have in overcoming catastrophic forgetting through the fact that task A performance completely deteriorated once training on task B begun when no method was used.

The toy dataset, with its simple structure and distinct task boundaries, provided an environment where retention-focused methods performed well. PNN excelled at retaining task-specific knowledge due to its isolated task architecture, with negligible forgetting observed. However, a small amount of variability in performance

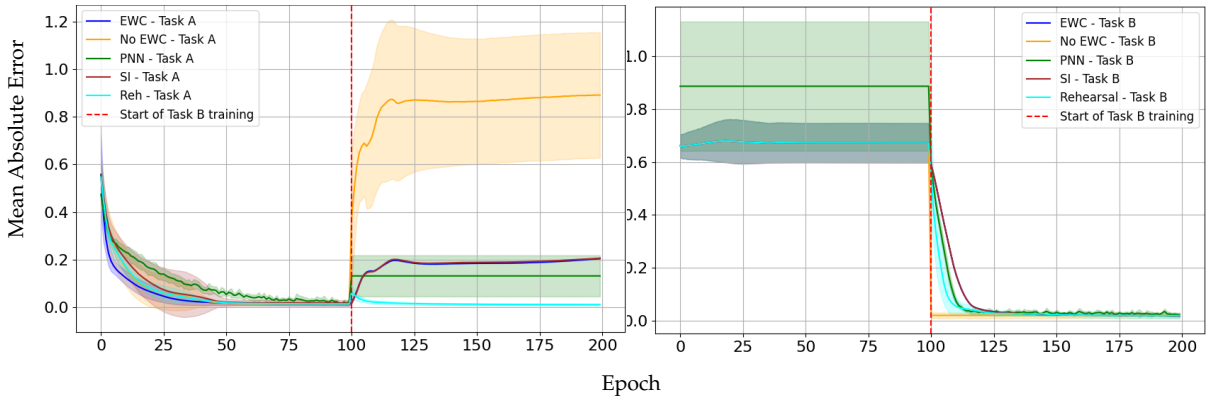


FIGURE 4.1: Task A MAE during task A training (left) and task B MAE during task B training (right) for the different algorithms over 200 epochs. The results highlight the impact that these continual learning methods have in overcoming catastrophic forgetting through the fact that task A performance completely deteriorated once training on task B begun when no method was used. Interestingly, EWC and SI show nearly identical performance and minimal variance in this experiment. This is likely due to the controlled nature of the toy dataset and the simplicity of the underlying regression task, which results in both regularisation methods converging to similar solutions. Additionally, the regression setting may reduce the sensitivity of importance-based regularisation, especially when the network converges rapidly and task interference is limited. The lack of observable variance may also be attributed to consistent initialization and deterministic evaluation across runs. The shaded areas for each algorithm represent 2 standard deviations above and below the mean performance.

retention was noted across runs, likely attributable to evaluation-related factors such as initialisation of lateral connections or randomness in weight initialisation, rather than interference between tasks—which PNNs are explicitly designed to prevent. These findings underscore the importance of averaging results across multiple runs to distinguish real performance trends from random fluctuations. The explicit memory mechanism found in Rehearsal effectively preserved earlier knowledge while enabling strong adaptation to new tasks. EWC also performed well in retaining prior knowledge, leveraging the FIM to preserve critical parameters without negatively impacting the adaptation to task B. SI exhibited similar retention and plasticity capabilities to EWC. The Baseline Method, lacking explicit retention mechanisms, adapted well to new tasks but suffered from severe catastrophic forgetting, making it unsuitable for sequential learning. A summary of knowledge retention versus plasticity performance for task A can be seen in Table C.1 while the performance of task B is summarised in Table C.2.

SI, EWC, and Rehearsal all exhibited little to no variance, while PNN prdouced moderate variability in knowledge retention of task A.

4.1.2 Benchmark Dataset

Figure 4.2 first shows the model accuracy of task A during training on task A and during training on task B and then the accuracy of task B during training on Task A and during training on task B. Figure 4.3 shows the individual runs of the Rehearsal, EWC, and SI algorithms. Model weights after training on Task A were reused for EWC and the baseline methods.

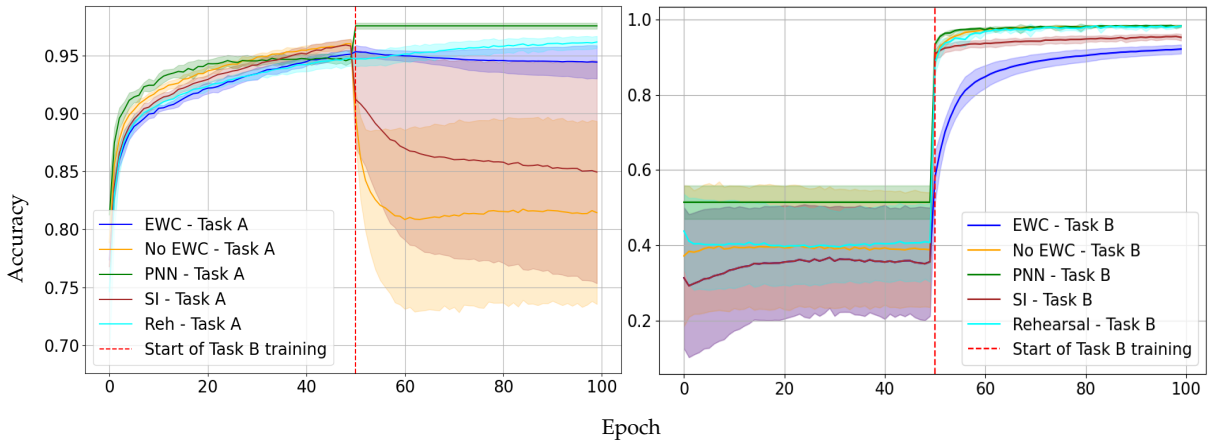


FIGURE 4.2: Task A accuracy during task A training (left) and task B accuracy during task B training (right) for the different algorithms over 100 epochs. The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The results from this experiment highlight the impact that parameter interference can have on the performance of SI, especially when tasks are not well separated [7].

The benchmark dataset introduced overlapping feature spaces (e.g., shared low-level features like strokes or shapes in the digits), increasing task interference and making the retention-adaptation trade-off more pronounced. Rehearsal remained the most balanced method, with strong retention and adaptability. However, its performance exhibited slightly higher variability than in the toy dataset, reflecting the challenge of managing overlapping feature representations. PNN demonstrated excellent retention with minimal task interference, benefiting from its task-specific pathways, with adaptability on par with Rehearsal. EWC performed well in knowledge retention, with slight variability, although its adaptability was slower

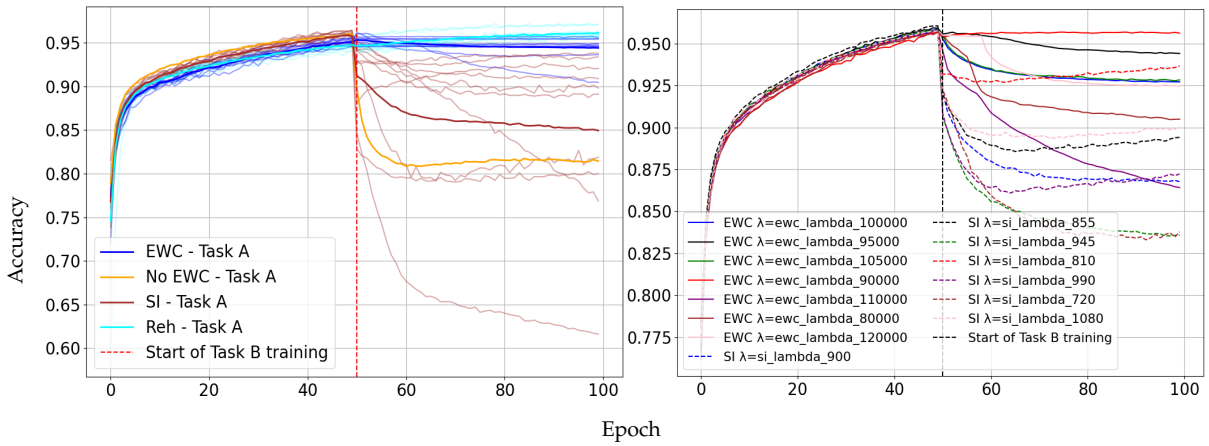


FIGURE 4.3: Comparison of the mean and individual learning paths of 10 runs for Rehearsal, EWC, and SI (left) and sensitivity analysis of lambda or importance values for EWC and SI. The spread of the individual runs highlights the variability of each method, where SI displays significantly greater fluctuation compared with EWC. Lambda values used in the sensitivity analysis were the baseline value (100000 for EWC and 900 for SI) then 95%, 105%, 90%, 110%, 80%, and 120% of base lambda.

and marginally lower. SI exhibited greater variability in this dataset and significantly poorer retention, reinforcing prior observations of its sensitivity to overlapping input spaces, as noted by Zenke et al. [7]. However, SI realised a more rapid convergence during early training on Task B, potentially refelcting a different retention–plasticity trade-off, where SI's dynamic, online importance estimation allows for faster initial learning at the potential cost of long-term stability. The dynamic nature of SI and its frequent reallocation of parameter importance can lead to inconsistencies, negatively impacting results in certain settings. This variability is evident in Figure 4.3, which not only illustrates the broader spread of learning trajectories for SI across runs, but also highlights its heightened sensitivity to the choice of importance values. In contrast to EWC—which showed relatively stable performance across a range of λ values—SI exhibited a noticeably wider performance range, underscoring the method's susceptibility to hyperparameter tuning and the challenges of achieving consistent retention in more complex scenarios. However, while SI produced a larger spread of individual runs, the variance band from Figure 4.2 of SI was strongly influenced by a single poor performing experimental run. The Baseline Method adapted well to the new task, but consistently failed to retain earlier knowledge, demonstrating the same catastrophic forgetting observed in the toy dataset. A summary of knowledge retention versus plasticity performance for

task A can be seen in Table C.3 while the performance of task B is summarised in Table C.4.

4.1.3 Real-World Dataset

The below charts show the model accuracy of task A during training on task A and during training on task B. The second chart shows the accuracy of task B during training on Task A and during training on task B. In this experiment the network weights after training on Task A were saved after each run and reused for EWC, no regularisation, SI, and the rehearsal methods. As a result, performance of task A during training on task A were the same for all these methods.

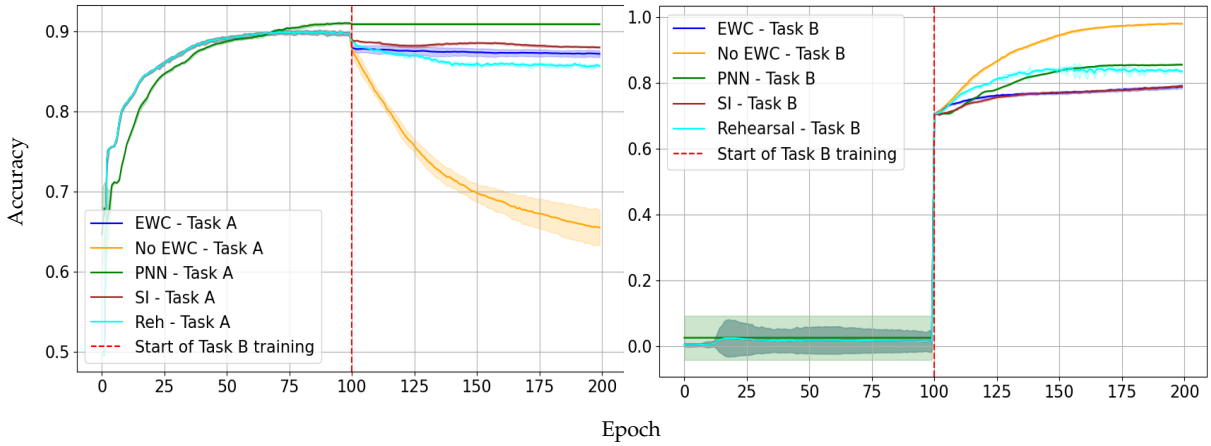


FIGURE 4.4: Task A accuracy during task A training (left) and task B accuracy during task B training (right) for the different algorithms over 200 epochs. The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. This experiment showcased how stable results can be when tasks are well-separated with little to no parameter interference which allows for an effective preservation of old knowledge. Additionally, the results highlight the inherent trade-off in knowledge retention and plasticity.

The real-world dataset represents the most realistic and complex challenge, due to its natural language modality, higher input variability, and the semantic overlap between the tasks of topic and location classification. While the observed forgetting was not uniformly worse across all methods, the linguistic complexity and overlapping task objectives made this scenario particularly demanding for retention-focused continual learning methods. Rehearsal maintained competitive performance by leveraging a replay ratio to reinforce earlier task representations while

adapting to new ones. This method was particularly effective in handling overlapping features while preserving task-specific distinctions. PNN continued to exhibit strong task retention, though its slower convergence (compared with the baseline method) emphasised a trade-off between retention and adaptability. However, PNN realised the best task B adaptation compared with the other algorithms. EWC, which excelled in simpler datasets, struggled more in this complex setting in terms of adaptability, however, task A retention was strong. SI retained task A knowledge marginally better than EWC, possibly benefitting from dynamic importance updates, and its adaptation to task B remained on par with EWC. The Baseline Method, as expected, adapted well to the new task but completely forgot earlier tasks, further reinforcing its unsuitability for real-world sequential learning. Overall, this experiment showcased how stable results can be when tasks are well-separated with little to no parameter interference which allows for an effective preservation of old knowledge. Additionally, the results highlight the inherent trade-off in knowledge retention and plasticity. A summary of knowledge retention versus plasticity performance for task A can be seen in Table C.5 while the performance of task B is summarised in Table C.6.

4.2 Domain Iterated Learning

4.2.1 Toy Dataset

Figure 4.5 shows the mean absolute error performance first of domain A during training on domain A and during training on domain B and the error of task B during training on domain A and during training on domain B. While Figure 4.6 shows the individual runs of SI and EWC. Similar to the task-iterated real-world experiment, the network parameter weights were saved after each run of domain A training and reused during each run of domain B training for rehearsal, EWC, and baseline methods. In other words, the network parameters were the same at the start of domain B training for each of these methods.

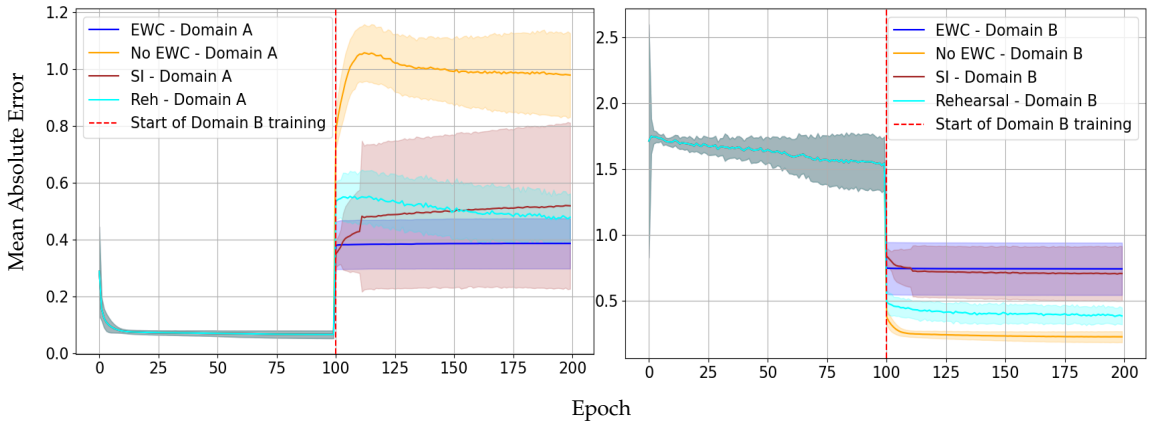


FIGURE 4.5: Domain A MAE during domain A training (left) and domain B MAE during domain B training (right) for the different algorithms over 200 epochs. The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The results from this experiment highlight the trade-off in knowledge retention and plasticity.

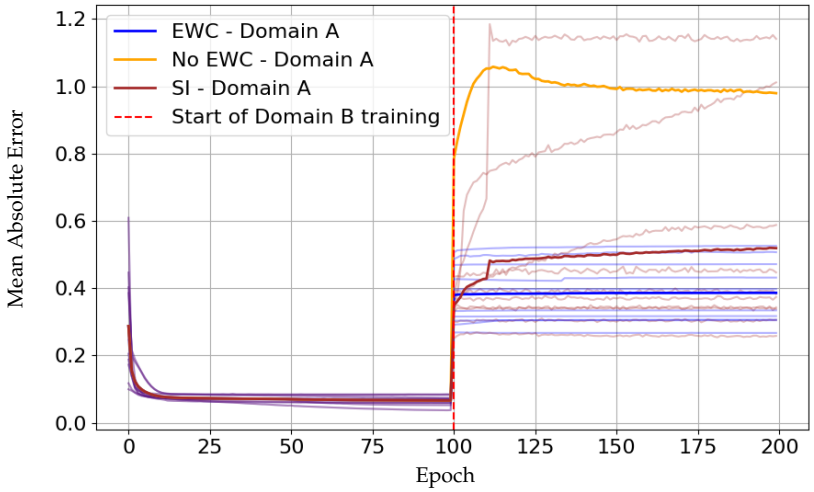


FIGURE 4.6: This plot compares the mean and individual learning paths of 10 runs for EWC and SI. This spread of the individual runs highlights the variability of each method, where SI displays significantly greater fluctuation compared with EWC.

These results highlight a difference in performance between EWC and SI. The higher variability and weaker domain A performance retention of SI can be attributed to the dynamic nature of how importance scores are updated and how contributions to loss are accumulated over time. These factors can lead to instability and increased variability, particularly when the algorithm encounters shifts in the input distribution. In contrast, EWC maintains a stricter regularisation approach, which ensures

that parameter updates for Domain B do not significantly alter those that were critical for Domain A. This inherently limits the variability in retention because the constraints are explicitly designed to prevent large shifts in important weights. EWC was able to more effectively preserve domain A performance with less variability compared to SI, while also achieving comparable plasticity in adapting to domain B. Rehearsal was able to adapt to domain B the most effectively thanks to its replay mechanism and struck a middle ground in domain A retention between SI and EWC with moderate variability. A summary of knowledge retention versus plasticity performance for domain A can be seen in Table C.7 while the performance of domain B is summarised in Table C.8.

These findings suggest that while regularisation-based methods can slow forgetting, they are inherently limited in how well they allow for adaptation, whereas rehearsal-based methods provide a more direct mechanism for maintaining past knowledge without overly constraining future learning. Figure 4.6 compares the variability of runs between EWC and SI, with SI showing a much higher degree of variability in the results with a wide range of performance. On the other hand, the runs of EWC are concentrated in a moderate band around the mean.

4.2.2 Benchmark Dataset

The below charts show the model accuracy of domain A during training on domain A and during training on domain B. The second chart shows the accuracy of domain B during training on domain A and during training on domain B. Network weights were reused for all methods, except Rehearsal, after domain A training and hence accuracy of domain A after training on domain A are the same for all methods. The Split MNIST dataset was used to create the separate domains where domain A is represented by the even digits and domain B as the odd digits.
The benchmark experiment examined how methods handled domain shifts when transitioning from even to odd digits. All three algorithms showed moderate performance in balancing retention and adaptability, influenced by the overlapping input spaces. This overlap likely constrained the ability of EWC and SI to maintain stability while adapting to new patterns as well as introduced complexity in the memory buffer for Rehearsal. Among the methods, Rehearsal exhibited the greatest variability, as its reliance on stored examples made it sensitive to the distribution

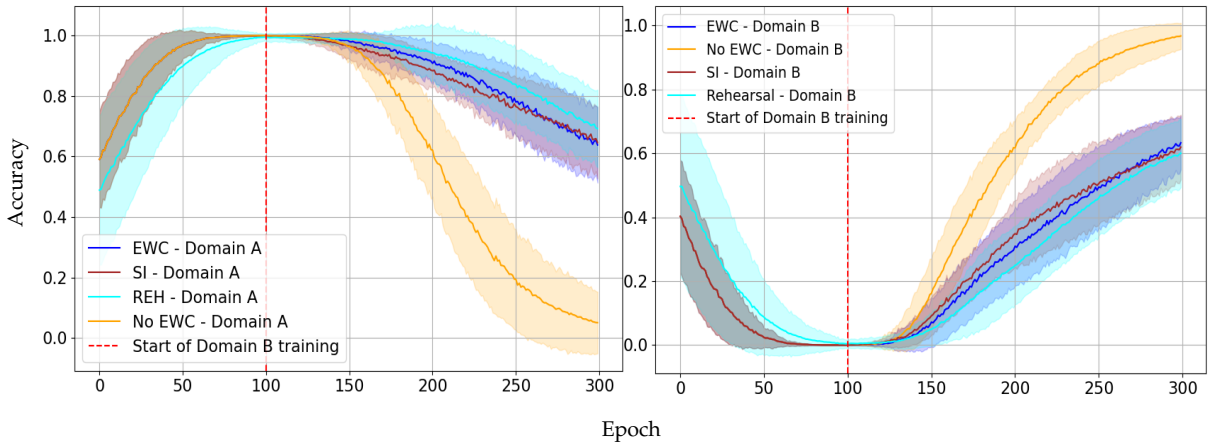


FIGURE 4.7: Domain A accuracy during domain A training (left) and domain B accuracy during domain B training (right) for the different algorithms. The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. This experiment challenged the regularisation methods, which realised moderate retention, reduced adaptability, and variable results. This is likely due to the overlapping input spaces inherent in the MNIST dataset. The overlapping feature spaces most likely strained the memory buffer used in rehearsal training leading to increased variability.

of past and new data. EWC produced slightly more stable results than SI, likely due to its structured constraint on parameter updates, whereas SI's dynamic importance tracking introduced greater performance fluctuation. A summary of knowledge retention versus plasticity performance for domain A can be seen in Table C.9 while the performance of domain B is summarised in Table C.10.

4.2.3 Real-World Dataset

The below charts show the model accuracy of domain A during training on domain A and during training on domain B. The second chart shows the accuracy of domain B during training on domain A and during training on domain B.

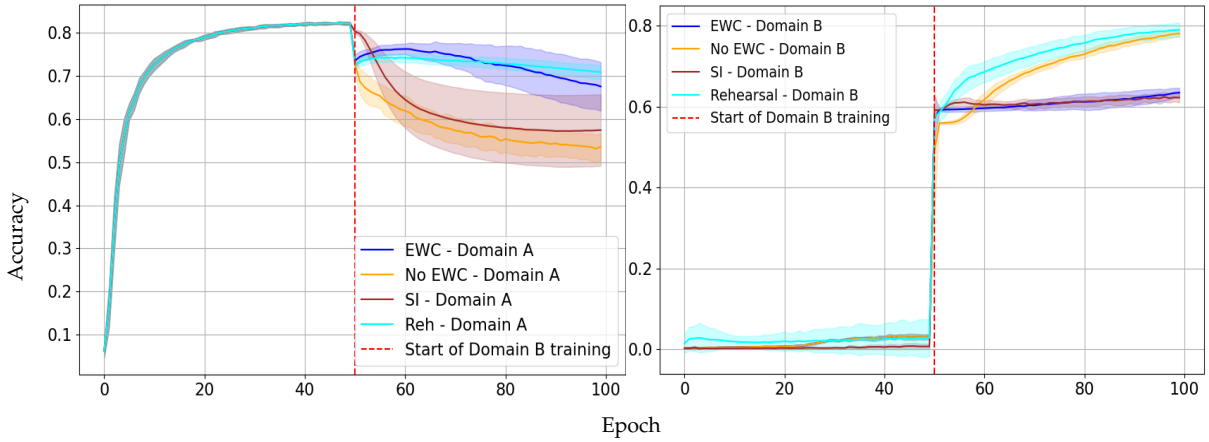


FIGURE 4.8: Domain A accuracy during domain A training (left) and domain B accuracy during domain B training for the different algorithms (right). The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The results showcase the susceptibility of SI in learning scenarios with large domain shifts [7], which can dilute parameter importance. The slower learning curve for EWC methods highlights the limitations of EWC in handling the complex nature of real-world datasets as noted by [27] and [28].

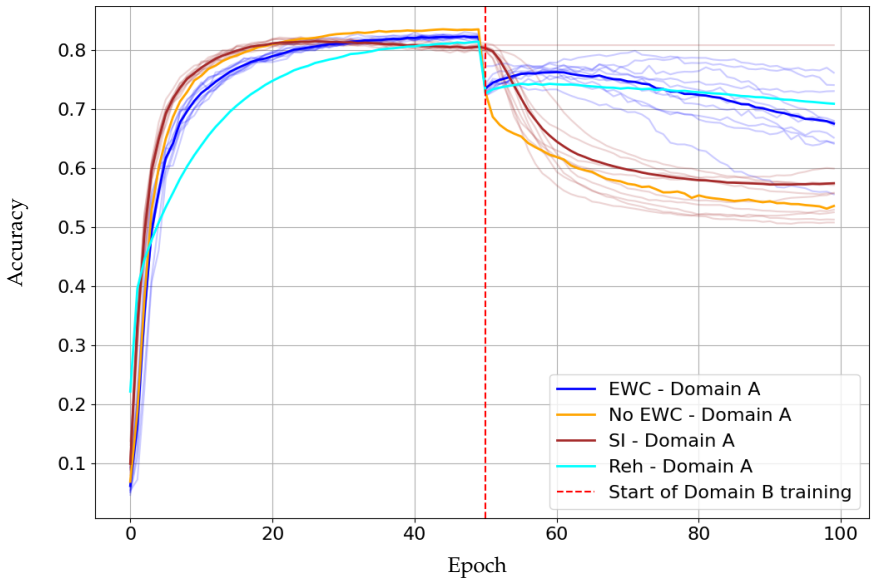


FIGURE 4.9: This plot shows the individual runs of EWC and SI and highlights the variability of both methods. In particular, it showcases SI's weakness in domains with large or sudden shifts.

SI struggled with domain A retention and domain B adaptation. This can be attributed to SI's weakness in scenarios with sudden domain shifts since it accumulates parameter importance gradually and does not directly prevent drastic weight

shifts, meaning catastrophic forgetting can occur more easily. EWC, by contrast, enforces a more rigid constraint which may provide a more stable foundation for retention while allowing better adaptation. In addition, this can help EWC constrain specific weights from shifting too far, effectively anchoring key aspects of domain A, while learning domain B, thus assisting in overcoming forgetting. EWC realised similar retention to Rehearsal, but with greater variability, while its slower learning curve compared with Rehearsal highlights the limitations of EWC in handling the complex nature of real-world datasets, as noted by Hsu et al. [27] and Nguyen et al. [28]. Figure 4.9 showcases how SI can perform worse than no algorithm in overcoming forgetting and how it can latch on to domain A knowledge with no forgetting, highlighting its unpredictability and instability. A summary of knowledge retention versus plasticity performance for domain A can be seen in Table C.11 while the performance of domain B is summarised in Table C.12.

4.3 Class Iterated Learning

4.3.1 Toy Dataset

The first chart shows the MAE of class A during training on class A, B, and C. The second chart shows the MAE of class B during training on class A, B, and C. The final chart shows class C MAE during training on class A, B, and C.

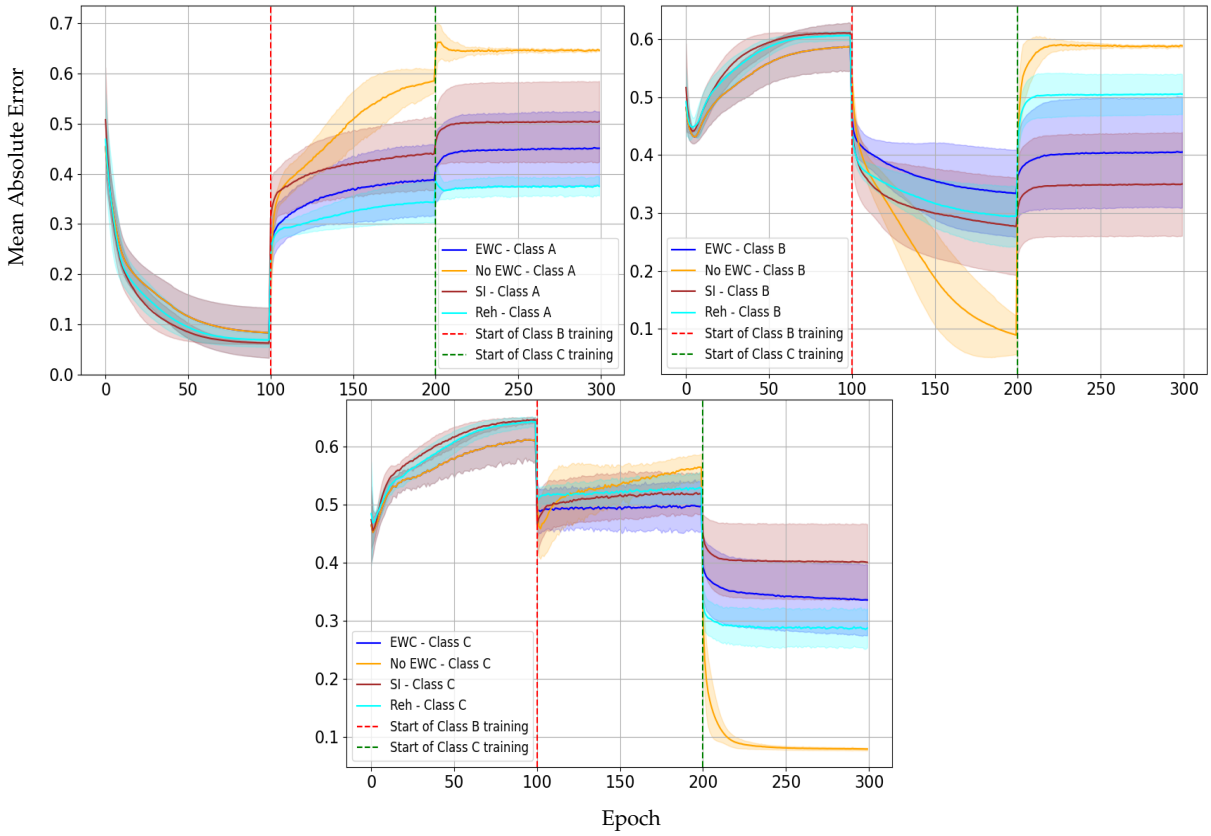


FIGURE 4.10: Class A accuracy during class A, B, and C training (top left), class B accuracy during Class A, B, and C training (top right), and class C accuracy during class A, B, and C training (bottom). The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The results from this experiment demonstrate how class iterated learning introduces additional complexity through the incremental inclusion of data, as reflected in the variable performance of SI and EWC. The findings also highlight that SI's dynamic importance tracking mechanism is more sensitive to task interference and overlapping input spaces, challenges that are particularly pronounced in class iterated learning scenarios [7].

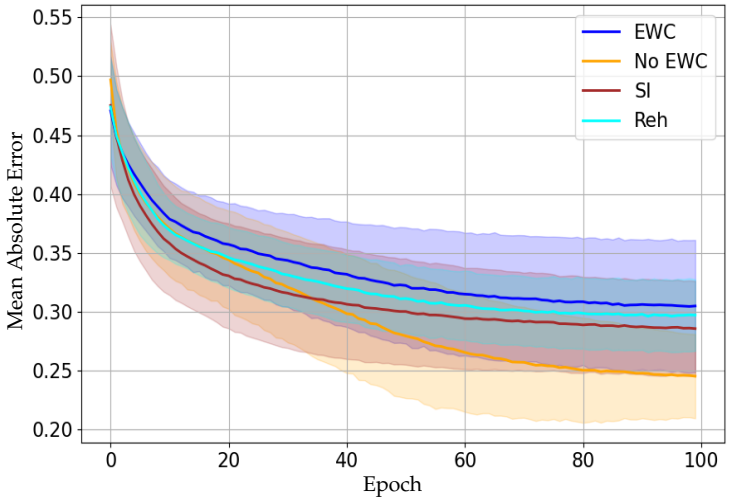


FIGURE 4.11: This chart shows the average performance of class A, class B, and class C per epoch for each algorithm with variance bands. The chart illustrates the average mean absolute error per class for each algorithm. The baseline method achieves the lowest overall MAE, but this is primarily due to its superior ability to learn new tasks, rather than an indication of effective knowledge retention. The baseline method is likely skewed by its strong plasticity, as it quickly adapts to new class distributions, not fully capturing the effects of catastrophic forgetting. Whereas EWC and SI show higher overall error due to their constraints on parameter updates. The wider variance bands for EWC and SI suggest greater instability in their learning trajectories, while Rehearsal remains relatively stable.

The class-iterated toy dataset experiment examined how methods adapted to progressively changing sinewave functions, where frequency and amplitude increased across classes. EWC demonstrated stronger retention of the initial sinewave (Class A) compared with SI but did not preserve prior knowledge as effectively as Rehearsal when transitioning to higher-frequency and higher-amplitude functions. Despite its structured parameter constraints, EWC adapted more slowly than Rehearsal and struggled with Class B, leading to lower initial performance and a more constrained ability to retain its features when learning Class C. However, its relative degradation from Class B to Class C was similar to SI, suggesting that while its absolute retention of Class B was lower, the extent of forgetting followed a comparable trend.

Rehearsal and SI adapted similarly when learning Class B, but SI showed superior retention of Class B knowledge after transitioning to Class C. This suggests

that SI's dynamic tracking of parameter importance effectively preserved intermediate knowledge even as new function variations were introduced. However, when adapting to Class C, SI exhibited the weakest plasticity among the three methods, indicating that its retention mechanisms restricted necessary updates for learning the final function. In contrast, Rehearsal adapted best to Class C, likely benefiting from stored examples that provided direct exposure to the new pattern while maintaining reference points from earlier classes. EWC followed, with moderate adaptability, while SI's constrained updates limited its ability to learn the final function effectively.

SI and EWC exhibited similar levels of variability across transitions, likely due to their reliance on parameter constraints that fluctuated based on the degree of change between classes. Rehearsal, in contrast, demonstrated the most stable performance, as its explicit memory buffer provided a consistent reference for past knowledge, reducing fluctuations in retention and adaptation.

These results highlight the trade-offs inherent in different continual learning approaches. EWC maintained structured retention but faced challenges in adapting to function shifts. SI preserved intermediate knowledge but struggled with plasticity when encountering the final function. Rehearsal consistently balanced adaptation and retention, albeit with some variability across transitions. The findings suggest that the effectiveness of each method depends on the nature of the sequential changes, with Rehearsal excelling in cases requiring continuous flexibility, while EWC and SI offer varying degrees of stability at different stages of learning. A summary of knowledge retention versus plasticity performance for class A can be seen in Table C.13 and Table C.14, for class B in Table C.15, while the performance of class C is summarised in Table C.16.

4.3.2 Benchmark Dataset

The first chart shows the accuracy of class A during training on class A, B, and C. The second chart shows the accuracy of class B during training on class A, B, and C. The final chart shows class C accuracy during training on class A, B, and C.

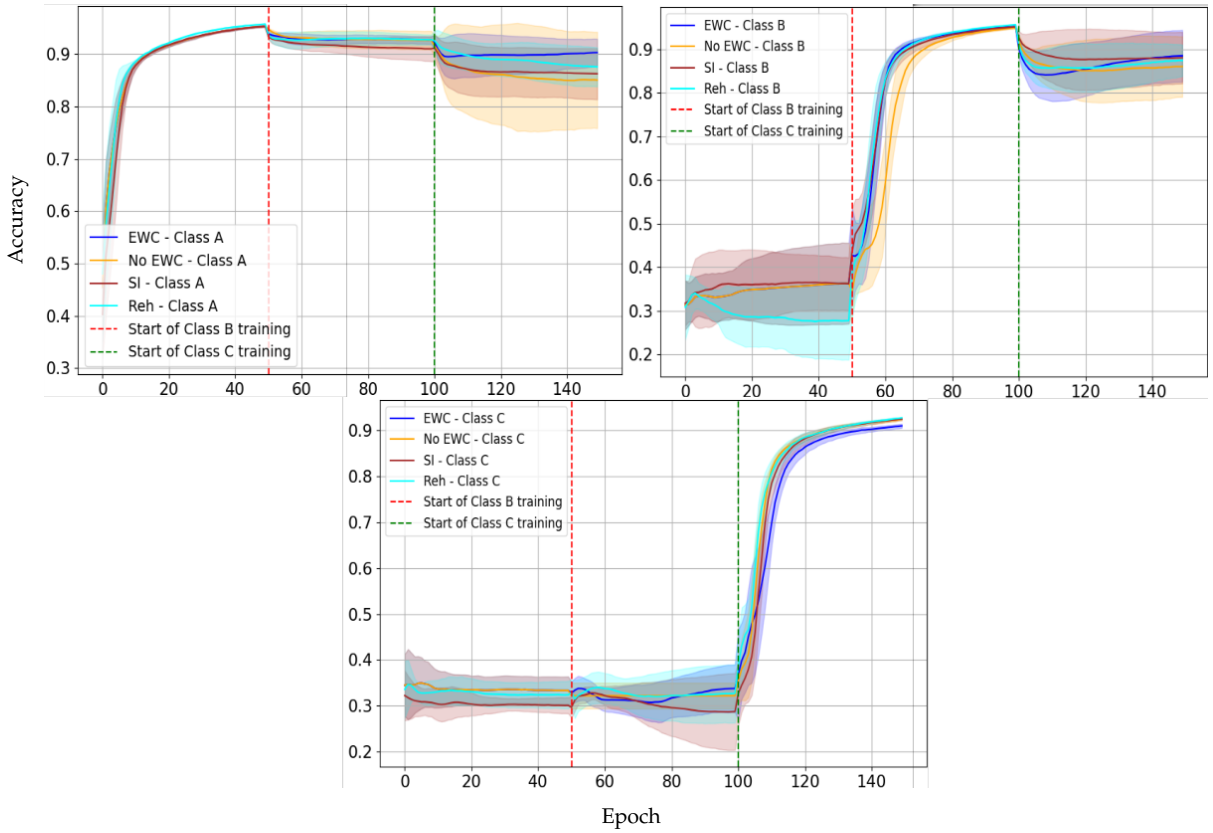


FIGURE 4.12: Class A accuracy during class A, B, and C training (top left), class B accuracy during class A, B, and C training (top right), and class C accuracy during Class A, B, and C training (bottom). The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. This experiment revealed key insights into how different algorithms manage knowledge retention and adaptability. EWC demonstrated strong retention during Class C training, with minimal variability and an upward trend in performance after an initial drop, highlighting its ability to mitigate forgetting. In contrast, SI displayed greater variability, particularly when transitioning to Class C, underscoring its sensitivity to task interference. These findings emphasise the strengths of EWC and Rehearsal in managing the complexities of class iterated learning with overlapping feature spaces.

The class iterated benchmark experiment revealed key insights into how different algorithms manage knowledge retention and adaptability. All the algorithms performed well, benefitting from the shared representations across the MNIST digit

classes. EWC demonstrated strong retention during Class C training, with minimal variability and an upward trend in performance after an initial drop, highlighting its ability to mitigate forgetting. In contrast, SI's dynamic parameter importance tracking led to increased variability, particularly when transitioning to Class C, underscoring its sensitivity to task interference. Rehearsal maintained the best balance between retention and adaptability, leveraging its explicit memory mechanism. Overall, these findings underscore the strengths of EWC and Rehearsal in managing the complexities of class-iterated learning with overlapping feature spaces. A summary of knowledge retention versus plasticity performance for class A can be seen in Table C.17 and Table C.18, for class B in Table C.19, while the performance of class C is summarised in Table C.20.

4.3.3 Real-World Dataset

The first chart shows the accuracy of class A after training on class A, B, C, and D. The second chart shows the accuracy of class B during training on all classes. The third chart is the accuracy of class C and the final chart of class D.

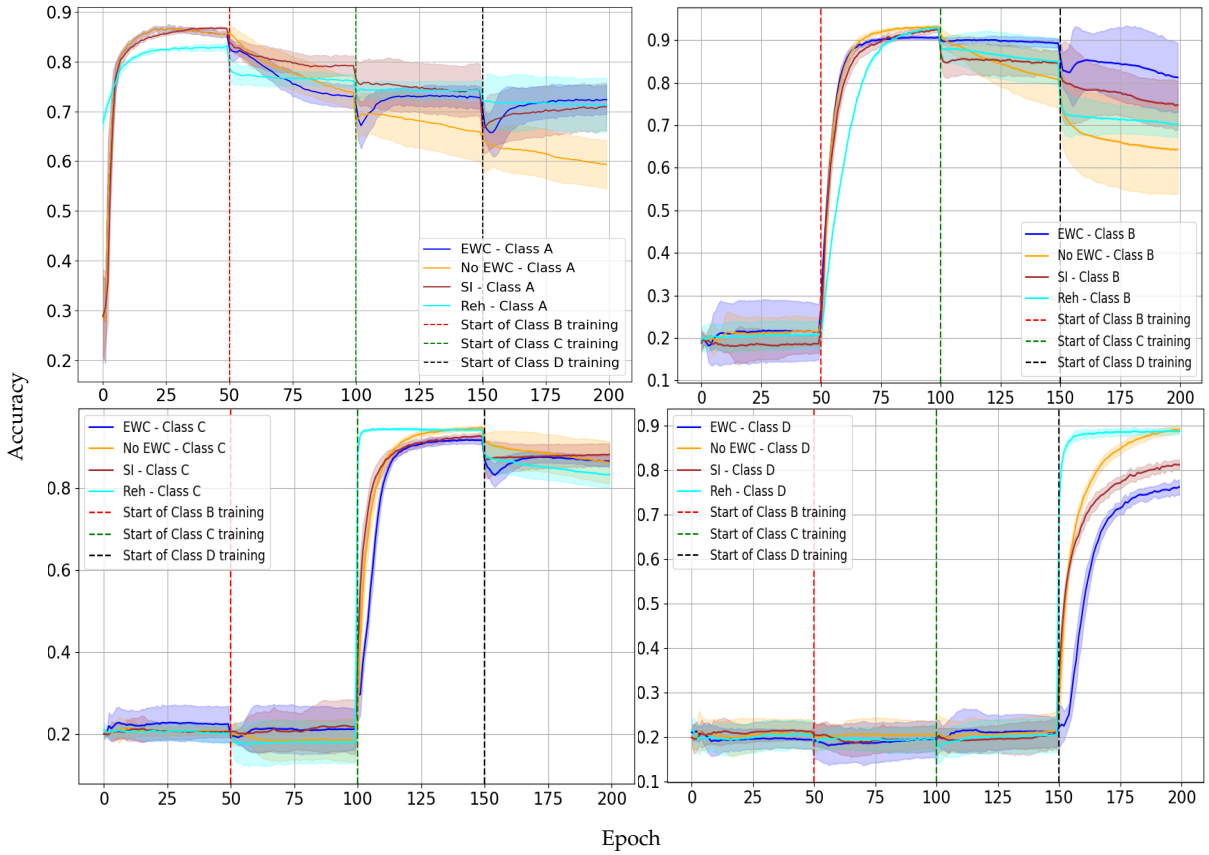


FIGURE 4.13: Class A accuracy during class A, B, C, and D training (top left), class B accuracy during Class A, B, C, and D training (top right), class C accuracy during class A, B, C, and D training (bottom left), and class D during Class A, B, C, and D training (bottom right). The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The performance variability observed for Class A and B during subsequent class training underscores the complexities inherent in class-iterated learning with real-world datasets. The variability in results for SI and EWC highlights their sensitivity to task-level parameter interference. Notably, EWC's performance initially drops but demonstrates a capacity to recover during training, reflecting its trade-off between retention and plasticity. This trade-off is particularly evident during Class D training, where EWC balances preserving earlier knowledge while adapting to new, more challenging classes.

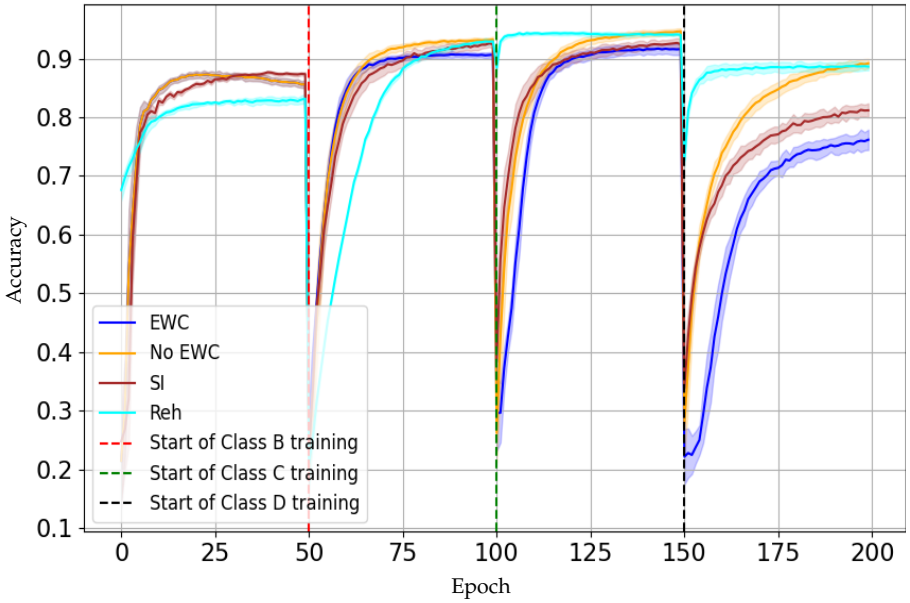


FIGURE 4.14: The chart shows the performance of class A during class A training, class B during class B training, class C during class C training, and class D during class D training. The shaded regions represent 2 standard deviations above and below the mean performance for each algorithm. The chart highlights the trade-off in knowledge retention and adaptability which becomes particularly evident during class D training where EWC and SI struggle to adapt to class D with EWC struggling more so. SI's selective consolidation enables it to allocate capacity more efficiently for new tasks.

The real-world dataset posed the greatest challenge, with progressively distinct topic groups amplifying the trade-offs between retention and adaptability. The performance variability observed for class A and B during subsequent class training underscores the complexities inherent in class-iterated learning with real-world datasets. This variability in results for SI and EWC highlights their sensitivity to task-level parameter interference. Notably, EWC's performance initially drops but demonstrates a capacity to recover during training, reflecting its trade-off between retention and plasticity. However, EWC retained class A and B knowledge the best as training progressed through the different classes, notably outperforming with class B retention, without suffering adaptability until class D training, which is consistent with prior findings by Huszár [29] and Ahn et al. [30], who note regularisation penalties can hinder performance in complex multi-task settings. SI displayed strong retention in earlier phases, but underperformed EWC in Class B retention, and faced limitations in adapting to the more complex class D tasks, reflecting its sensitivity to task-specific interference, however, it outperformed EWC. Overall,

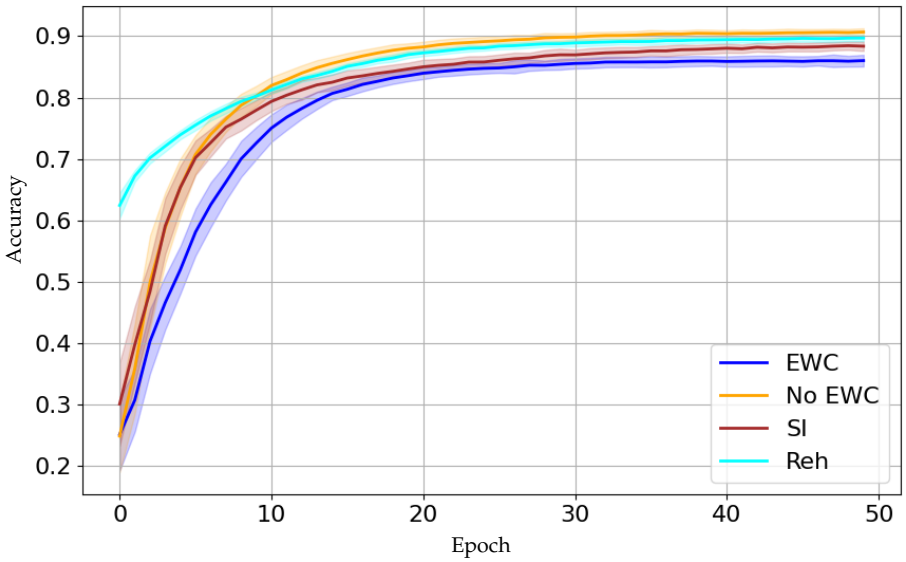


FIGURE 4.15: This chart shows the average performance of class A, class B, class C, and class D per epoch for each algorithm with variance bands.

EWC emerged as the stand out method in terms of retention across all classes, with adaptability suffering only during class D training. A summary of knowledge retention versus plasticity performance for class A can be seen in Table C.21, Table C.22, and Table C.23, for class B in Table C.24 and Table C.25, for class C in Table C.26, while the performance of class D is summarised in Table C.27.

Overall, the class iterated experiments revealed distinct strengths and limitations for each method. Rehearsal proved to be the most versatile, maintaining consistent performance across tasks of varying complexity. EWC excelled in knowledge retention, particularly in structured datasets, but its adaptability diminished as task complexity increased. SI offered a middle ground, balancing retention and adaptability but with greater variability, especially in overlapping feature spaces. The baseline method, while highly adaptable, was unsuitable for sequential learning due to catastrophic forgetting. These findings underscore the importance of tailoring methods to the specific challenges posed by task complexity and dataset characteristics in class-iterated learning scenarios.

Chapter 5

Discussion

This chapter discusses the broad findings of the experiments conducted in this study, focusing on the interplay between retention and adaptability across different continual learning methods and the variabity of these results. Retention refers to a model's ability to preserve knowledge from previous tasks without significant degradation in performance, while adaptability denotes its capacity to effectively learn new tasks without excessive constraints from prior learning. The results presented in the previous chapters highlight the trade-offs between these two aspects and how different methods, including EWC, SI, and Rehearsal, balance them across various experimental settings.

5.1 Performance Evaluation Analysis

This study assessed the performance of EWC across three sequential learning scenarios—task iterated, domain iterated, and class iterated learning using datasets of increasing complexity: a toy dataset, a benchmark dataset, and a real-world text-based dataset. The experiments revealed valuable insights into EWC's strengths, limitations, and comparative performance relative to other methods like Rehearsal, Progressive Neural Networks (PNN), and Synaptic Intelligence (SI).

EWC demonstrated consistent strengths in knowledge retention, particularly in preserving earlier task performance. In task-iterated learning, EWC effectively mitigated forgetting in both the TIL-SinewaveShift and TIL-MNIST experiments, strongly outperforming SI in the TIL-MNIST experiment with more reliable results by preserving critical parameters through its Fisher Information Matrix. In domain-iterated learning, EWC performed comparably to SI and Rehearsal in the

DIL-SplitMNIST experiment which highlighted the trade-off in retention and plasticity. EWC outperformed SI, and performed on par with Rehearsal, in domain A retention in the DIL-ReutersNewsGroup experiment but suffered with domain B adaptability compared with Rehearsal. This outperformance highlights SI's comparative weakness to EWC in scenarios with larger domain shifts, for example, the 20 News Group dataset which contains more general topics versus the Reuters dataset which is more business and finance orientated likely resulting in different term distributions. Zenke et al. [7] acknowledge that deep learning models, including those utilising SI, struggle when data distribution changes substantially during learning. The DIL-SplitSinewave experiment saw EWC outperform Rehearsal and SI in knowledge retention with low variability without sacrificing plasticity compared with SI, and only moderately underperforming Rehearsal in adaptation. The increased variability and inferior retention of SI in the TIL-MNIST and DIL-SplitSinewave experiments is likely due to its sensitivity to overlapping input spaces as well as the way which SI accumulates contributions to loss over time. This, combined with the dynamic importance updates, can lead to instability and increased variability when shifting domains or tasks. In the CIL-SinewaveClasses experiment, EWC outperformed Rehearsal and SI in knowledge retention, particularly for class A during class C training, without realising subpar adaptation. EWC retained knowledge well in the CIL-IncrementalNews experiment, outperforming the other algorithms particularly in class B retention with adaptability only suffering during class D training, and only marginally so compared with SI. The class iterated toy dataset experiment produced mixed results, with no single method clearly outperforming the other algorithms.

EWC's adaptability to new domains was constrained in the real-world experiments, where greater noise and complexity challenged its regularisation mechanisms. However, it performed comparably in the task iterated scenario, below only PNN. In the domain iterated scenario it adapted better than SI but worse than Rehearsal and PNN. Class iterated learning saw EWC realise the worst adaptation, underperforming all other algorithms only on class D learning, highlighting its limitation in multi-class settings [29, 30]. These challenges underscore the difficulty in balancing retention and plasticity in overlapping input spaces. EWC's slower adaptation to new classes highlighted the limitations of its conservative parameter updates,

which, while effective for retention, hindered its ability to respond quickly to new tasks.

Comparative analyses revealed nuanced performance dynamics across methods. Rehearsal emerged as the most balanced method, excelling in both retention and adaptability, with superior performance in the real-world and benchmark experiments. PNN demonstrated exceptional task-specific retention and plasticity, however the architecture of PNN requires the use of a task identifier. EWC and SI mostly exhibited comparable retention capabilities, but EWC's conservative regularisation approach provided slightly better stability, particularly in earlier tasks. However, SI displayed higher variability, likely due to its dynamic importance tracking, which is more sensitive to task interference and overlapping input spaces.

The experiments also underscored the scalability of EWC across complexity levels. In simpler scenarios, such as the task-iterated experiments EWC maintained stable performance, effectively managing task interference and overlapping features. In contrast, the DIL-SplitMNIST and class iterated experiments exposed its limitations, with increasing task complexity amplifying challenges in balancing retention and adaptability. These results suggest that EWC is most effective in environments with clear task separability and minimal feature overlap, where its regularisation mechanism can effectively preserve critical parameters without significantly hindering new learning. However, as task complexity increases, particularly in scenarios with shared feature spaces or gradual class introduction, EWC struggles to maintain adaptability, leading to greater performance degradation. This highlights a key limitation of regularisation methods in settings requiring flexible knowledge integration, suggesting that EWC alone may not be sufficient for continual learning in high-complexity domains.

5.2 Performance and Variability of the Regularisation Methods

This section focuses on a direct comparison between EWC and SI, given their shared regularisation-based foundations. While both aim to reduce forgetting by penalising parameter drift, their differing strategies for importance estimation result in

notable differences in both retention and variability. The performance and variability comparison reveals distinct strengths and weaknesses across a range of learning scenarios. Both methods effectively mitigated catastrophic forgetting but differed in their retention, adaptability, and consistency under varying dataset complexities.

In terms of performance, EWC and SI were generally comparable, showcasing strong retention across most experiments. In task-iterated learning, both methods performed similarly in the TIL-Sinewave and TIL-TopicPlace experiments, although SI marginally outperformed EWC in adaptation in the TIL-TopicPlace experiment likely due to its ability to dynamically adjust importance weights, allowing it to balance retention and plasticity more effectively in the shift from topic classification to place identification. However, in the TIL-MNIST experiment, EWC significantly outperformed SI in knowledge retention while only marginally underperforming in adaptability, highlighting its robustness over SI in scenarios with overlapping feature spaces. For domain-iterated learning, both methods were comparable in the DIL-SplitMNIST, while EWC outperformed SI in the DIL-SplitSinewave especially in retention and retention stability as SI's continuous updates and accumulation of loss contributions can lead to instability especially when shifting between domains with overlapping feature representations. The DIL-ReutersNewsGroup experiment showcased the superior performance of EWC in retention in learning scenarios with sudden and or large domain shifts where EWC explicitly ties important parameters to their prior values making it better at maintaining representations from domain A, [8], while SI's gradual adjustments can struggle when domains have no or fewer overlapping features and when there are abrupt or sudden domain shifts [7]. In class-iterated learning, EWC achieved marginally better retention in the CIL-IncrementalMNIST due to its conservative regularisation mechanism, while SI showed superior adaptability in the CIL-IncrementalNews, demonstrating its flexibility in handling complex scenarios. This adaptability of SI is due to its selective consolidation, which enables it to allocate capacity more efficiently for new tasks [7].

From a variability perspective, EWC consistently showed lower variability compared with SI across most scenarios. This is attributed to EWC's reliance on the Fisher Information Matrix, which stabilises critical parameters and mitigates task interference, ensuring more consistent performance even in overlapping or highly

complex feature spaces. Conversely, SI exhibited higher variability, particularly in the task-iterated benchmark, domain-iterated toy, and class-iterated benchmark experiments. Its dynamic importance tracking, while enabling greater adaptability, introduced sensitivity to task interference and overlapping input spaces, especially in scenarios with shared input features or significant task overlap.

EWC's stability benefits stem from its static Fisher Information Matrix, which "locks in" critical parameters and prevents excessive updates during subsequent tasks. This approach proved especially effective in overlapping feature spaces, such as the benchmark dataset, where shared features like MNIST digit curves were reused across tasks. On the other hand, the dynamic nature of SI results in the parameter importance values being recalculated during training, which makes it more susceptible to variability when overlapping features create task-specific interference. For example, the TIL-MNIST experiment highlighted SI's increased variability due to the overlapping classification tasks, while the DIL-SplitSinewave exposed challenges in isolating task-specific parameters for periodic characteristics.

In simpler datasets or well-separated domains, such as the TIL-SinewaveShift experiment, both EWC and SI displayed similar variability, as tasks offered clearer boundaries. However, as complexity increased, particularly in all the class-iterated experiments, DIL-SplitMNIST, DIL-SplitSinewave, and the TIL-MNIST experiments, SI struggled to manage interference, leading to greater variability. This increased variability of SI is highlighted in Figure 4.6 and Figure 4.3 which show the large spread between individual runs. In addition, the latter figure showcases SI's sensitivity to importance values which is likely due to the less structured nature of SI's regularisation effect compared with ewc. This dynamic highlights the trade-offs between EWC's stability and SI's adaptability, with EWC excelling in retaining prior knowledge in stable environments and SI proving more flexible but less consistent in scenarios with overlapping features.

5.3 Dataset Characteristics vs Learning Scenario

The study results suggest that while dataset characteristics play a more dominant role in determining performance, the learning scenario also adds an essential layer

of complexity that cannot be overlooked.

Dataset characteristics, such as feature overlap, task complexity, and shared representations, significantly influence the performance and variability of methods. For instance, EWC showed less variability in the DIL-ReutersNewsGroup experiment compared with the DIL-SplitSinewave experiment, even within the same learning scenario. This difference is likely due to the better task separation in the real-world dataset, where distinct domains offered clearer boundaries than the periodic similarities in the sinewave function. Similarly, task complexity played a pivotal role in the CIL-IncrementalNews experiment, where EWC struggled more due to the increasing dissimilarity and complexity of tasks compared to the CIL-SinewaveClasses and CIL-IncrementalMNIST experiments. Additionally, overlapping feature spaces in MNIST allowed EWC to perform consistently across task and domain iterated settings, underscoring the influence of shared low-level features on outcomes.

The learning scenario introduces unique challenges that shape how methods interact with dataset characteristics. In task-iterated learning, sequentially presented objectives (e.g., sine wave prediction vs. phase-shifted sine wave) introduce task-specific interference, requiring models to manage transitions without eroding earlier knowledge. In domain-iterated learning, where tasks share objectives but differ in input distributions (e.g., even/odd classification across domains), methods must balance domain generalisation with the retention of domain-specific knowledge. Meanwhile, class-iterated learning stresses retention through the incremental addition of classes, which can amplify task interference, as seen in the variability of EWC and SI in these scenarios.

While dataset characteristics can strongly influence outcomes, the learning scenario shapes the nature of challenges faced by models. Together, these factors highlight the interplay between dataset and scenario design, underscoring the need to consider both when evaluating methods for sequential learning.

Chapter 6

Conclusion

6.1 Concluding Remarks

This study explored the application of EWC and other continual learning methods in sequential learning scenarios beyond their traditional use in vision tasks. Specifically, the effectiveness of EWC was evaluated in mitigating catastrophic forgetting across text-based domains, leveraging datasets such as 20 Newsgroups and Reuters News, and in regression based tasks. By demonstrating EWC's adaptability to these modalities, a significant gap in existing literature was addressed, which has largely focused on vision tasks like MNIST or CIFAR.

The results from the class-iterated experiments revealed that SI and EWC were able to effectively preserve knowledge in earlier tasks; however, their adaptability to new tasks remained a challenge, especially in environments with overlapping feature spaces. This aligns with findings by Zenke et al. [7] regarding SI's retention capabilities, while Kirkpatrick et al. [8], Huszár [29] highlight EWC's ability to stabilise critical parameters via the FIM when balancing retention and plasticity. For EWC, the FIM becomes less effective at distinguishing task-specific importance as overlap increases, leading to a dilution effect where earlier parameter constraints interfere with new learning. In addition, as task complexity increases or overlaps shift to higher-level parameters, EWC struggles with plasticity. SI offers greater flexibility thanks to its dynamic importance tracking, but it is more susceptible to variability due to its sensitivity to interference when there are shared features or overlapping distributions. In scenarios with sudden domain or task shifts, the explicit constraining of key weights by EWC allows for better performance compared with SI, which aligns with findings by Kirkpatrick et al. [8]. The dynamic nature

of SI's importance tracking can make it more sensitive to such shifts, as noted by Zenke et al. [7]. The FIM used by EWC provides a more structured penalty on parameter updates, which stabilises forgetting and results in less spread across different lambda values. In contrast, SI updates the importance score dynamically, making it more sensitive to lambda variations [8, 29, 7]. Across all scenarios and the varying degrees of dataset complexity, EWC produced more stable and reliable results compared with SI.

These findings highlight the importance of employing the proper methods based on the characteristics of the data and task to ensure continual learning. They also provide a foundation for further research into hybrid approaches that combine the retention strength of EWC with the flexibility of methods like Rehearsal. Such approaches may be especially valuable in scenarios requiring both retention of earlier knowledge and efficient learning of new tasks.

This study makes several key contributions to the field of continual learning. First, it demonstrates EWC's applicability to text-based and regression based domains, addressing a significant gap in the literature where most applications of EWC have been restricted to vision tasks. Second, by evaluating EWC across task, domain, and class iterated scenarios, it provides a comprehensive assessment of its strengths and limitations under varying levels of task complexity and interference. Third, the insights into EWC's retention and adaptability trade-offs deepen our understanding of how regularisation techniques can be refined for different data modalities and task structures. Lastly, through the meticulous design of datasets of increasing complexity that adhere to the nuanced definitions of the three distinct learning scenarios: task-iterated, domain-iterated, and class-iterated learning. These datasets were carefully crafted to isolate and evaluate specific challenges inherent to each learning scenario, providing a robust framework for systematic analysis. The toy datasets facilitated controlled experimentation in simplified environments, enabling the identification of foundational patterns in retention and adaptability. The benchmark datasets provided a standardised platform for comparing methods under moderate complexity, while the real-world datasets introduced practical challenges such as noise, variability, and scalability, reflecting the realities of applied machine learning. By addressing the unique characteristics of each learning scenario, these datasets ensure a comprehensive evaluation of methods like EWC,

offering valuable insights into their strengths, limitations, and applicability across diverse tasks and domains.

The findings also reveal that dataset characteristics, such as feature overlap, task separability, and complexity, play a significant role in determining algorithm performance and variability than the specific learning scenario. However, the learning scenario still influences task structuring and presentation, which can exacerbate or mitigate the challenges posed by the dataset. Task-iterated scenarios introduce task complexity, as models must adapt to entirely new high-level representations without eroding earlier features. Domain-iterated learning introduces complexity by leveraging shared low-level representations while requiring adaptation to domain-specific features. Class-iterated learning typically introduces feature overlap across classes, combined with increasing task complexity, as models handle interference from shared features while preserving distinct earlier representations. The interplay between the learning scenario and dataset characteristics significantly influences the challenges and outcomes of sequential learning and must be considered holistically.

Overall, this study contributes to the growing understanding of EWC's applicability across diverse sequential learning scenarios, demonstrating its utility in tasks ranging from synthetic to real-world settings. While EWC's strengths in retention make it a reliable choice for tasks with moderate complexity, its limitations in adaptability underscore the importance of considering both task characteristics and dataset features when selecting or designing continual learning methods.

6.2 Ethical Considerations

The research report adhered to all relevant ethical standards regarding the use and handling of datasets. All real-world datasets were publicly available and used in compliance with their respective licensing terms, ensuring there was no infringement on intellectual property rights. Additionally, no personally identifiable or sensitive data was used in this study. The findings of this work do not pose foreseeable risks in terms of propagating bias or injustice through the use of machine learning in the future. Nonetheless, responsible deployment of continual learning

methods should always consider potential biases in training data and their impact on decision-making processes.

6.3 Limitations

While this study provides valuable insights into the effectiveness of EWC and other methods across various sequential learning scenarios, several limitations must be acknowledged.

The study was conducted within the constraints of available computational resources, which influenced the extent of hyperparameter exploration. While significant effort was made to optimise hyperparameters for methods like EWC and SI, further computational resources would have allowed for a more fine-grained search. Given that regularisation-based methods are highly sensitive to hyperparameter selection, a broader search space might have led to additional insights into performance trends and variability. However, the key hyperparameters were carefully tuned to balance retention and adaptability, ensuring a rigorous evaluation of each method.

Secondly, all task-iterated and domain-iterated experiments involved only two tasks or domains, respectively. While this setup allowed for a clear assessment of the retention and adaptability trade-offs, it does not fully capture the challenges of more extensive sequential learning scenarios with multiple tasks or domains. Real-world continual learning often involves a larger number of tasks with more complex interactions, which could reveal additional limitations or strengths of the methods under evaluation.

Lastly, the study focused exclusively on specific types of datasets: toy datasets, benchmark datasets, and real-world text-based datasets. Although these datasets were carefully designed to adhere to the definitions of the three learning scenarios, they do not encompass all possible data modalities or task structures. For instance, multimodal datasets or reinforcement learning scenarios were not explored, which could limit the generalisability of the findings to other applications of continual learning.

These limitations highlight the need for further research to address these gaps, such as incorporating additional tasks, exploring other types of datasets, and leveraging more computational resources to refine hyperparameter selection.

6.4 Further Research

This study highlights several avenues for future exploration in the field of continual learning. One promising direction involves developing hybrid approaches that combine EWC with methods like Rehearsal or Synaptic Intelligence. Such combinations could potentially balance retention and adaptability more effectively, especially in scenarios with overlapping input spaces or long task sequences. Another area for investigation is scaling EWC to handle more complex domains, such as multimodal datasets or tasks involving continuous data streams, which would provide deeper insights into its scalability and performance in real-world applications.

Optimising parameter tuning during training presents another valuable opportunity. Dynamically adjusting regularisation strength could help address the trade-offs between retention and adaptability that were observed in this study. Additionally, exploring how EWC performs across a wider variety of task types, including unsupervised learning and reinforcement learning scenarios, could significantly broaden its applicability and utility in diverse learning environments.

Finally, integrating EWC with dynamic architectures, such as progressive or expandable neural networks, could help mitigate its challenges when managing a large number of tasks or overlapping feature spaces. These directions collectively offer a rich framework for advancing EWC's capabilities and enhancing its role in the evolving landscape of sequential learning.

Bibliography

[1] 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. doi: 10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

[2] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017. URL https://arxiv.org/abs/1706.08840.

[3] Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends In Cognitive Sciences*, 24:1028–1040, 2020. URL https://doi.org/10.1016/j.tics.2020.09.004.

[4] Mark D. McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023. URL https://doi.org/10.48550/arXiv.2307.02251.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. *arXiv preprint arXiv:1711.09601*, 2018. URL https://arxiv.org/abs/1711.09601.

[6] Sebastian Lee, Stefano Sarao Mannelli, Claudia Clopath, Sebastian Goldt, and Andrew Saxe. Maslow's hammer for catastrophic forgetting: Node re-use vs node activation. *arXiv preprint arXiv:2205.09029*, 2022. URL https://arxiv.org/abs/2205.09029.

[7] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017. URL https://arxiv.org/abs/1703.04200.

[8] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. URL https://www.pnas.org/content/114/13/3521.

[9] Gido M. Van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2018. URL https://arxiv.org/abs/1904.07734.

[10] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733v3*, 2018. URL https://arxiv.org/abs/1805.09733v3.

[11] Ying Xu, Antonio Jose Jimeno Yepes Xu Zhong, and Jey Han Lau. Forget me not: Reducing catastrophic forgetting for domain adaptation in reading comprehension. *arXiv preprint arXiv:1911.00202v3*, 2019. URL https://arxiv.org/abs/1911.00202v3.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

[13] Pablo Sprechmann, Siddhant M. Jayakumar, Jack W. Rae, Alexander Pritzel, Adrià Puigdomènech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. Memory-based parameter adaptation. *arXiv preprint arXiv:1802.10542v1*, 2019. URL https://arxiv.org/abs/1802.10542v1.

[14] Nicolas Masse, Gregory Grant, and David Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *arXiv preprint arXiv:1802.01569*, 2018. URL https://arxiv.org/abs/1802.01569.

[15] Chrisantha Fernando, David Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels large-scale learning. *Nature Communications*, 8:1–12, 2017.

[16] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7:123–146, 1995. URL https://doi.org/10.1080/09540099550039318.

[17] Jonathan Schwarz, Jelena Luketina, Wojciech Marian Czarnecki, Agnieszka Grabska-Barwińska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, pages 4528–4537. PMLR, 2018.

[18] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3264–3272, 2016. URL https://arxiv.org/abs/1606.04671.

[19] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

[20] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 348–358. Curran Associates, Inc., 2019.

[21] 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):4069, 2020. doi: 10.1038/s41467-020-17866-2.

[22] Ameya Prabhu, Philip HS Torr, and Puneet Kumar Dokania. Gdumb: A simple baseline method for continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 524–540. Springer, 2020. doi: 10.1007/978-3-030-58568-6_31.

[23] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. URL http://yann.lecun.com/exdb/mnist/.

[24] TheDevastator. Uncovering financial insights with the reuters dataset. Kaggle, 2023. URL https://www.kaggle.com/datasets/thedevastator/uncovering-financial-insights-with-the-reuters-2.

[25] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Machine Learning Conference (ICML'95)*, pages 331–339, 1995.

[26] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[27] Y. C. Hsu, Y. C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *NeurIPS Continual Learning Workshop*, 2018. URL https://arxiv.org/abs/1810.12488. arXiv:1810.12488.

[28] C. V. Nguyen, Y. Li, T. Bui, and R. E. Turner. Variational continual learning. *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.10628. arXiv:1710.10628.

[29] Ferenc Huszár. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2018.

[30] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Continual learning through expandable elastic weight consolidation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 272–281. PMLR, 2019.

Appendix A

Implementation Details and Technical Contributions

To ensure robust experimentation, all models were implemented using Python with PyTorch as the primary deep learning framework. While some components such as tokenization and pre-trained architectures (e.g., DistilBERT) were accessed via the Hugging Face Transformers library, the core continual learning algorithms—including EWC, SI, and PNN—were implemented from based on the original research papers. This required a detailed understanding of the mathematical formulations and the careful integration of custom regularisation terms into the training loop.

Beyond model implementations, all training loops, evaluation metrics, and experiment tracking mechanisms were developed manually to accommodate various learning paradigms (task-, domain-, and class-iterated scenarios). Custom PyTorch Dataset and DataLoader classes were created to structure the data appropriately, and experiments were repeated multiple times with different seeds to enable meaningful averaging and standard deviation calculations. These implementation efforts reflect a significant level of technical involvement beyond simple usage of out-of-the-box libraries.

Appendix B

Hyperparameter Settings

TABLE B.1: Training Settings Across Experiments

Learning Scenario	**Task-Iterated Learning**			**Domain-Iterated Learning**			**Class-Iterated Learning**		
Hyperparameter	Toy Dataset	Bench	Real-World	Toy Dataset	Bench	Real-World	Toy Dataset	Bench	Real-World
Learning Rate	0.0001	0.00001	0.0001	0.0001	5E-07	0.0001	0.0001	1E-06	0.0001
Batch Size	64	64	64	64	64	32	64	64	32
Epochs	100	50	100	100	100	50	100	50	50
EWC Training Function	Task-Defined	Incremental	Task-Defined	Task-Defined	Incremental	Incremental	Incremental	Incremental	Incremental
EWC Importance	1E+15	1E+05	5E+07	1.25 E+12	1500	50000	16;400	1500	150;150;500
SI Omega	5E+11	900	1E+05	1E+09	9600	9E+13	9E+06; 9E+06	5;5	2;2;2
Rehearsal Training	Replay Ratio	Buffer	Replay Ratio	Replay Ratio	Buffer	Buffer	Buffer	Buffer	Buffer
Buffer Size/Replay Ratio	60%	750	55%	85%	2000	4000	50	1000	4000
Evaluation	Regular	Regular	Regular	Regular	Regular	Regular	Masked	Masked	Masked
Loss Function	MSE	Cross Entropy	Cross Entropy	MSE	BCE Logits	Cross Entropy	MSE	Cross Entropy	Cross Entropy
Code Name	TIL-Sine-wave-Shift	TIL-MNIST	TIL-Topic-Place	DIL-Split-Sine-wave	DIL-Split-MNIST	DIL-Reuters-News-Group	CIL-Sine-wave-Classes	CIL-Incremental-MNIST	CIL-Incremental-News

Appendix C

Results

C.1 Task Iterated Learning

C.1.1 Toy Dataset

TABLE C.1: Summary of MAE for task A after training on task A and after task B training. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Task A MAE	Mean MAE After B	Max MAE After B	Min MAE After B	Upper Bound	Lower Bound
EWC	**0.009028**	0.203859	0.205163	0.202256	0.205657	0.202061
Baseline	0.012242	0.891465	1.284128	0.472704	1.418037	0.364894
PNN	0.013667	0.130995	0.297887	0.038914	0.304121	-0.042131
SI	0.016142	0.206001	0.206721	0.205336	0.207138	0.204493
Rehearsal	0.013189	**0.010394**	**0.017279**	**0.005278**	0.018648	0.002139

TABLE C.2: Summary of MAE for Task B After Task B Training. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final MAE	Max Final MAE	Min Final MAE	Upper Bound	Lower Bound
EWC	0.017011	0.017130	0.016902	0.017127	0.016895
Baseline	0.019119	0.038426	0.007288	0.040533	-0.002294
PNN	**0.016360**	0.028732	0.006416	0.031005	0.001714
SI	0.016803	**0.017056**	0.016539	0.016986	0.016610
Rehearsal	0.018378	0.032179	**0.004283**	0.039532	-0.002776

C.1.2 Benchmark Dataset

TABLE C.3: Summary of Accuracy for Task A After Training on Task A and After Training on Task B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After Task A	Mean After Task B	Max After Task B	Min After Task B	Upper Bound	Lower Bound
EWC	**0.960133**	0.944267	0.956000	0.904667	0.97283	0.915711
Baseline	**0.960133**	0.814400	0.928333	0.631667	0.972103	0.656697
PNN	0.947000	**0.975491**	**0.978583**	**0.970833**	0.980644	0.970338
SI	0.958133	0.849467	0.938000	0.616000	1.041476	0.657457
Rehearsal	0.947633	0.961600	0.971333	0.956667	0.971940	0.951260

TABLE C.4: Summary of Accuracy for Task B After Training on Task B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final Accuracy	Max Final Accuracy	Min Final Accuracy	Upper Bound	Lower Bound
EWC	0.922000	0.947000	0.907000	0.946150	0.897850
No EWC	0.983200	0.986000	0.978000	0.987492	0.978908
PNN	**0.986176**	**0.988083**	**0.982250**	0.989392	0.982960
SI	0.953300	0.964667	0.936000	0.970752	0.935848
Rehearsal	0.983067	0.986000	0.977667	0.988220	0.977913

C.1.3 Real-World Dataset

TABLE C.5: Task A Accuracy After Training on Task A and After Training on Task B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After Task A	Max After Task B	Min After Task B	Mean After Task B	Upper Bound	Lower Bound
EWC	0.895138	0.880146	0.864593	0.871729	0.880066	0.863392
No EWC	0.895138	0.682525	0.602928	0.654803	0.697989	0.611617
PNN	**0.913060**	**0.910339**	**0.906679**	**0.908783**	0.910796	0.906770
SI	0.895138	0.881976	0.878317	0.879780	0.882124	0.877437
Rehearsal	0.895138	0.859103	0.853614	0.856450	0.860413	0.852487

TABLE C.6: Final Accuracy for Task B After Training on Task B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final Accuracy	Max Final Accuracy	Min Final Accuracy	Mean After Task B	Upper Bound
EWC	0.789387	0.800549	0.779506	0.802501	0.776273
Baseline	**0.980787**	**0.983532**	**0.974382**	0.986456	0.975117
PNN	0.851784	0.853614	0.849954	0.854619	0.848949
SI	0.790759	0.796889	0.784081	0.798985	0.782533
Rehearsal	0.835865	0.842635	0.827081	0.845021	0.826708

C.2 Domain Iterated Learning

C.2.1 Toy Dataset

TABLE C.7: Domain A MAE After Training on Domain A and After Domain B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Domain A After A	Mean MAE After B	Max MAE After B	Min MAE After B	Upper Bound	Lower Bound
EWC	0.0654	**0.3861**	**0.5257**	0.2668	0.5630	0.2091
No EWC	0.0654	0.9795	1.1510	0.5888	1.2749	0.6841
SI	0.0654	0.5189	1.1412	**0.2581**	1.1042	-0.0665
Rehearsal	0.0654	0.4786	0.6203	0.3266	0.6461	0.3111

TABLE C.8: Final MAE for Domain B During Domain B Training. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final MAE	Max Final MAE	Min Final MAE	Upper Bound	Lower Bound
EWC	0.7447	0.9850	0.3124	1.1421	0.3474
No EWC	0.2294	0.2706	0.1480	0.3141	0.1447
SI	0.7093	0.9168	0.2691	1.1240	0.2947
Rehearsal	**0.3874**	**0.4792**	**0.2636**	0.5150	0.2599

C.2.2 Benchmark Dataset

TABLE C.9: Domain A Accuracy After Training on Domain A and After Domain B. Upper and lower bound values represent mean ± standard deviation, respectively.

Method	Domain A After A	Max Acc After B	Min Acc After B	Mean Acc After B	Upper Bound	Lower Bound
EWC	**0.999644**	0.789795	0.445160	0.638278	0.888169	0.388386
No EWC	**0.999644**	0.340736	0.000000	0.049347	0.249107	-0.150413
SI	0.999271	0.841499	**0.455501**	0.646449	0.883386	0.409511
Rehearsal	0.994971	**0.890659**	0.256993	**0.692426**	0.949520	0.435331

TABLE C.10: Accuracy for Domain B After Domain B Training. Upper and lower bound values represent mean ± standard deviation, respectively.

Method	Mean Final Acc	Max Final Acc	Min Final Acc	Upper Bound	Lower Bound
EWC	0.632956	0.749754	0.518027	0.801325	0.464588
No EWC	**0.967224**	**0.999**	**0.882170**	1.049144	0.885303
SI	0.619453	0.743691	0.458047	0.822182	0.416723
Rehearsal	0.600412	0.879056	0.364962	0.813832	0.386992

C.2.3 Real-World Dataset

TABLE C.11: Domain A Accuracy After Training on Domain A and After Domain B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Domain A After A	Max Acc After B	Min Acc After B	Mean Acc After B	Upper Bound	Lower Bound
EWC	0.834961	0.761500	0.555556	0.675407	0.788349	0.562465
No EWC	0.834961	0.598372	0.496461	0.535810	0.986456	0.975117
SI	0.834961	**0.800426**	0.507785	0.574275	0.739869	0.408680
Rehearsal	0.834961	0.728946	**0.680113**	**0.708953**	0.845021	0.826708

TABLE C.12: Accuracy for Domain B During Domain B Training. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final Acc	Max Final Acc	Min Final Acc	Upper Bound	Lower Bound
EWC	0.634413	0.650472	0.612686	0.658142	0.610684
No EWC	0.780715	0.793103	**0.766284**	0.795523	0.765907
SI	0.621772	0.639676	0.607287	0.607287	0.607272
Rehearsal	**0.789862**	**0.822454**	0.756463	0.826444	0.753279

C.3 Class Iterated Learning

C.3.1 Toy Dataset

TABLE C.13: Class A MAE After Training on Class A and After Class B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Class A After A	Mean MAE After B	Max MAE After B	Min MAE After B	Upper Bound	Lower Bound
EWC	0.083031	0.387149	0.530472	0.286392	0.527841	0.246456
No EWC	0.083031	0.585417	0.601684	0.529009	0.629852	0.540982
SI	**0.062626**	0.432672	0.557936	0.332540	0.571708	0.293636
Rehearsal	0.068908	**0.334147**	**0.380080**	**0.259138**	0.411572	0.256722

TABLE C.14: Class A MAE After Training on Class A, B, and C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	MAE After A	Mean MAE After B	Mean MAE After C	Max MAE After C	Min MAE After C	Upper Bound	Lower Bound
EWC	0.083031	0.387149	0.451089	0.557730	0.354582	0.598243	0.303934
No EWC	0.083031	0.585417	0.646542	0.649878	0.642849	0.650851	0.642234
SI	**0.062626**	0.432672	0.500839	0.623296	0.382445	0.659041	0.342637
Rehearsal	0.068908	**0.334147**	**0.378212**	**0.401725**	**0.331100**	0.416415	0.340010

TABLE C.15: Class B MAE After Training on Class B and After Class C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	MAE After B	Mean MAE After C	Max MAE After C	Min MAE After C	Upper Bound	Lower Bound
EWC	0.333813	0.405140	0.563338	**0.207330**	0.596782	0.213499
No EWC	**0.089496**	0.588444	0.592661	0.585523	0.593071	0.583816
SI	0.276979	**0.352130**	**0.501091**	0.220968	0.531325	0.172935
Rehearsal	0.294494	0.518543	0.563510	0.429597	0.589970	0.447116

TABLE C.16: Final MAE for Class C

Method	Mean Final MAE	Max Final MAE	Min Final MAE	Upper Bound	Lower Bound
EWC	0.385942	0.473560	0.232992	0.529409	0.242474
No EWC	**0.079099**	**0.085287**	**0.077400**	0.083664	0.074535
SI	0.400518	0.464063	0.245690	0.531961	0.269075
Rehearsal	0.287326	0.382376	0.259175	0.354107	0.220545

C.3.2 Benchmark Dataset

TABLE C.17: Class A Accuracy After Training on Class A and After Class B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After A	Mean After B	Max After B	Min After B	Upper Bound	Lower Bound
EWC	0.953919	0.927192	0.938182	**0.914141**	0.941247	0.913137
No EWC	0.953919	0.926727	**0.952121**	0.895556	0.963047	0.890408
SI	0.953616	0.910646	0.934545	0.846667	0.958547	0.862746
Rehearsal	**0.957980**	**0.927596**	0.945051	0.905657	0.947172	0.908020

TABLE C.18: Class A Accuracy After Training on Class A, B, and C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After A	Mean After B	Mean After C	Max After C	Min After C	Upper Bound	Lower Bound
EWC	0.953919	0.927192	**0.903293**	0.941212	**0.850303**	0.956307	0.850279
No EWC	0.953919	0.926727	0.850848	**0.946465**	0.598586	1.035421	0.666276
SI	0.953616	0.919076	0.862828	0.942424	0.796768	0.960622	0.765034
Rehearsal	**0.957980**	**0.934905**	0.876020	0.944040	0.820404	0.950856	0.801184

TABLE C.19: Class B Accuracy After Training on Class B and After Class C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After B	Mean After C	Max After C	Min After C	Upper Bound	Lower Bound
EWC	0.952299	**0.885274**	0.938009	0.750291	1.003319	0.767228
No EWC	0.949447	0.860477	0.926077	0.694121	0.999273	0.721681
SI	0.951775	0.880704	**0.941793**	0.760768	0.998997	0.762412
Rehearsal	**0.956577**	0.874563	0.922293	**0.797439**	0.947213	0.801914

TABLE C.20: Final Accuracy for Class C. Upper and lower bound values represent mean ± 2 standard deviations.

Method	Mean Final Acc	Max Final Acc	Min Final Acc	Upper Bound	Lower Bound
EWC	0.9103	0.9172	0.9031	0.9191	0.9016
No EWC	0.9249	0.9291	0.9192	0.9299	0.9198
SI	0.9251	0.9303	0.9211	0.9301	0.9202
Rehearsal	**0.9581**	**0.9906**	**0.9242**	1.0190	0.8973

C.3.3 Real-World Dataset

TABLE C.21: Class A Accuracy After Training on Class A and After Class B. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	After A	Mean After B	Max After B	Min After B	Upper Bound	Lower Bound
EWC	0.854468	0.733191	0.790071	0.700709	0.780906	0.685477
No EWC	0.854468	0.737730	0.780142	0.683688	0.801196	0.674265
SI	**0.873475**	**0.795319**	**0.860993**	**0.770213**	0.885509	0.745129
Rehearsal	0.831773	0.771064	0.777305	0.758865	0.784799	0.757329

TABLE C.22: Class A Accuracy After Training on Class A, B, and C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean After A	Mean After B	Mean After C	Max After C	Min After C	Upper Bound	Lower Bound
EWC	0.854468	0.733191	0.733759	0.758865	0.686525	0.779958	0.687560
No EWC	0.854468	0.737730	0.658440	0.764539	0.588652	0.776651	0.540228
SI	**0.873475**	**0.795319**	**0.758545**	**0.829787**	0.687234	0.836582	0.701149
Rehearsal	0.831773	0.771064	0.748085	0.760284	**0.733333**	0.767707	0.728463

TABLE C.23: Class A Accuracy After Training on Class A, B, C, and D. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean After A	Mean After B	Mean After C	Mean After D	Max After D	Min After D	Upper Bound	Lower Bound
EWC	0.854468	0.733191	0.733759	**0.737021**	0.775887	0.670922	0.776288	0.657755
No EWC	0.854468	0.737730	0.658440	0.593475	0.680851	0.510638	0.691823	0.495127
SI	**0.873475**	**0.795319**	**0.758545**	0.709220	**0.822695**	0.628723	0.892184	0.626255
Rehearsal	0.831773	0.771064	0.748085	0.712908	0.744681	**0.696454**	0.747214	0.678601

TABLE C.24: Class B Accuracy After Training on Class B and After Class C. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean After B	Mean After C	Max After C	Min After C	Upper Bound	Lower Bound
EWC	0.908367	**0.891795**	0.907989	0.856950	0.892979	0.850611
No EWC	**0.932389**	0.807827	**0.910931**	0.678812	0.942657	0.672998
SI	0.924696	0.851282	0.893387	0.782726	0.930717	0.771847
Rehearsal	0.929285	0.856815	0.905533	0.716599	0.959446	0.754184

TABLE C.25: Class B Accuracy After Training on Class B, C, and D. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean After B	Mean After C	Mean After D	Max After D	Min After D	Upper Bound	Lower Bound
EWC	0.908367	**0.891795**	**0.819190**	0.911552	0.654588	0.891046	0.647335
No EWC	**0.932389**	0.807827	0.643455	0.832659	0.479082	0.853187	0.433723
SI	0.924696	0.851282	0.748178	0.800270	**0.700405**	0.817510	0.678847
Rehearsal	0.929285	0.856815	0.702429	**0.870445**	0.672645	0.782237	0.652622

TABLE C.26: Class C Accuracy After Training on Class C and After Class D. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean After C	Mean After D	Max After D	Min After D	Upper Bound	Lower Bound
EWC	0.905526	0.864151	0.881402	0.832884	0.891501	0.836801
No EWC	**0.940296**	0.861456	0.911051	0.742588	0.962932	0.759979
SI	0.922911	**0.880323**	**0.920485**	**0.842318**	0.941132	0.819514
Rehearsal	0.934232	0.824798	0.880485	0.777628	0.906442	0.803154

TABLE C.27: Final Accuracy for Class D. Upper and lower bound values represent mean ± 2 standard deviations, respectively.

Method	Mean Final Acc	Max Final Acc	Min Final Acc	Upper Bound	Lower Bound
EWC	0.744270	0.756672	0.726845	0.763391	0.725149
No EWC	**0.892057**	0.895400	0.857143	0.891219	0.852894
SI	0.815385	0.830455	0.803768	0.834286	0.796484
Rehearsal	0.886342	**0.896389**	**0.877551**	0.898995	0.873690