Journal of High School Science

Navigating parameter space: mitigating catastrophic forgetting in continual learning

Huang H

Submitted: June 5, 2026, Revised: version 1, July 13, 2026, version 2, July 24, 2026
Accepted: July 25, 2026

Abstract
Artificial general intelligence requires neural networks to continually acquire new knowledge while preserving previously learned information. However, sequential learning causes catastrophic forgetting, whereby performance on earlier tasks deteriorates as new tasks are learned. Parameter regularization methods, such as Elastic Weight Consolidation (EWC), attempt to address this challenge by constraining updates to parameters deemed important for previous tasks. This work investigates the influence of fully connected (FC) layer architecture on parameter regularization in the class incremental learning setting using a modified ResNet-18 trained on the CIFAR-10 dataset. Experiments demonstrate that a dynamically expanding FC architecture substantially outperforms a conventional static FC layer. To address the limited memory retention observed with EWC in static FC architectures, a new parameter regularization approach, termed Range Matrix, is proposed. Unlike EWC, which selectively constrains parameters identified by the Fisher Information Matrix, Range Matrix constrains all network parameters within dynamically computed bounds derived from loss gradients. Experimental results show that Range Matrix improves memory retention and reduces catastrophic forgetting relative to EWC when using a static FC architecture. Control experiments further demonstrate that the improved performance of Dynamic FC arises primarily from context partitioning rather than increased model capacity alone. Finally, analysis of the overlap between Fisher-important parameters and Range Matrix-constrained parameters shows that the two methods identify substantially different parameter subsets despite achieving similar performance under Dynamic FC, suggesting that FC architecture is the dominant factor governing continual learning performance. These findings provide both a novel parameter regularization strategy and new insights into the interaction between network architecture and catastrophic forgetting in continual learning.

Keywords
Continual learning, Catastrophic forgetting, Class-incremental learning, Elastic Weight Consolidation (EWC), Range Matrix, Parameter regularization, Dynamic Neural Networks, Context partitioning, ResNet, CIFAR-10

Henry Huang, Palos Verdes Peninsula High School, 27118 Silver Spur Rd, Rolling Hills Estates, CA 90274, USA.
henryhuang011@gmail.com

https://doi.org/10.64336/001c.165645

1. Introduction

Existing machine learning models have been shown to match or even surpass human-level performance on certain tasks. However, classical training paradigms generally rely on the assumption that the training dataset comprehensively represents all potential samples a model may encounter during inference. This assumption fails in dynamic real-world environments, where live data is constantly evolving. State-of-the-art models are typically trained on large, static datasets and subsequently frozen. When these models are updated with new data, their performance on previously learned data can degrade substantially. This phenomenon, known as catastrophic forgetting, presents a significant challenge to the development of robust and adaptable models (1).

Consequently, the field of continual learning (CL) has emerged to address this limitation. In CL, a model incrementally learns from a sequence of distinct tasks, referred to as contexts, rather than learning all tasks simultaneously (2). Effective CL can offer significant advantages over traditional training frameworks, including reduced computational cost and training time. In conventional settings, the arrival of new data typically necessitates retraining. For state-of-the-art models, this process can require months of time and millions of dollars in associated costs (3). Continual learning aims to reduce or eliminate the need for such repeated retraining.

Achieving effective CL requires balancing the trade-off between learning plasticity and memory stability. Plasticity enables a model to acquire new information, whereas stability helps preserve knowledge acquired from previous contexts. A variety of CL methods have been proposed to address this stability-plasticity trade-off, including replay-based methods (4), dynamic architectures (5), and parameter regularization techniques (6).

1.1 Summary

This work focuses specifically on parameter regularization. Elastic Weight Consolidation (EWC) (7), a pioneering parameter regularization method, is evaluated across different fully connected (FC) layer architectures. The results indicate that the structure of the FC layer significantly affects both overall accuracy and the effectiveness of hyperparameter optimization. Furthermore, EWC with a static FC layer is observed to exhibit high levels of catastrophic forgetting.

To further investigate these observations, a new parameter regularization approach, termed the Range Matrix, is proposed. The Range Matrix constrains each parameter to a specific range derived from loss gradients, introducing greater control and facilitating the identification of a parameter space that accommodates both current and previously learned contexts. The primary contributions of this work are summarized as follows:

[1] This work presents experiments demonstrating the significance of the final fully connected (FC) layer structure in continual learning performance. Specifically, a dynamically updated FC layer is shown to outperform a static FC layer.

[2] A parameter regularization approach, termed the Range Matrix, is proposed to investigate the limitations of EWC identified through experimentation. An implementation of the

Range Matrix exhibits greater resistance to catastrophic forgetting than EWC on Static FC architectures through improved memory stability. These findings provide insights and potential directions for future research into parameter regularization methods for continual learning.

2. Previous work

2.1 Continual learning scenarios

Previous work categorizes continual learning into three primary scenarios: Task-Incremental Learning (TIL), Domain-Incremental Learning (DIL), and Class-Incremental Learning (CIL) (8). As illustrated in Figure 1, in TIL, a model sequentially learns distinct contexts, with the context identity explicitly provided during both training and inference. DIL is characterized by contexts in which the model encounters different types of input data but must predict from the same set of possible labels. Because the complete label space is shared across contexts, the context identity is not required during inference.

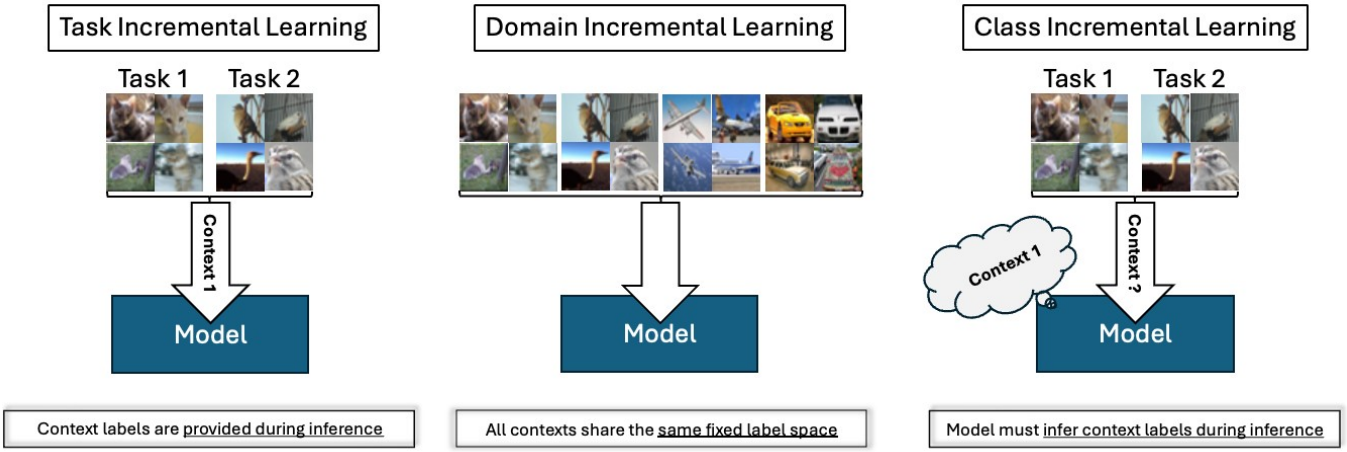


Figure 1. An overview of the 3 continual learning scenarios. Class Incremental Learning (CIL) requires models to infer context identity, making it the most realistic and complex of the three.

Finally, CIL represents the most challenging of the three scenarios. In CIL, the model is trained on a sequence of contexts, each introducing new classes. For example, a model may first learn to classify cats and dogs and later learn to classify cars and airplanes. During inference, however, the model is not told from which context a task comes from. It must therefore learn to distinguish among all classes it has thus far encountered. This requirement makes CIL particularly challenging and closely aligned with the dynamic nature of real-world environments (9).

2.2 Continual learning approaches

Prior approaches to mitigating catastrophic forgetting can be broadly categorized into three primary groups: replay-based methods, architectural methods, and parameter regularization methods. Replay-based methods store, generate, or otherwise retain information from previously encountered contexts and

incorporate this information during training on new contexts (10–12). This rehearsal mechanism helps preserve previously acquired knowledge while enabling the model to learn new information. However, their effectiveness is constrained by factors such as the amount of memory available for storing past data, the computational cost of replay, and, in some applications, privacy or data-retention restrictions. Generative replay methods can reduce the need to store raw data but introduce additional computational and modeling requirements.

Architectural approaches address catastrophic forgetting by modifying the model structure as new contexts are encountered. Expert Gate (13), for example, uses context-specific components in which distinct subnetworks or models are allocated to different contexts. Such methods can effectively reduce interference between contexts by isolating the parameters responsible for learning different information. However, these approaches often require the model to determine which context-specific components should process a given input. If this routing requires an explicitly provided context identity, the approach is not directly compatible with the standard CIL setting. Other methods dynamically expand the model architecture as new contexts or classes are learned, while employing techniques such as parameter freezing or pruning to preserve previously acquired knowledge (14, 15). Although these approaches can reduce interference between old and new knowledge, continual expansion can lead to increasing memory requirements, computational costs, and architectural complexity as the number of contexts increases.

Unlike replay and architectural methods, parameter regularization aims to preserve previously learned knowledge without storing past data or continually expanding the model architecture. A central challenge is identifying a parameter space that enables the model to learn new contexts while preserving performance on previously learned contexts. Several approaches have explored geometric solutions to this problem. Wołczyk et al. (16) introduced a Hyperrectangle training framework, which models the parameter space as a hyperrectangle fully contained in the hyperrectangles of previous tasks. This approach shows resistance to catastrophic forgetting, though is limited by an incompatibility with batch normalization. Teng et al. (17) proposed Direction-Constrained Optimization (DCO), modeling the high performing regions in parameter space as cones. Other regularization approaches also include trust region continual learning (18), which combines generative replay with an EWC-inspired trust constraint. This approach depends on the quality of the Fisher matrix. Similarly, Garg et al. (19) imposed Fisher-orthogonal constraints on all network parameters to mitigate forgetting. Elsayed et al. (20) introduced a weight-clipping approach to continual and reinforcement learning with the goal of addressing loss of plasticity and policy collapse due to increasing weight magnitudes. The clipping range is calculated once before training and remains fixed throughout. Range Matrix follows a similar clipping approach but differs from all the works discussed by imposing dynamically changing range values calculated through the loss gradient. Overall, the concept of parameter regularization is to constrain network parameters and limit changes to parameters deemed important for retaining

knowledge of previous tasks. The difference lies within the methods used to do so, and how they discover a shared parameter space that satisfies both the current and previous task simultaneously.

2.3 EWC parameter regularization

Elastic Weight Consolidation (EWC) is a pioneering parameter regularization method for continual learning. EWC modifies the training loss with the aim of helping the network converge on a parameter space that works for both current and previous tasks. The relative strength of this constraint is controlled by the hyperparameter lambda, which governs the trade-off between stability and plasticity.

The structure of the final fully connected (FC) layer is an important architectural consideration in parameter regularization. Hidden layers extract features from the input, while the final FC layer maps these features to the model's output classes. Consequently, the structure and training behavior of the FC layer can directly affect performance and the extent to which new learning interferes with previously acquired knowledge. Two FC layer configurations are considered in this work: static and dynamic. A static FC layer maintains a fixed number of neurons throughout continual learning. In contrast, a dynamic FC layer is incrementally expanded as new contexts are encountered. Additionally, only the newly added FC neurons are trained for the corresponding context, while previously learned neurons remain frozen (but a variation is also investigated in this study). This can reduce direct interference between parameters associated with different contexts and may therefore help preserve previously acquired knowledge.

The interaction between lambda and FC layer structure, particularly the difference between static and dynamically expanded FC layers, has not been extensively investigated. This work will study the effect of FC layer structure on EWC parameter regularization and present an approach to address its limitations with mitigating catastrophic forgetting in class incremental learning scenarios.

3. Methods

3.1 EWC implementation

EWC was re-created by calculating the fisher information matrix and loss equation following the implementation detailed in the original work as, $L = L_o + \sum_i \left(\frac{\lambda}{2} \cdot F_i \cdot \left(\theta_i - \theta_i^*\right)^2\right)$ (Eq.1). In the loss calculation for EWC shown in equation 1, L_o represents the loss value obtained from calculating cross entropy loss. This is added to the EWC regularization term, which consists of the squared difference between the current weight of the parameter θ_i and the previous weight θ_i^*. The difference is multiplied by the corresponding value in the calculated Fisher Information matrix and a hyperparameter λ. When a parameter is important, F_i is large. Combined with a large enough λ value, the regularization term causes the weight value to remain close to θ_i^*, thereby preserving memory of prior contexts. Thus, tuning the value of lambda can affect the plasticity-stability tradeoff in EWC.

3.2 Resnet architecture

Resnet (21) features a residual learning framework that allows for the training of deeper models. The model architecture was originally

designed for the ImageNet (22) dataset. For all experiments in this work, a modified Resnet-18 was trained on the CIFAR-10 dataset (23). To enable compatibility with CIFAR-10, modifications to the architecture were made (Figure 2),

[1] The Max Pool and Average Pool layers at the beginning and end of the model were removed to accommodate lower resolution CIFAR-10 images without losing features.

[2] The size of the final FC layer was decreased to 10 neurons, corresponding to the 10 classes in CIFAR-10.

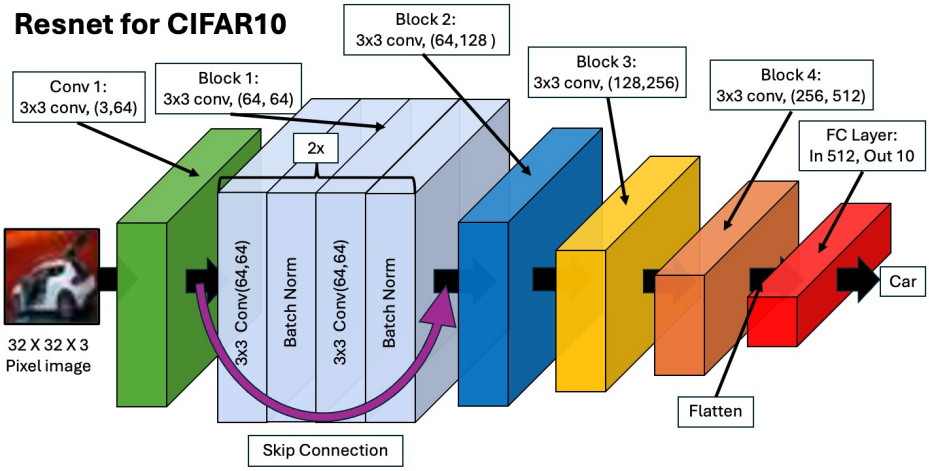


Figure 2. The modified Resnet-18 architecture, with pooling layers removed and a smaller final FC layer to accommodate the lower resolution images of CIFAR10.

In addition to altering the model architecture, the training framework must also be adapted. Continual learning format requires training data to be split into separate contexts, with each context having a set number of distinct tasks. During training, the model is allowed to train on each context once. The 10 categories in CIFAR-10 were split into 5 distinct contexts, each containing 2 tasks.

3.3 FC layer architecture

Within parameter regularization, the final fully connected (FC) layer can adopt a static or dynamic architecture (24). For the model used in experimentation, the static FC layer was implemented as a standard linear layer with input dimensions matching the flattened feature representations from the Resnet convolution blocks, and an out-feature size matching the total number of classes in the dataset (i.e. 10 for the CIFAR-10 dataset). Conversely, the dynamic FC layer is more complex and must be updated following each learned context. As illustrated in Figure 3, this was achieved by expanding the number of output neurons in the layer to accommodate new contexts while copying the weights and biases learned during prior ones. Furthermore, representations of previous contexts were consolidated by restricting parameter updates to the newly added neurons—for example, computed exclusively over the two active output neurons in a scenario

where a model learns CIFAR-10 data in increments of two tasks per context.

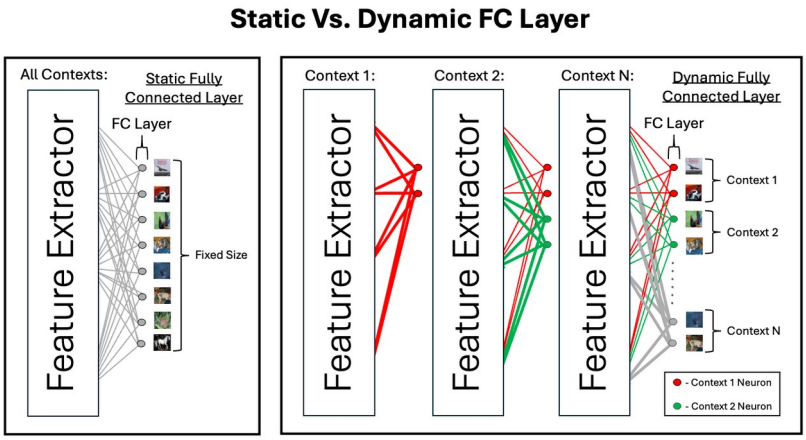


Figure 3. A visualization of Static FC (left) and Dynamic FC (right). Static FC remains the same size throughout training, while the size of Dynamic FC increases as new contexts are learned. This characteristic of Dynamic FC results in each group of neurons being trained for a specific context.

3.4 Range Matrix

To address the shortcomings in EWC, a new parameter regularization approach termed the Range Matrix was introduced. A key limitation of EWC stems from its parameter update mechanism. Specifically, EWC only limits parameters deemed "important" by the Fisher Information matrix. During training, EWC effectively anchors these important parameters near their original values, leaving the remaining parameters completely unconstrained. Changes to these unconstrained parameters improve performance on the current context but can inadvertently disrupt the performance on prior ones. The proposed Range Matrix method aimed to mitigate this issue by establishing permissible update ranges for *all* parameters as, $\Delta\theta_i = \frac{P \cdot L(\theta_i)}{\frac{dL}{d\theta_i}} = \frac{\Delta L}{\frac{dL}{d\theta_i}}$ (Eq.2). The formula for the range value of a parameter θ_i is shown in Equation 2. $L(\theta_i)$ represents the loss obtained from the loss function, and it is multiplied by a value P. This term represents ΔL, a measure of how much the loss is allowed to change. ΔL is divided by the gradient $\frac{dL}{d\theta_i}$ to obtain the range value of that parameter. A derivation of the formula is provided in Appendix 1. A tunable hyperparameter, P acted as the controller of plasticity. With a larger value of P, adaptability to new tasks is improved but forgetting may also increase. Conversely, a smaller value of P results in more memory stability but decreased plasticity when learning new tasks.

During the training phase, the model learned the first context without any Range Matrix constraint. Upon completing parameter optimization, the corresponding range matrix was calculated using Equation 2 (Appendix 4). During subsequent training on the second

context, a custom optimizer (Appendix 2) enforced the constraints from the calculated range matrix. Specifically, if a parameter update exceeded its allowed bounds as defined by its entry in the Range Matrix, the optimizer clipped the update at that maximum boundary. Following this bounded update, a new range matrix was computed for the second context. Finally, this new matrix was aggregated with the prior one by selecting the most restrictive bound between the two matrices for each parameter (Appendix 3). This training process was performed for all subsequent contexts.

A potential limitation of this approach stems from the model's plasticity. By imposing bounds across the entire parameter space, the Range Matrix progressively reduces the space in which parameters can be updated as additional contexts are learned. Consequently, the feasible parameter space may eventually become too constrained to support meaningful learning. Furthermore, if the optimal parameter space for a new context lies outside the bounds imposed by the Range Matrix, learning that context will be impaired. This limitation was exacerbated by the current aggregation strategy, in which the most restrictive bound was selected for each parameter when combining successive Range Matrices. As a result, the feasible parameter space could only remain unchanged or decrease over time, potentially leading to cumulative plasticity loss. Addressing this long-term reduction in plasticity remains an open problem and will require investigation of alternative Range Matrix aggregation strategies in future work.

3.5 Evaluation metrics

As mentioned previously, the Lambda hyperparameter was a critical component of the EWC calculation, dictating the relative importance of retaining prior task knowledge versus acquiring new information. Because variations in lambda have the potential to significantly impact accuracy, performing a hyperparameter search across a range of values allowed an optimal setting to be determined.

To improve repeatability during experimentation, all weights were initialized using the same seed. The performance of EWC and Range Matrix were evaluated utilizing two metrics. The first was overall accuracy for Static and Dynamic FC, which was recorded after training on each context. Overall accuracy was calculated by computing the mean accuracy of the model on the test data for all contexts it had been exposed to.

The second metric was accuracy distribution. Accuracy distribution was calculated by evaluating the model on every context it had been exposed to individually. Similarly to overall accuracy, this metric was calculated after each context was learned.

Additional metrics were recorded to quantify both learning plasticity and memory stability. Specifically, Backward Transfer (BWT) (25) was computed as the average difference between the final accuracy on each learned context and the accuracy immediately after that context was learned, measuring the extent to which subsequent learning affected prior knowledge. A negative BWT was indicative of catastrophic forgetting, whereas a positive BWT meant that learning subsequent contexts improved performance on previously learned contexts. Average Forgetting (AF) (26) and the

Retention Ratio quantified performance degradation relative to each context's peak accuracy achieved during continual learning. Higher AF values indicated greater catastrophic forgetting, whereas Retention Ratio values close to 1 indicated better preservation of knowledge. Plasticity was quantified using New-Task Accuracy, defined as the accuracy on each context immediately after the model had been trained on that context. Lower average New-Task Accuracy, or a progressive decline in New-Task Accuracy across contexts, could be indicative of reduced plasticity and a diminished capacity to efficiently acquire new knowledge.

4. Results

4.1 Baseline

To evaluate model performance, three baselines were established as shown in Figure 4. The first served as an upper bound: a network trained using standard offline frameworks, wherein it is exposed to the entire training dataset simultaneously. This network achieved optimal performance because it has concurrent exposure to all potential tasks it may encounter. This upper baseline confirmed that the model architecture possessed sufficient capacity to learn these tasks, thereby establishing a target performance for the continual learning methods. Conversely, two lower bound baselines were defined: one utilizing a Dynamic FC layer and the other a Static FC layer. These networks were trained on a continuous stream of contexts—simulating a dynamic real-world environment where all data is not available at once—but lacked any continual learning mechanisms to mitigate catastrophic forgetting. As a result, the accuracy decreased as the model was exposed to more contexts. These lower baselines served to quantify the impact of catastrophic forgetting experienced by current models in continual learning scenarios.

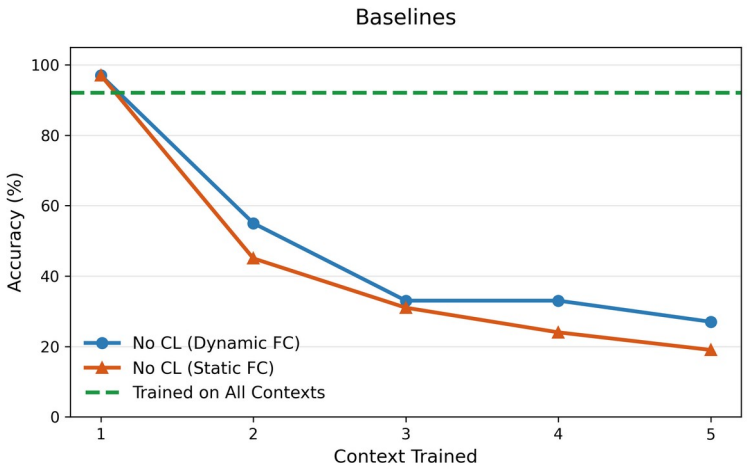


Figure 4. A graph showing the three baselines. The upper baseline (green) shows parameter regularization's potential, while the lower baselines (blue and orange for Dynamic and Static FC, respectively) highlight existing performance without continual learning.

4.2 Hyperparameter search results

A hyperparameter search was performed to identify an optimal lambda value for EWC. The results illustrated a dependency on the FC layer structure. For Dynamic FC, Figure 5 shows a distinct, optimal range of lambda values that maximized accuracy. Conversely, Static FC showed invariance to changes in lambda. This behavior could be explained by the difference in structures between the two FC layers. Lambda dictates the importance of remembering the previous context over learning the new one. This was effective when using Dynamic FC, where neurons are inherently grouped by context. Because the Static FC architecture lacked this context-specific grouping, the network did not have meaningful memory stability, and thus changes in lambda had negligible effect.

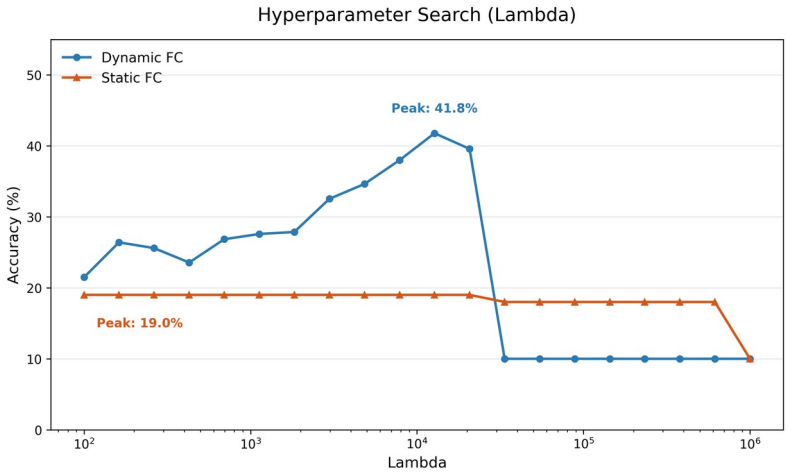


Figure 5. Lambda hyperparameter search for EWC. An optimal value can be determined for Dynamic FC (blue). However, Static FC (orange) shows fixed performance regardless of lambda, showing that EWC does not mitigate catastrophic forgetting at all.

At low values of lambda (approaching 0), the regularization term in Equation 1 becomes negligible, causing Dynamic FC to degrade to the low baselines, such as to those presented in Figure 4. In contrast, high lambda values inflate the regularization term, leading to gradient explosion and a severe drop in accuracy, as shown in Figure 5. Therefore, to achieve the best level of performance, a balanced lambda value must be chosen.

A hyperparameter search was also conducted to optimize the P hyperparameter introduced in the Range Matrix. P represents the maximum permissible deviation of the loss value from its original state. Range Matrix with Dynamic FC performs similarly to EWC with Dynamic FC. As shown in Figure 5, EWC with a Static FC layer lacked hyperparameter sensitivity, maintaining constant performance regardless of lambda. However, Range Matrix with Static FC was able to be optimized by tuning P (Figure 6). This suggests that the Range Matrix allows Static FC architectures to maintain memory stability.

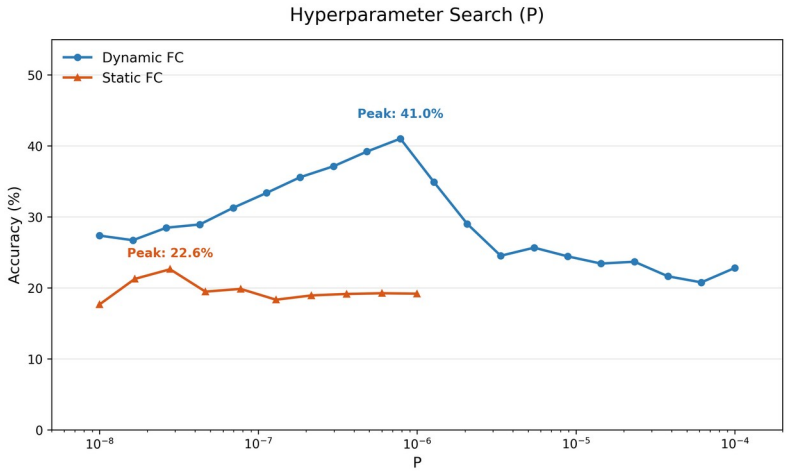


Figure 6. P hyperparameter search for Range Matrix. Range Matrix with Static FC is sensitive to hyperparameter change, unlike EWC with Static FC. This result demonstrates that Range Matrix can provide Static FC architectures with memory stability.

At low values of P, the boundaries calculated by the Range Matrix formula (Equation 2) were overly restrictive, resulting in a loss of learning plasticity. At high values of *P*, the calculated ranges became so broad that they ceased to meaningfully constrain parameter updates, increasing plasticity but causing insufficient memory stability. Therefore, P can be characterized as dictating the balance between stability and plasticity. Different values of P affect how much memory the model retains from previous tasks and how capable it will be at learning new ones.

4.3 EWC and Range Matrix accuracy results

Overall accuracy was evaluated for EWC

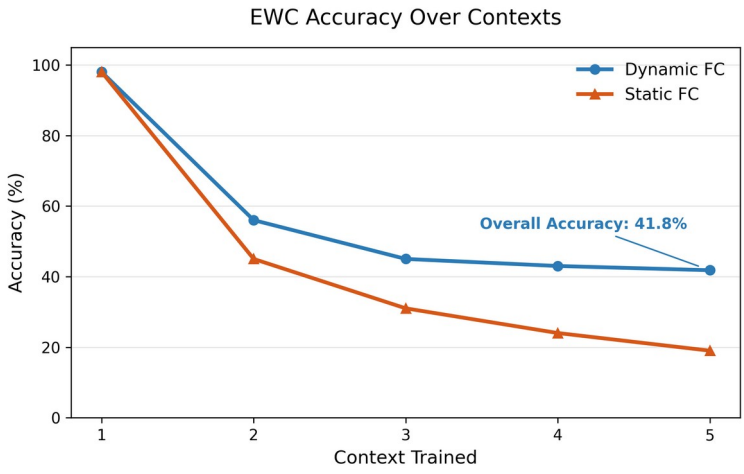


Figure 7. EWC overall accuracy graph with optimized lambda value. Dynamic FC improves from the baseline in Figure 4, while Static FC suffers from catastrophic forgetting and achieves the same performance as the baseline.

utilizing the optimized lambda determined from the hyperparameter search. As shown in Figure 7, the performance of Dynamic FC improved over the baseline value shown in Figure 4, achieving a final accuracy of 41.8%. In contrast, the Static FC architecture remained unaffected, yielding a final accuracy less than 20%. This result shows that EWC parameter regularization has no effect on Static FC for CIL scenarios, making it equivalent to the low baseline trained without a continual learning method (Figure 4).

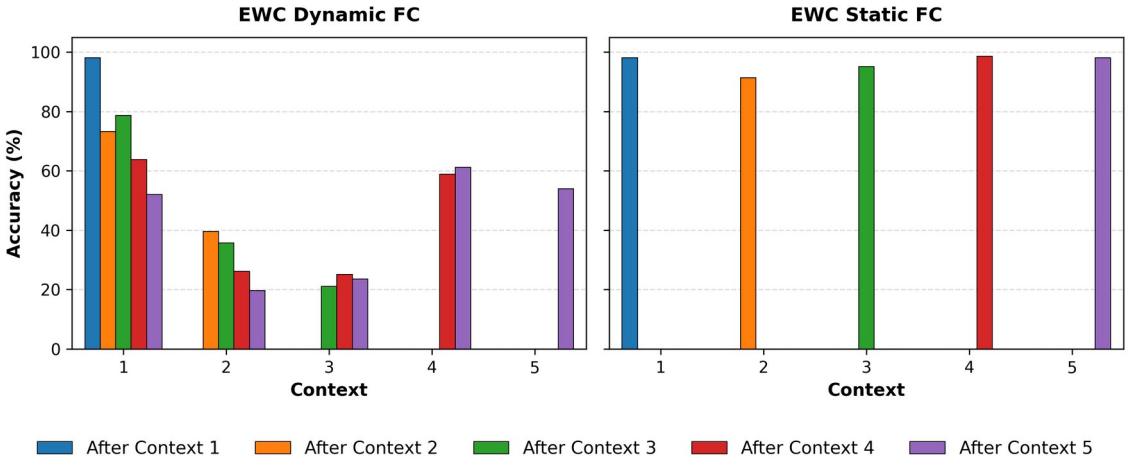


Figure 8. Accuracy distribution for EWC with Dynamic and Static FC. Dynamic FC shows residual memory of previous contexts. Conversely, Static FC suffers from catastrophic forgetting, only performing well on the currently learned context.

Figure 8 illustrates the accuracy distribution across all learned contexts for EWC using Static and Dynamic FC architectures. Each color represents the model's accuracy on every context after training on a particular context. For example, after completing training on Context 2, the Dynamic FC model achieved approximately 40% accuracy on Context 2 while retaining approximately 70% accuracy on the previously learned Context 1, indicating that a substantial portion of the earlier knowledge was preserved. As expected, accuracy on Contexts 3–5 remained at 0% because those contexts had not yet been encountered. In contrast, the Static FC model achieved high accuracy only on the current context and exhibited nearly 0% accuracy on previously learned contexts, indicating that earlier representations had been almost completely overwritten. This pattern persisted throughout continual learning: Dynamic FC consistently retained measurable performance on previously learned contexts, whereas Static FC retained little or no performance once a new context was learned. These results demonstrate that Dynamic FC substantially mitigates catastrophic forgetting by preserving knowledge acquired from earlier contexts, whereas Static FC exhibits severe catastrophic forgetting.

Figure 8 hence isolates the effect of FC architecture by holding the continual learning

algorithm constant (EWC). The results demonstrate that replacing the Static FC architecture with Dynamic FC substantially improves retention of previously learned contexts, indicating that FC architecture itself plays a significant role in mitigating catastrophic forgetting. Having established the influence of FC architecture, the following experiments examine whether the proposed Range Matrix can similarly improve memory retention when the FC architecture is held constant.

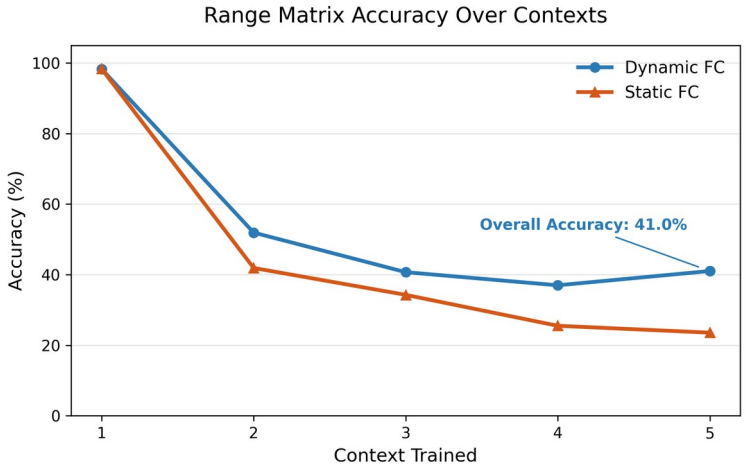


Figure 9. Range Matrix overall accuracy graph with optimized P value. Range Matrix with Static FC improves over EWC with Static FC.

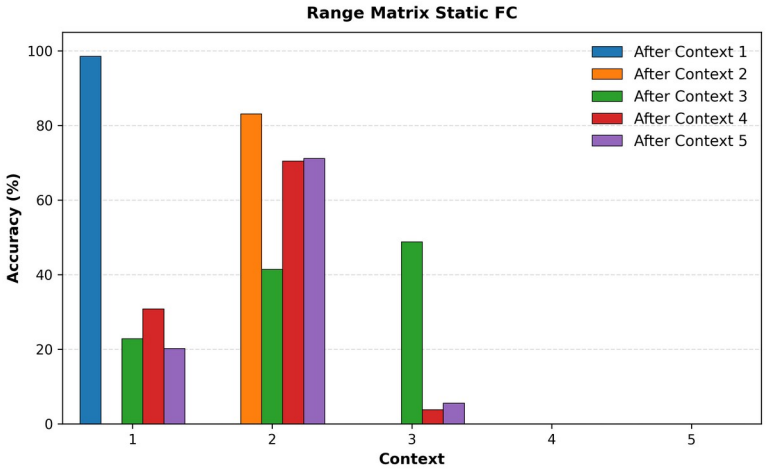


Figure 10. Accuracy distribution graph for Range Matrix with Static FC. Range Matrix with Static FC shows residual memory of previous contexts, demonstrating some memory stability.

Figure 9 shows that the overall accuracy of Range Matrix with Dynamic FC was 41%, which is comparable to the performance achieved by EWC with Dynamic FC (Figure 7). This indicates that replacing EWC with the proposed Range Matrix does not substantially alter performance when context partitioning is already provided by the Dynamic FC architecture. To further investigate the behaviour of Range Matrix under Static FC architectures, where hyperparameter sensitivity was observed during the hyperparameter search, an accuracy distribution was calculated. As shown in Figure 10, Range Matrix with Static FC retains measurable accuracy on previously learned contexts, demonstrating that earlier knowledge is partially preserved. In contrast, EWC with Static FC (Figure 8) exhibits near-complete catastrophic forgetting, with previously learned contexts reduced to almost zero accuracy. These results indicate that the Range Matrix provides greater memory stability than EWC when applied to a Static FC architecture.

To compare overall performance, Range Matrix and EWC were evaluated using Static FC architectures over five random seeds (listed in Appendix 5). As shown in Table 1, Range Matrix achieved an average accuracy of 22.81% (standard deviation 1.0434), compared with 19.53% (standard deviation 0.0327) for EWC. Although this improvement in overall accuracy is modest, it is accompanied by substantially improved retention of previously learned contexts, as demonstrated in Figure 10. Nevertheless, the overall accuracy remains considerably lower than that achieved by Dynamic FC architectures, indicating that improved parameter regularization alone cannot fully compensate for the limitations of a Static FC architecture. These findings suggest that while the proposed Range Matrix partially mitigates catastrophic forgetting in Static FC networks, context partitioning provided by the Dynamic FC architecture remains the dominant contributor to continual learning performance.

Table 1. EWC and Range Matrix Static FC overall accuracy comparison over 5 runs with different seeds.

	Avg.	**Std.**	**Min.**	**Max.**
EWC Static FC	19.53 %	0.0327	19.49 %	19.58 %
Range Matrix Static FC	22.81 %	1.0434	21.11 %	23.66 %

4.4 Stability-plasticity comparison

To understand the stability-plasticity tradeoff, four continual learning metrics were reported: Average New-Task Accuracy, Backwards Transfer, Average Forgetting, and the Retention Ratio. To quantitatively evaluate the stability-plasticity trade-off, Table 2 reports Average New-Task Accuracy, Backward Transfer (BWT), Average Forgetting, and Retention Ratio for each continual learning configuration. These metrics complement the accuracy distribution plots by directly measuring the acquisition of new knowledge and the preservation of previously learned information.

Table 2. Plasticity and stability metrics for each of the experiments.

	Avg. New-Task Accuracy	BWT	Avg. Forgetting	Retention Ratio
EWC Static FC	96.27%	-95.80%	95.80%	0.00
EWC Dynamic FC	54.34%	-15.34%	16.93%	0.74
Range Matrix Static FC	46.06%	-33.38%	33.38%	0.29
Range Matrix Dynamic FC	56.15%	-18.93%	18.93%	0.67

EWC with a Static FC architecture exhibits the highest Average New-Task Accuracy (96.27%), indicating that the model readily learns each new context immediately after training. However, this high plasticity comes at the expense of memory stability. The BWT of -95.80%, Average Forgetting of 95.80%, and Retention Ratio of 0.00 demonstrate that learning subsequent contexts almost completely destroys previously acquired knowledge. These quantitative results are consistent with the accuracy distributions shown in Figure 8, confirming that EWC with a Static FC architecture experiences severe catastrophic forgetting.

In contrast, Range Matrix substantially improves memory retention for Static FC architectures. Average Forgetting decreases from 95.80% to 33.38%, while the Retention Ratio increases from 0.00 to 0.29, indicating that a significant proportion of previously learned knowledge is preserved. Although Average New-Task Accuracy decreases to 46.06%, this reduction reflects the intended trade-off between plasticity and stability produced by constraining all network parameters rather than only those identified as important by the Fisher Information Matrix. These results demonstrate that Range Matrix sacrifices some immediate adaptation in exchange for markedly improved long-term memory retention.

Both Dynamic FC architectures exhibit substantially better stability than their Static FC counterparts. EWC Dynamic FC achieves an Average Forgetting of 16.93% and a Retention Ratio of 0.74, while Range Matrix Dynamic FC reports similar values of 18.93% and 0.67, respectively. Likewise, both methods produce relatively small negative BWT values compared with Static FC, indicating that subsequent learning has a much smaller detrimental effect on previously learned contexts. The similar performance of EWC and Range Matrix under Dynamic FC suggests that the architectural separation of task representations largely mitigates catastrophic forgetting, reducing the relative influence of the underlying parameter regularization strategy.

Overall, Table 2 demonstrates two important findings. First, the proposed Range Matrix significantly improved memory stability over EWC when using Static FC architectures, validating the proposed parameter regularization approach. Second, Dynamic FC architectures consistently outperformed Static FC architectures regardless of the continual learning algorithm employed, supporting the conclusion that context partitioning within the FC layer is a dominant factor governing catastrophic forgetting.

5. A Study of FC Architectures

Dynamic FC consistently achieved higher continual-learning performance than Static FC. A key structural difference between these architectures lies in how task representations are stored within the final fully connected (FC) layer. In a Static FC architecture, all contexts share the same fixed set of output neurons. Consequently, learning new contexts requires repeatedly updating the same parameters, increasing the likelihood that optimization for the current context will interfere with representations learned for previous contexts. In contrast, the Dynamic FC architecture incrementally expands the FC layer as new contexts are introduced. Newly added neurons are dedicated to the current context, while neurons associated with previously learned contexts remain fixed. This architectural partitioning reduces direct interference between task-specific representations and is hypothesized to be a primary mechanism underlying the improved memory retention observed in Dynamic FC.

5.1 Context partitioning versus architectural expansion

To evaluate this hypothesis, a control experiment was designed to distinguish the effects of architectural expansion from those of context partitioning. A Dynamic FC architecture was used in which the FC layer continued to expand after each context, thereby preserving the increase in model capacity. However, instead of restricting optimization to the newly added neurons, the entire FC layer remained trainable throughout continual learning. This modification removes context partitioning while maintaining the same architectural expansion, allowing the effect of increased parameter count to be isolated from the effect of task-specific parameter partitioning.

Table 3. Continual learning metrics for the Dynamic FC control experiment. The continual learning metrics for EWC and Range Matrix with no CP (Context Partitioning) are reported.

	Avg. New-Task Accuracy	BWT	Avg. Forgetting	Retention Ratio
EWC no CP	96.31%	-95.91%	95.91%	0.00
Range Matrix no CP	94.88%	-94.45%	94.45%	0.00

Table 3 reports the continual learning metrics for the Dynamic FC control experiment in which the FC layer continues to expand after each context, but all FC neurons remain trainable, thereby removing context partitioning while preserving the increase in model capacity.

Both methods exhibit severe catastrophic forgetting. EWC without context partitioning achieves an Average New-Task Accuracy of 96.31%, indicating that the model readily learns each new context immediately after training. However, this high plasticity is accompanied by a BWT of -95.91%, an Average Forgetting of 95.91%, and a Retention Ratio of 0.00, demonstrating that previously acquired knowledge is almost completely lost after subsequent training. Similarly, Range Matrix without context partitioning achieves an Average New-Task Accuracy of 94.88%, but produces a BWT of -94.45%, an Average Forgetting of 94.45%, and a Retention Ratio of

0.00. Although Range Matrix exhibits slightly less forgetting than EWC, both methods retain virtually none of the knowledge acquired from earlier contexts.

The results are comparable to those obtained using Static FC architectures, where catastrophic forgetting is likewise observed. In contrast, both Dynamic FC configurations employing context partitioning (Table 2) achieve substantially lower forgetting and markedly higher retention ratios. Because the control experiment preserves the increase in model capacity while removing only context partitioning, the re-emergence of catastrophic forgetting indicates that architectural expansion alone is insufficient to improve continual learning performance. Instead, the results demonstrate that partitioning task-specific representations within the FC layer is the dominant factor associated with the improved memory retention observed in Dynamic FC architectures.

5.2 FC architecture vs. continual learning method

As shown in Section 4.3, EWC and Range Matrix achieve comparable performance when combined with the Dynamic FC architecture. To investigate whether this similarity arises because both methods preserve the same subset of network parameters, the overlap between the parameters identified by each method was quantified. Specifically, after each learning context, the Top-K parameters with the largest Fisher Information values (EWC) and the Top-K parameters with the most restrictive Range Matrix bounds were identified. The Top-K Overlap was then computed as the proportion of parameters common to both sets. A high overlap would indicate that both methods preserved largely the same parameters despite employing different regularization strategies, whereas a low overlap would suggest that they identified different subsets of parameters while achieving similar continual learning performance.

Table 4: Top-K overlap for Fisher and Range Matrix importance estimates.

	Context 1	**Context 2**	**Context 3**	**Context 4**	**Context 5**
Top 1% overlap	61.82%	23.72%	23.39%	25.20%	17.51%
Top 5% overlap	71.87%	34.91%	33.09%	32.18%	26.15%
Top 10% overlap	73.09%	42.26%	40.30%	38.41%	33.23%
Top 25% overlap	77.66%	57.91%	54.96%	53.42%	49.06%

Table 4 reports the Top-K overlap between the parameters identified as most important by the Fisher Information Matrix (EWC) and those assigned the most restrictive bounds by the Range Matrix after each learning context. Four Top-K thresholds (1%, 5%, 10%, and 25%) are considered to evaluate the consistency of the two importance estimates across increasingly larger subsets of network parameters.

Following training on the first context, both methods exhibit relatively high agreement. Depending on the threshold, the overlap ranges from 61.82% (Top 1%) to 77.66% (Top 25%), indicating that both approaches initially identify

many of the same parameters as being important for preserving learned knowledge.

However, as additional contexts are learned, the overlap decreases substantially across all Top-K thresholds. The most pronounced reduction is observed for the Top 1% parameters, where the overlap decreases from 61.82% after the first context to 17.51% after the fifth context. Similar trends are observed for the Top 5% (71.87% to 26.15%), Top 10% (73.09% to 33.23%), and Top 25% (77.66% to 49.06%) subsets. Although larger Top-K thresholds consistently exhibit greater overlap than smaller thresholds, all thresholds demonstrate a progressive decline as continual learning proceeds.

The reduction in overlap indicates that EWC and Range Matrix increasingly prioritize different subsets of parameters as successive contexts are incorporated. The decline is particularly pronounced for the highest-ranked parameters, suggesting that the two methods diverge most strongly in identifying the parameters considered most critical for knowledge preservation. In contrast, the larger Top-K subsets retain moderate overlap throughout training, indicating that broader regions of parameter importance remain partially shared despite differences in the highest-ranked parameters.

Despite these increasingly different parameter importance estimates, Section 4.4 shows that EWC with Dynamic FC and Range Matrix with Dynamic FC achieve comparable continual learning performance. Together, these results indicate that similar performance can be obtained even when the two methods preserve substantially different subsets of parameters. This observation is consistent with the results of the Dynamic FC control experiment, which demonstrated that removing context partitioning causes catastrophic forgetting to re-emerge despite preserving architectural expansion. Collectively, these findings suggest that the context-partitioned Dynamic FC architecture exerts a stronger influence on continual learning performance than the specific parameter importance estimates used by either regularization method.

6. Limitations and perspective

While the proposed Range Matrix demonstrates improved memory retention over EWC for Static FC architectures, several limitations remain. First, experiments were conducted using a modified ResNet-18 on the CIFAR-10 dataset under the Class Incremental Learning (CIL) setting. Although CIL is among the most challenging continual learning scenarios, additional evaluation on larger-scale benchmarks and alternative architectures is necessary to determine the generality of the findings.

Second, the current implementation of the Range Matrix progressively restricts the feasible parameter space by retaining the most restrictive parameter bounds across contexts. While this strategy improves memory stability, it may reduce plasticity as the number of learned contexts increases. Future work should investigate alternative aggregation strategies that dynamically relax or adapt parameter bounds while maintaining previously acquired knowledge.

Third, although this work demonstrates that context partitioning, rather than architectural

expansion alone, is the principal contributor to the improved performance of Dynamic FC architectures, the precise mechanisms by which partitioned representations reduce interference remain incompletely understood. Further studies examining feature representations, gradient interactions, and layer-wise parameter evolution may provide deeper theoretical insight into why context partitioning is so effective.

Finally, the Top-K overlap analysis demonstrated that EWC and Range Matrix identify substantially different subsets of important parameters despite producing similar performance under Dynamic FC architectures. This observation suggests that architectural design may dominate the behaviour of parameter regularization methods in certain continual learning settings. Future work should investigate whether architecture-aware regularization methods can jointly optimize network structure and parameter constraints, potentially achieving greater memory retention without sacrificing learning plasticity.

7. Conclusion

Catastrophic forgetting remains one of the principal challenges preventing neural networks from learning continuously in dynamic environments. This work investigated parameter regularization methods for Class Incremental Learning by examining both the behaviour of Elastic Weight Consolidation (EWC) and a newly proposed parameter regularization approach, Range Matrix, using modified ResNet-18 architectures on the CIFAR-10 dataset.

The experiments demonstrate that the structure of the final fully connected (FC) layer is a critical determinant of continual learning performance. Dynamic FC architectures consistently outperformed static FC architectures, while Range Matrix provided substantially improved memory retention over EWC when using static FC layers. Unlike EWC, which selectively constrains parameters identified by the Fisher Information Matrix, Range Matrix constrains all network parameters within dynamically computed bounds, providing improved resistance to catastrophic forgetting under static architectures.

Beyond introducing a new parameter regularization strategy, this work provides new insight into the mechanisms governing continual learning performance. A control experiment showed that the superior performance of Dynamic FC does not arise simply from increasing model capacity. Rather, it is the partitioning of task representations into separate output neurons that largely mitigates catastrophic forgetting. Furthermore, comparison of Fisher-important parameters with Range Matrix-constrained parameters demonstrated that the two methods identify substantially different subsets of parameters despite exhibiting similar performance under Dynamic FC architectures. These findings suggest that network architecture can exert a greater influence on continual learning performance than the choice of parameter regularization algorithm alone.

The proposed Range Matrix therefore contributes both a novel parameter regularization framework and a broader understanding of the interaction between parameter constraints and architectural design in continual learning. Future work should

investigate alternative strategies for aggregating Range Matrices to preserve long-term plasticity, evaluate the method on larger continual learning benchmarks, and explore whether architectural partitioning and parameter regularization can be jointly optimized to further reduce catastrophic forgetting.

Acknowledgements

I would like to thank my high school science research teacher, Mark Greenberg, for his guidance and feedback throughout the process of this project. I would also like to thank Biqin Huang for his help in building the computer used to run this experiment as well as for his feedback on the project.

8. References

1. Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., & Zhang, Y. (2025). An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 33*, 3776–3786. https://doi.org/10.1109/TASLPRO.2025.3606231

2. Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(8), 5362–5383. https://doi.org/10.1109/TPAMI.2024.3367329

3. Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., Walsh, T., Hamrah, A., Santarlasci, L., Lotufo, J. B., Rome, A., Shi, A., & Oak, S. (2025). *Artificial Intelligence Index Report 2025*. Stanford University, Institute for Human-Centered Artificial Intelligence (HAI), AI Index Steering Committee. https://hai.stanford.edu/ai-index/2025-ai-index-report

4. Bagus, B., & Gepperth, A. (2021). An investigation of replay-based approaches for continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–9). IEEE. https://doi.org/10.1109/IJCNN52387.2021.9533862

5. Li, H., Lin, S., Duan, L., Liang, Y., & Shroff, N. B. (2025). Theory on mixture-of-experts in continual learning. In *The Thirteenth International Conference on Learning Representations (ICLR)*. OpenReview. https://openreview.net/forum?id=7XgKAabsPp

6. Awasthi, A., & Sarawagi, S. (2019). Continual learning with neural networks: A review. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CODS-COMAD '19)* (pp. 362–365). Association for Computing Machinery. https://doi.org/10.1145/3297001.3297062

7. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

8. van de Ven, G. M., Tuytelaars, T., & Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence, 4*(12), 1185–1197. https://doi.org/10.1038/s42256-022-00568-3

9. Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., & Liu, Z. (2024). Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 46*(12), 9851–9873. https://doi.org/10.1109/TPAMI.2024.3429383

10. van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications, 11*(1), Article 4069. https://doi.org/10.1038/s41467-020-17866-2

11. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., & Wayne, G. (2019). Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf

12. Chaudhry, A., Ranzato, M'A., Rohrbach, M., & Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations (ICLR)*. OpenReview. https://openreview.net/forum?id=Hkf2_sC5FX

13. Aljundi, R., Chakravarty, P., & Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3366–3375). IEEE. https://doi.org/10.1109/CVPR.2017.753

14. Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). *Progressive neural networks* (arXiv:1606.04671). arXiv. https://doi.org/10.48550/arXiv.1606.04671

15. Yoon, J., Yang, E., Lee, J., & Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations (ICLR)*. OpenReview. https://openreview.net/forum?id=Sk7KsfW0-

16. Wołczyk, M., Postels, J., Yu, T., Van Gool, L., Tombari, F., & Yu, F. (2022). Continual learning with guarantees via weight interval constraints. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th International Conference on Machine Learning* (Vol. 162, pp. 23897–23911). *Proceedings of Machine Learning Research*. https://proceedings.mlr.press/v162/wolczyk22a.html

17. Teng, Y., Choromanska, A., Campbell, M., Lu, S., Ram, P., & Horesh, L. (2022). *Overcoming catastrophic forgetting via direction-constrained optimization* (arXiv:2011.12581). arXiv. https://doi.org/10.48550/arXiv.2011.12581

18. Wang, Z., Gupta, A., & MacLellan, C. J. (2026). *Trust region continual learning as an implicit meta-learner* (arXiv:2602.02417). arXiv. https://doi.org/10.48550/arXiv.2602.02417

19. Garg, I., Kolhe, N., Peng, A., & Gopalam, R. (2026). *Fisher-orthogonal projected natural gradient descent for continual learning* (arXiv:2601.12816). arXiv. https://doi.org/10.48550/arXiv.2601.12816

20. Elsayed, M., Lan, Q., Lyle, C., & Mahmood, A. R. (2024). Weight clipping for deep continual and reinforcement learning. *Reinforcement Learning Journal, 5*, 2198–2217.

21. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

22. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE. https://doi.org/10.1109/CVPR.2009.5206848

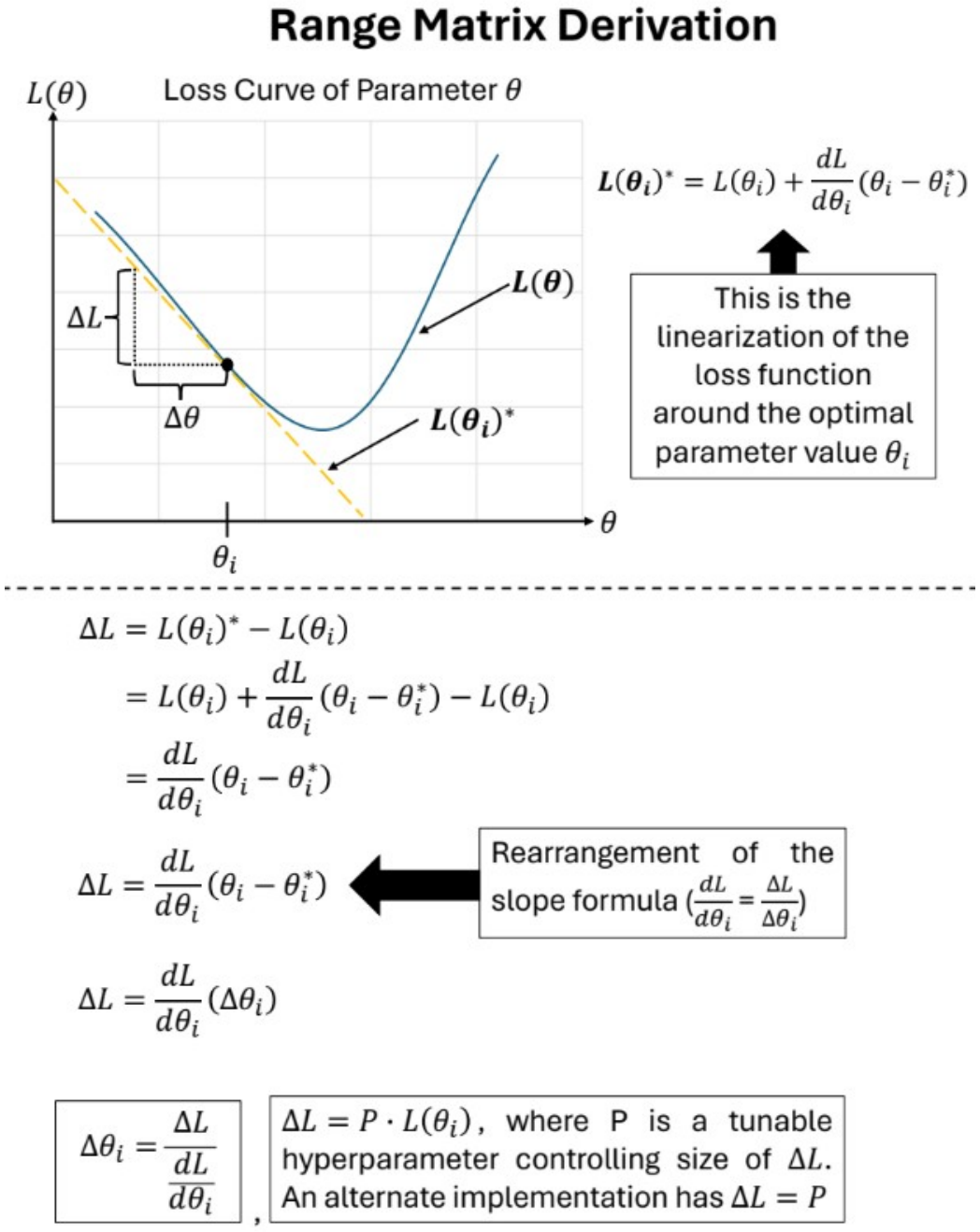
23. Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (Technical Report). University of Toronto. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

24. Zhou, D.-W., Wang, F.-Y., Ye, H.-J., & Zhan, D.-C. (2023). PyCIL: A Python toolbox for class-incremental learning. *Science China Information Sciences, 66*(9), 197101. https://doi.org/10.1007/s11432-022-3600-y

25. Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30, pp. 6467–6476). Curran Associates, Inc.

26. Chaudhry, A., Dokania, P. K., Ajanthan, T., & Torr, P. H. S. (2018). *Riemannian walk for incremental learning: Understanding forgetting and intransigence*. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 532–547)

Appendix 1. Range Matrix derivation



Appendix 2. Custom Range Matrix optimizer implementation

Algorithm 2 Custom optimizer for Range Matrix

Given parameter groups $\boldsymbol{G}$, previous parameters $\boldsymbol{P^*}$, learning rate $\boldsymbol{L}$, and Range Matrix $\boldsymbol{R}$, optimize the parameter P

```
procedure Optimize(G, R, P*, L)
    for group in G do
        for parameter P in group do
            if P_gradient is none then
                continue
            end if

            g ← P_gradient
            WeightUpdate ← −g · L
            P ← WeightUpdate + P

            if R is not empty then
                M ← R[group.name]
                Prev ← P*[group.name]
                Diff ← P − Prev

                U ← The indices where Diff
                    is greater than M

                Δθ_upper ← M[U]

                P[U] ← P*[U] + Δθ_upper

                V ← The indices where Diff
                    is less than R

                Δθ_lower ← M[V]

                P[V] ← P*[V] − Δθ_lower
            end if
        end for
    end for
end procedure
```

Appendix 3. Range Matrix update implementation

Algorithm 3 Updating the Range Matrix between tasks

Given Range Matrix **R**, task number **N**, hyperparameter **P**, optimizer **O**, and model **M**, update the Range Matrix

```
procedure Train(N,O,P,R)
    CurrParams ← model.Params
    R ← empty dictionary
    Train_Model() *this changes the model parameters*

    if N = 0 or N = 1 then
        PrevParams ← CurrParams
    end if

    CurrParams ← model.Params

    if R is empty then
        R ← GetRangeMatrix(P)
        O.RangeMatrix ← R
        O.PreviousParameters ← CurrParams
    else
        NewR ← GetRangeMatrix(P)
        for each key in NewR do
            U ← the element wise maximum between
                 NewR + CurrParams and R + PrevParams

            L ← the element wise minimum between
                 CurrParams − NewR and PrevParams − R

            range ← (U − L) ÷ 2
            center ← (U − range)
            R[key] ← range
            PrevParams[key] ← center
            O.RangeMatrix ← R
            O.PreviousParameters ← PrevParams
        end for
    end if
end procedure
```

Appendix 4. Calculating the Range Matrix

Algorithm 1 Calculating the Range Matrix

Given model **M**, training dataset **T**, optimizer **O**, loss function **L**, and hyperparameter **P**, return the calculated Range Matrix

```
procedure GetRangeMatrix(M, T, O, L, P)
    R ← empty copy of model parameter dictionary (this
        becomes the Range Matrix)
    for each (input, label) in T do
        O.Clear_Gradients()
        obtain model prediction y_hat
        loss ← calculate_loss(y_hat, label)

        ΔL ← P · loss*

        perform Backpropagation on loss to
        accumulate gradients for each parameter

        for (name, Parameter) in M do
            if Parameter has a gradient then
                R[n] ← |ΔL / parameter.gradient|
            end if
        end for
    end for

    for name, value in R do
        R[n] ← P / T.length
    end for

    return R
end procedure
```

* An alternate implementation would be $\Delta L \leftarrow P$

Appendix 5. Training protocol

Random seeds

42	1123581321	4815162342	7778889991	9021087654

Implementation details

	Epochs	Optimizer	lr	Batch Size	LR Scheduler	Stopping criterion
Range Matrix	200	Range Matrix (See section 3.4 & Appendix 2)	$1x10^{-2}$	128	Multi-step Learning Rate Decay (Decay factor 0.1, milestones [100,175])	Fixed epochs
EWC	200	SGD (momentum 0.9, weight decay 2e-4)	$1x10^{-3}$	128	Multi-step Learning Rate Decay (Decay factor 0.1, milestones [100,175])	Fixed epochs