Mind the Gap: Layerwise Proximal Replay for Stable Continual Learning

Oskar Hage[1]
Supervisors: Tom Viering[1]**, Gido van de Ven**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Oskar Hage
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Gido van de Ven, Alan Hanjalic

Abstract

Continual learning aims to train models that can incrementally acquire new knowledge over a sequence of tasks while retaining previously learned information, even in the absence of access to past data. A key challenge in this setting is maintaining stability at task transitions, where even methods like experience replay can suffer from temporary performance degradation known as the *stability gap*. In this work, we evaluate *Layerwise Proximal Replay* (LPR), a recently proposed optimisation strategy that constrains updates at the layer level to preserve internal representations of past data. We implement LPR on a simple multi-layer perceptron and benchmark it against an incremental joint training baseline on a domain-incremental variant of Rotated MNIST. To quantify the stability gap, we track accuracy drops immediately following task switches and compute local minima after transitions. Our results show that LPR consistently reduces the stability gap across a range of learning rates, with statistically significant improvements at higher values. However, this improvement comes at the cost of reduced performance on later tasks. These findings demonstrate that LPR significantly mitigates short-term performance degradation at task boundaries while maintaining high learning rates, offering a practical solution for increased stability in continual learning.

1 Introduction

Continual learning, also known as lifelong learning, enables machine learning models to progressively acquire and integrate new knowledge over time while retaining previously learned information [9, 2]. This paradigm addresses the challenge of training models on a sequence of tasks or data distributions, often without access to past data and under memory and computational constraints. A fundamental obstacle in this field is *catastrophic forgetting*, where the acquisition of new tasks leads to a significant degradation in performance on earlier ones [10, 8].

To mitigate catastrophic forgetting, *experience replay (ER)* is a widely adopted strategy that involves storing and reusing a subset of past data during training, thereby interleaving old and new examples during updates [10]. While effective in reducing forgetting, recent research by De Lange et al. has revealed that ER methods remain susceptible to performance drops, particularly immediately following task transitions. This phenomenon is defined as the *stability gap*, arising from unstable optimisation trajectories [3, 12]. The existence of this gap has significant implications, especially in safety-critical applications such as autonomous driving or medical diagnostics, where momentary lapses in predictive reliability can result in unacceptable outcomes [5]. Moreover, models suffering from repeated instability may require longer convergence times, increasing both computational cost and energy consumption [4, 14, 6, 11].

Beyond its practical impact, the stability gap also provides insight into suboptimal learning dynamics. The fact that performance often recovers after a task switch suggests that the model is capable of reconciling new and old knowledge, but does so through a transient period of degradation. This implies that the instability arises not from inherent task incompatibility, but from inefficient optimisation paths during transitions. Reducing the stability gap, then, is not an end in itself, but a means of encouraging more direct and robust updates that accelerate convergence and improve retention. An example of this phenomenon is shown in Figure 1, where the test accuracy on an earlier task drops sharply after each new task is introduced but gradually recovers over time.

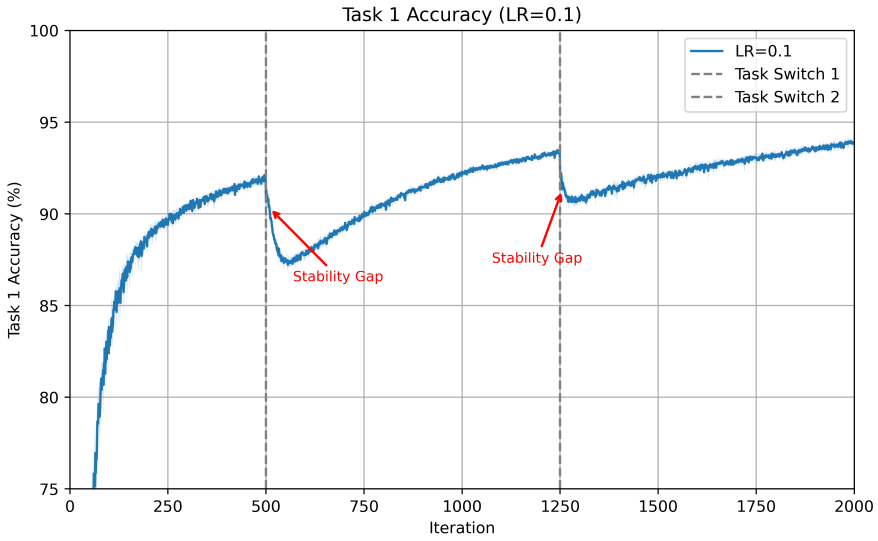


Figure 1: Illustration of the stability gap. The plot shows accuracy on Task 1 during baseline training (learning rate $\eta = 0.1$). Red arrows indicate accuracy drops after task switches, which recover over time, highlighting the temporary instability caused by suboptimal updates.

In this work, we evaluate *Layerwise Proximal Replay (LPR)*, a recently proposed optimisation strategy by Yoo et al. designed to enhance continual learning stability [15]. LPR operates by constraining updates at the layer level, which preserves internal representations of past data. Unlike methods that alter the loss function or project gradients, LPR functions as a *gradient preconditioning method*, updating each layer's weights using a preconditioner derived from replayed activations. This mechanism penalises updates that would cause substantial changes in the model's activations for past data, aiming to reduce large accuracy drops without unduly hindering new learning [15].

We implement LPR on a simple multi-layer perceptron and benchmark its performance against an incremental joint training baseline on a domain-incremental variant of Rotated MNIST [7]. To quantify the stability gap, we track accuracy drops immediately following task switches and compute local minima after transitions, focusing on immediate instability rather than long-term averages. Our findings consistently demonstrate that LPR significantly reduces the stability gap across a range of learning rates, with statistically significant improvements observed at higher learning rates where instability is typically most pronounced. We further investigate the role of the regularising coefficient ω, a key parameter in the LPR algorithm, hypothesising an optimal intermediate range that balances stability with the model's adaptability to new tasks. The results confirm that LPR provides substantial optimisation stability, improving long-term retention compared to the baseline.

The remainder of this paper is structured as follows: Section 2 provides essential background on continual learning, catastrophic forgetting, experience replay, and a detailed explanation of Layerwise Proximal Replay. Section 3 discusses the motivation behind studying the stability gap and outlines our hypotheses regarding LPR's effectiveness and its interaction with learning rates. Section 4 details the experiment methodology, including the

Rotated MNIST benchmark, LPR, model architecture, and the methodology for evaluating the stability gap. Section 5 presents our results, including baseline comparisons, LPR's performance across various hyperparameters, and statistical analyses. Section 6 discusses the implications of our findings and outlines avenues for future work. Finally, Section 7 concludes the paper with Section 8 addressing responsible research practices, reproducibility, and ethical considerations.

2 Background

This section provides the necessary context for our study. We first introduce the continual learning setting and the specific challenges it presents, particularly the phenomenon of catastrophic forgetting and the recently defined stability gap. We then review experience replay as a common mitigation strategy and describe our experimental setup based on the Rotated MNIST benchmark. Finally, we present Layerwise Proximal Replay (LPR), the core method evaluated in this work, and outline its mechanism for improving training stability across sequential tasks.

2.1 Continual Learning

Continual learning refers to the ability of machine learning models to continuously acquire and integrate new knowledge over time while retaining previously learnt information [9]. It addresses the challenge of learning from a sequence of tasks or data distributions, often without access to past data and under constraints on memory and computation [2].

We focus on the *domain-incremental* setting, where each task introduces data from a different input distribution while the output label space remains fixed, and no task identity is provided at test time [13].

A core challenge in continual learning is *catastrophic forgetting*, the phenomenon where learning new tasks causes performance degradation on earlier ones [2]. One approach to mitigate forgetting is *experience replay* (ER), which stores and reuses a subset of past data during training [10]. ER can significantly reduce forgetting by interleaving old and new examples during updates. However, recent work by De Lange et al. [3] demonstrates that ER is still susceptible to performance drops, particularly just after task switches. This phenomenon, called the *stability gap*, arises from unstable optimisation trajectories even when forgetting is controlled.

We evaluate models on a domain-incremental variant of Rotated MNIST, where digit images are rotated by fixed angles $[0°, 60°, 120°, 180°]$ across tasks. This setup changes the visual appearance of the inputs while keeping the output label space unchanged, requiring the model to adapt without explicit task identity. Implementation details are provided in Section 4.3.

While LPR's design suggests potential benefits for optimisation stability, its empirical evaluation has largely focused on long-term retention and final accuracy. In contrast, our study focuses on short-term stability immediately after task switches, specifically examining how it is influenced by the learning rate and the regularisation parameter ω, which is defined in the next subsection.

2.2 Layerwise Proximal Replay

To address the stability gap, that persists even with ER, we investigate LPR, a recently proposed optimisation strategy by Yoo et al. [15]. LPR improves continual learning stability by preconditioning each layer's gradient updates to preserve internal representations of past data.

Rather than modifying the loss function or projecting gradients, LPR constructs a per-layer preconditioner matrix from replayed activations, penalising parameter changes that would disrupt these activations. This allows the model to suppress disruptive updates without entirely preventing learning. A key element of LPR is ω, which controls the strength of the regularisation by scaling the penalty on activation changes; higher values enforce greater stability at the cost of flexibility. The implementation details of this mechanism are described in Section 4.2. In the next Section we hypothesise how changing ω could affect the trade-off between preserving prior knowledge and adapting to new tasks.

We reproduce the original algorithm by Yoo et al. in **Appendix A** for completeness. However, our implementation deviates in several key ways to simplify integration with our experimental setup, mainly for performance related reasons. The adapted procedure used in our experiments and a comparison to the original implementation is presented in Section 4.3.

3 Motivation and Hypothesis

This section outlines our motivation for studying LPR and its effect on the stability gap, and then presents the hypotheses that guide our experimental investigation.

3.1 Motivation

Despite the effectiveness of ER in mitigating catastrophic forgetting, recent work by De Lange et al. [3] has revealed a persistent issue: the *stability gap*, a temporary drop in performance immediately after task transitions. This phenomenon persists even under ideal replay conditions, such as incremental joint training, and suggests that the instability arises not from forgetting, but from poor optimisation trajectories during transitions.

Yoo et al. [15] proposed LPR to address this gap by preconditioning gradients at the layer level using replayed activations. While their results suggest improved retention and overall performance, their evaluation focused primarily on final accuracy and qualitative evidence of stability. In contrast, our work provides a more granular and quantitative assessment of LPR's ability to reduce short-term instability, using the stability gap metric introduced by De Lange et al. Furthermore, we examine LPR's interaction with learning rate, an aspect that was not explicitly explored in the original LPR study.

3.2 Hypothesis

We hypothesise that LPR could mitigate the stability gap by penalising updates that interfere with internal representations learned from earlier tasks. The strength of this penalty would be governed by the stability coefficient ω, which controls how strongly LPR regularises each layer's gradient.

We anticipate the following effects:

1. **When ω is very large**, LPR will aggressively preserve previous representations, suppressing instability but also reducing the model's ability to adapt. In this regime, we expect low stability gaps but poor learning on new tasks due to over-regularisation.

2. **When ω is very small**, LPR becomes nearly equivalent to standard SGD. Here, we expect little effect on the stability gap or performance, as regularisation is negligible.

We therefore hypothesise the existence of an intermediate ω range that balances stability and adaptability.

3.3 Interaction with Learning Rate

We further expect the benefits of LPR to be modulated by the learning rate η. At higher learning rates, gradient steps are more aggressive, increasing the likelihood of overshooting and destabilising prior knowledge. In such settings, LPR should be especially beneficial by tempering updates in sensitive directions. At lower learning rates, gradient descent behaves more conservatively, which may already provide some stability and thus reduce the benefit of LPR.

Therefore, we predict that:

1. LPR will offer the greatest improvements in stability when used with higher learning rates.

2. The optimal ω will likely shift depending on η, highlighting a potential interaction between the two hyperparameters.

In summary, our investigation seeks to explore the trade-off between stability and adaptability in LPR by varying both ω and η, and to assess whether this method can offer an improvement to continual learning systems.

4 Methodology

This section outlines the methodology used to evaluate the effectiveness of LPR in reducing the stability gap during continual learning. We begin by describing the baseline method, which serves as a comparative benchmark. We then detail our implementation of LPR, including its integration into a simple Multilayer Perceptron (MLP) architecture. Next, we present the experimental setup, including the domain-incremental Rotated MNIST benchmark, model specifications, and training procedures. Finally, we define our evaluation protocol.

4.1 Baseline Methods

As a baseline, we implement incremental joint training, where at each task the model is trained on the cumulative dataset from all previously seen tasks. This setup assumes full access to past data and thus corresponds to a form of perfect replay. Although this is not memory-constrained like typical experience replay methods, it provides a strong upper-bound baseline to isolate instability effects unrelated to forgetting. The goal is to create a benchmark on performance stability, highlighting how this idealised baseline handles task transitions in the absence of memory constraints or regularisation.

To assess learning stability, we measure the test accuracy on each task throughout training. In particular, we track the accuracy of Task 1 after each subsequent task is introduced, using the local minimum after each task switch to quantify performance degradation.

Figure 2 illustrates the stability gap that occurs in the baseline setup following each task transition. The plot tracks the test accuracy on Task 1 throughout training with a learning rate of 0.5. The vertical dotted lines indicate task switch points, after which the model begins learning from new rotated tasks.

As shown, accuracy on Task 1 declines sharply immediately after each transition, despite the model having access to data from all previous tasks, demonstrating the instability of incremental joint training. This instability is especially pronounced at higher learning rates, as is illustrated in the Appendix in Figure 6 .

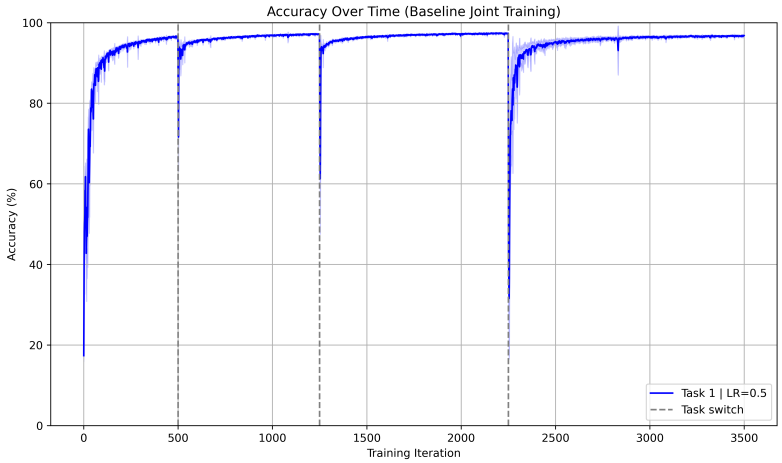


Figure 2: Task 1 accuracy on baseline experiment with Learning Rate 0.5. Note the sudden and large drops in accuracy after task switches.

4.2 Implementation of Layerwise Proximal Replay

We implemented LPR on top of a simple MLP architecture with three fully connected layers. The key mechanism in LPR is a layer-wise preconditioning of gradients using information derived from a replay buffer containing activations of previously seen data. After each task is trained, a subset of examples is stored in the buffer, and the corresponding hidden layer activations are used to construct positive-definite preconditioner matrices. These preconditioners are then used to modify the direction of incoming gradients for subsequent tasks, limiting abrupt changes to internal representations.

Our implementation follows the structure outlined in Algorithm 1 and is based on the method originally proposed by Yoo et al. [15]. However, several modifications were made to simplify integration with our experimental setup. Most notably,for computational efficiency, we use a fixed-size replay buffer per task (500 examples) and recompute preconditioners only once per task instead of at a fixed interval. We also apply the preconditioner only at the first and second hidden layers (fc1 and fc2), rather than across all layers. For completeness, the original algorithm from Yoo et al. is reproduced in the Appendix as Algorithm 2.

Algorithm 1 Training with Layerwise Proximal Replay

Initialise model f_θ and empty replay buffer $\mathcal{M}$
for each task $t = 1, 2, \ldots, T$ **do**
    Train model on joint data from tasks 1 to t
    if $t > 1$ **then**
        Compute preconditioner matrices P_ℓ^{-1} from $\mathcal{M}$ for each layer ℓ
    end if
    for each training iteration **do**
        Sample minibatch (x, y)
        Compute forward pass and loss $\mathcal{L}_{\text{CE}}$
        Compute gradients $\nabla\mathcal{L}$ by backpropagation
        if $t > 1$ **then**
            Precondition gradients at fc1 and fc2: $g \leftarrow P_\ell^{-1} g$
        end if
        Update weights with SGD
        Periodically evaluate accuracy on all tasks seen so far
    end for
    Store a fixed number of examples from task t into buffer $\mathcal{M}$
end for

The replay buffer stores up to 500 examples per task. Activations at key layers (input, first hidden, second hidden) are cached and used to compute the layer-specific preconditioners $P_\ell^{-1} = \left(I + \omega Z^\top Z\right)^{-1}$, where Z is the matrix of hidden activations from replay data, and ω is a regularisation strength. When $\omega = 0$, the preconditioner reduces to the identity matrix, i.e., $P_\ell^{-1} = I$, meaning that gradients are left unmodified and standard stochastic gradient descent (SGD) is recovered. In this regime, no stabilising effect is applied, and the model is free to make large representational shifts. As ω increases, the term $\omega Z^\top Z$ grows in influence, and the preconditioner increasingly penalises updates that would significantly alter the layer's response to previously seen data. Thus, ω dictates the trade-off between stability (preserving past internal representations) and plasticity (adapting to new inputs). Preconditioners are applied directly to gradient vectors at each layer prior to the weight update step.

This implementation allows the model to adapt to new tasks while maintaining representational consistency for old tasks, thereby reducing the stability gap and catastrophic forgetting.

4.3 Experimental Setup

We built our experiments on top of the `continual-learning` codebase by Gido van de Ven[1], which provides a modular framework to evaluate continual learning strategies. We adapted the codebase to suit our experimental design and implemented a custom training script for applying LPR to a multi-layer perceptron. Our implementation, including all experiments and logging utilities, is publicly available on the author's GitHub repository[2].

We evaluate the effect of LPR on the stability gap in continual learning using a domain-incremental variant of the Rotated MNIST benchmark. In this benchmark, each task

[1] `https://github.com/GMvandeVen/continual-learning`
[2] Oskar Hage's Github: `https://github.com/kingossi`

presents the original MNIST digit classification dataset [3] with a fixed image rotation, while the label space remains unchanged. We define four tasks with increasing rotations of $[0°, 60°, 120°, 180°]$.

For all experiments, we use a three-layer fully connected network MLP with ReLU activations and 400 hidden units per layer. We evaluate model performance across four different learning rates $\eta \in \{0.01, 0.1, 0.5, 1.0\}$ to examine how sensitivity to learning rate affects stability, both with and without LPR. All models are trained using stochastic gradient descent; furthermore, each experiment was run 5 times to ensure consistency and reliability of the results.

4.4 Evaluation and Visualisation

We define the stability gap as the difference between the test accuracy immediately before and the local minimum immediately after a task switch. This metric captures short-term degradation in previously learned tasks when the model begins learning new ones. It is not equivalent to the *average minimum accuracy* proposed by De Lange et al. [3], which averages the lowest accuracy over all subsequent training steps. Instead, we focus on local minima near each task transition to capture immediate instability. It also allows us to capture multiple drops in performance, as we take into account every task switch. We do this because the accuracy drop directly reflects the impact of task switches on optimisation dynamics, independent of long-term forgetting. By isolating the immediate response of the model after a task switch, our stability gap provides a measure of short-term instability. Moreover, this allows us to quantify the stabilising effect of methods like LPR, which aim to reduce abrupt representational drift without necessarily improving long-term averages.

The procedure to calculate the stability gap is shown in Algorithm 3 in Appendix D.

5 Results

In this section, we present our empirical evaluation of LPR on a domain-incremental variant of Rotated MNIST. Our analysis addresses three key aspects of continual learning performance.

First, we examine the *stability gap*, defined as the short-term drop in accuracy that occurs immediately after a task switch. Second, we investigate task-level retention, with a focus on the model's ability to preserve accuracy on earlier tasks, particularly mid-sequence ones, over time. Finally, we evaluate the average final accuracy at the end of training across all tasks, which reflects the cumulative effect of both forgetting and adaptation throughout the learning process.

5.1 Stability Gap Analysis

We first investigate how the stability gap varies with learning rate, both with and without LPR. The stability gap is defined as the drop in accuracy immediately after a task switch, as described in Section 4.4.

[3] `http://yann.lecun.com/exdb/mnist`

5.1.1 Visual Evidence of Stability Improvement

In Figure 3, the baseline (red curve) shows sharp accuracy drops on Task 1 following each task switch, with the most pronounced degradation occurring at the final transition. In contrast, LPR (shades of blue) maintains significantly more stable performance across task boundaries. This stabilising effect is especially evident for lower ω values (e.g., 0.01 and 0.1), which allow the model to adapt while still preserving prior knowledge. These visual trends support the hypothesis that LPR effectively mitigates short-term instability during continual learning.

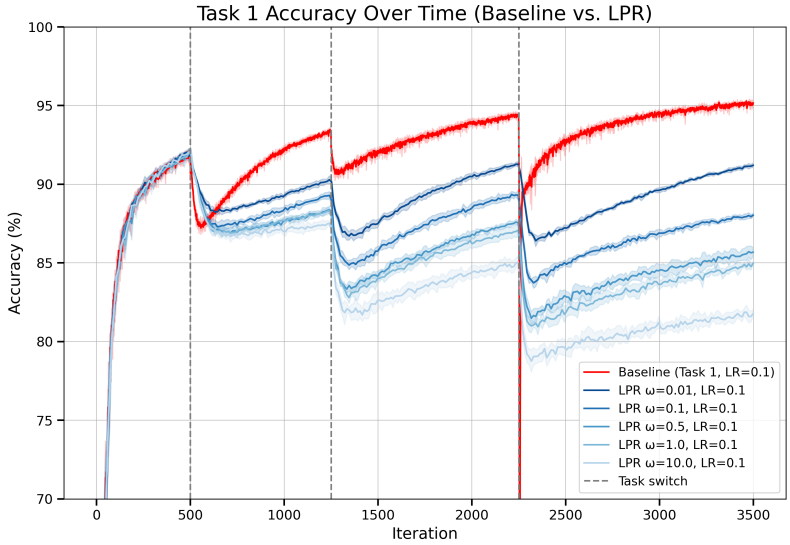


Figure 3: Task 1 accuracy over time for the baseline (in red) and LPR variants (in shades of blue) at learning rate $\eta = 0.1$. Lighter lines correspond to higher values of ω. While the baseline achieves higher peak performance, it also exhibits larger drops in accuracy immediately following task switches. In contrast, LPR reduces the severity of these drops in a ω-dependent manner. Dashed vertical lines indicate task transitions. Note the y-axis starting at 70% accuracy and Task 1 dropping to below 70 after Task switch 3 (49.60%).

5.1.2 Quantitative Stability Gap Measurements

To confirm these visual trends, we compute the average stability gap over five independent runs at each learning rate using Algorithm D found in Appendix 3, both for the baseline and the best-performing LPR configuration.

Baseline Results. Table 1 reports the mean and standard deviation of the stability gap for Task 1 on the baseline, computed over 15 samples per learning rate (3 task switches x 5 runs). As expected, higher learning rates lead to greater instability.

Table 1: Stability gap for training baseline, averaged over 15 samples per learning rate. Lower is better. Higher learning rates introduce a lot of instability.

Learning Rate	Mean Gap (%)	SEM (%)
0.01	0.22	0.07
0.10	14.71	4.55
0.50	48.12	5.61
1.00	46.37	6.50

LPR Results. Table 2 shows the corresponding results for the best LPR configuration (i.e., the lowest mean gap) at each learning rate. LPR substantially reduces the stability gap at all tested learning rates, in some cases by over 45 percentage points.

Table 2: Best LPR configuration per learning rate. Values reflect mean ± SEM over 5 runs. The last column reports the absolute reduction in stability gap relative to the baseline. Lower is better.

Learning Rate	Best ω	Mean Gap (%)	SEM (%)	Reduction (p.p.)
0.01	1.0	0.09	0.20	**0.13**
0.10	0.01	1.38	1.64	**13.33**
0.50	0.01	0.65	0.55	**47.47**
1.00	0.01	0.63	0.61	**45.74**

These quantitative results confirm that LPR offers robust regularisation against instability, even in high learning-rate regimes.

5.1.3 Statistical Significance

To assess whether the observed reductions are statistically reliable, we conducted one-tailed Welch's t-tests between the baseline and best LPR configurations at each learning rate. The differences were statistically significant ($p < 0.05$) for learning rates $\eta \in \{0.1, 0.5, 1.0\}$, with the strongest result at $\eta = 1.0$ ($p = 0.0015$). At $\eta = 0.01$, the result was borderline ($p = 0.050$), likely due to the already low gap in the baseline condition.

5.2 Task-Level Retention: Study on Task 2

To complement the stability gap analysis, we performed a focused evaluation of Task 2 accuracy throughout training. This provides insights into how well the model retains performance on a mid-sequence task, which is especially sensitive to interference from both earlier and later tasks in the sequence.

In continual learning, even when using incremental joint training that includes data from all previously seen tasks, performance can still degrade as the model adapts to new input distributions. To assess this degradation, we track Task 2 accuracy at three critical moments during training: first, immediately after Task 2 concludes but before Task 3 begins (i.e., at iteration ≤ 1250); second, following the completion of Task 3 (iteration ≤ 2250); and finally, after training on all tasks has concluded (iteration ≤ 3500).

Visually we present in Figure 4 the accuracy of Task 2 throughout training. The red curve shows the baseline performance without LPR. The blue curves show the performance with LPR across different ω values. The baseline achieves higher accuracy overall, but exhibits greater volatility at task transitions, particularly after the second task switch. The quantitative results are summarised in Table 3, comparing baseline and LPR performance across various ω values. We report accuracy immediately before each task switch as opposed to our previous technique of calculating stability gaps. As we shift our focus to the model's ability to learn new tasks, we change the focus to the achieved accuracy on tasks.

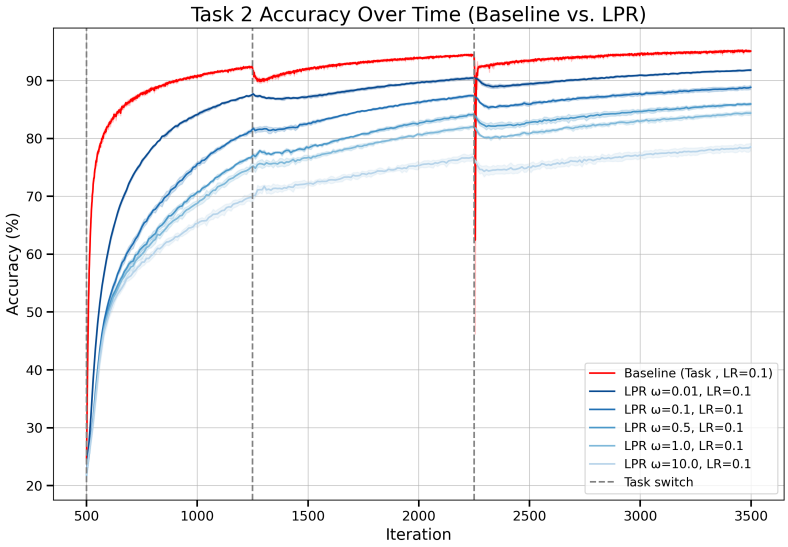


Figure 4: Task 2 accuracy over time with learning rate 0.1. The red curve shows baseline performance , while the blue curves show performance with LPR across different ω values. Task 2 begins at iteration 500, marked by the first vertical dashed line.

Table 3: Accuracy (%) immediately *before* each task switch at learning rate 0.1. Values are the mean ± SEM over five runs; bold figures mark the highest mean per column.

Method	**Before Switch 1**	**Before Switch 2**	**At the final iteration**
LPR $\omega = 0.01$	92.11 ± 0.06	87.70 ± 0.08	88.84 ± 0.05
LPR $\omega = 0.1$	92.09 ± 0.14	81.77 ± 0.12	84.02 ± 0.14
LPR $\omega = 0.5$	91.87 ± 0.21	77.16 ± 0.12	77.82 ± 0.16
LPR $\omega = 1.0$	92.23 ± 0.08	75.23 ± 0.24	73.93 ± 0.13
LPR $\omega = 10.0$	**92.29 ± 0.09**	70.02 ± 0.34	66.12 ± 0.33
Baseline (no LPR)	91.94 ± 0.04	**92.37 ± 0.05**	**93.49 ± 0.09**

Interpretation. Compared to the baseline, LPR slows down learning in later stages. While all LPR variants begin with competitive accuracy before the first task switch, their performance declines markedly by the second switch and final iteration, especially at higher

values of ω. The baseline, in contrast, maintains stable and even improving accuracy throughout, achieving the best final performance overall.

This suggests that while LPR can regularise initial learning effectively, it may overly constrain adaptation to new tasks, particularly at larger ω. Among LPR variants, smaller values of ω (0.01-0.1) better preserve short-term learning while limiting forgetting, but even these fall short of the baseline in final accuracy. Thus, although LPR manages to reduce the stability gap, it does so at the cost of long-term performance.

5.3 Final Accuracy Across Tasks

While stability gaps and mid-sequence retention provide valuable insight, in the context of continual learning it is also important to evaluate end-of-training performance. In this subsection, we compare the average final test accuracy achieved by the baseline and LPR across a range of learning rates and regularisation strengths across all tasks. Table 4 presents the average and standard error of the mean (SEM) of the final test accuracy over 5 runs for both the baseline and LPR across a range of learning rates and, for LPR, different regularisation strengths (ω). For the baseline, performance improves with increasing learning rate up to 0.5, but degrades sharply at LR = 1.0, likely due to high instability. In contrast, LPR demonstrates more stable and competitive performance across learning rates, particularly for lower values of ω. Notably, at higher learning rates (e.g., 1.0), LPR significantly outperforms the baseline, which struggles with instability.

Table 4: Final test accuracy (%) across learning rates for the baseline and LPR with different ω values. Values are reported as mean ± SEM over 5 runs. Best result per learning rate is bolded.

LR	**Baseline**	$\omega = 0.01$	$\omega = 0.1$	$\omega = 0.5$	$\omega = 1.0$	$\omega = 10.0$
0.01	**71.66 ± 0.16**	60.05 ± 0.19	50.31 ± 0.15	47.64 ± 0.12	46.90 ± 0.27	44.44 ± 0.26
0.10	**94.74 ± 0.03**	90.57 ± 0.05	86.76 ± 0.07	83.27 ± 0.07	81.29 ± 0.12	75.09 ± 0.30
0.50	96.51 ± 0.05	**96.10 ± 0.03**	94.66 ± 0.02	93.13 ± 0.07	92.69 ± 0.04	89.79 ± 0.13
1.00	78.86 ± 15.46	**96.26 ± 0.15**	94.58 ± 0.49	93.77 ± 0.47	93.10 ± 0.39	89.36 ± 3.43

Interpretation. LPR improves over the baseline in high learning rate settings, where the baseline suffers from catastrophic forgetting and optimisation instability. The configuration $\omega = 0.01$ yields the best trade-off across all learning rates, achieving both high final accuracy and low variance for all learning rates, except for when the LR is 0.01. As ω increases, performance gradually degrades, suggesting over-regularisation. At higher learning rates this degradation is much smaller, which implies that the strict regularisation imposed by large ω values may be compensating for the instability introduced by aggressive updates.

These results reinforce the earlier findings: LPR not only stabilises learning after task switches but also supports stronger overall performance in regimes where the baseline breaks down due to instability. However, at lower learning rates, it does so at the cost of learning future tasks.

5.4 Discussion of Results

Our results demonstrate that LPR is effective at improving optimisation stability in domain-incremental continual learning. Specifically, LPR significantly reduces the accuracy drops

that occur immediately after task switches. These improvements are consistent across a range of learning rates and are most pronounced at higher values, where baseline models are especially variable.

This outcome aligns well with our original hypotheses: small to moderate values of the stability coefficient ω (e.g., 0.01-0.1) strike a good balance between preserving past representations and allowing new learning. Larger ω values, in contrast, lead to over-regularisation, which manifests as underfitting on later tasks.

Importantly, LPR achieves these benefits without requiring a reduction in learning rate. This is notable because many continual learning methods rely on conservative learning schedules to maintain stability. In contrast, LPR maintains robustness even under aggressive gradient steps, making it compatible with fast training setups.

Overall, the empirical findings confirm our expectations and hypotheses: LPR reduces instability at task switches, especially for higher learning rates, and exhibits a predictable regularisation trade-off controlled by ω, balancing stability against plasticity in learning. These results validate both our experimental design and the underlying mechanism of LPR as these trends are also broadly consistent with the findings reported by Yoo et al. [15], who proposed LPR and observed similar trade-offs between stability and plasticity. While not surprising, they offer a clear and replicable contribution to the growing body of work on continual learning stability.

6 Discussion & Further Work

This work demonstrates the effectiveness of LPR in reducing the stability gap in continual learning settings, particularly within a domain-incremental Rotated MNIST benchmark. Our experiments show that LPR consistently reduces the accuracy drops observed after task transitions, especially at higher learning rates. This aligns with findings from Mirzadeh et al. [8], who observed that training regimes significantly affect stability and plasticity in continual learning. The statistical significance of these reductions, as shown in 5.1.3, especially for learning rates of 0.1, 0.5, and 1.0, underscores LPR's effectiveness as an optimisation strategy.

A key finding is the existence of an optimal range for the regularisation parameter ω. While small to moderate values of ω (e.g., 0.01-0.1) effectively balance short-term performance and long-term retention, excessively large values can lead to over-regularisation, potentially hindering the model's ability to adapt to new tasks and resulting in underfitting on later tasks. This highlights a trade-off between stability and plasticity, where overly aggressive constraints on parameter updates can compromise the model's capacity for continued learning.

Another important implication of our results is that LPR achieves these stability improvements without necessitating a reduction in the learning rate. This is particularly important because high learning rates are known to accelerate convergence in deep networks [1], and being able to retain these benefits while maintaining stability makes LPR a practically attractive solution for continual learning.

One limitation of this study is the relatively simple experimental setting: we evaluated LPR on a small-scale domain-incremental benchmark (Rotated MNIST) using a shallow multi-layer perceptron. While this choice made it easier to isolate the effect of LPR, it limits the generalisability of our findings to more complex datasets or architectures. In addition, our evaluation focuses on short-term stability (i.e., accuracy drops after task switches), and

does not quantify other aspects such as memory efficiency, training time, or the time needed to recover fully from the stability gap.

Despite the promising results, several avenues for further work exist. One area is to explore the interaction between LPR and different experience replay strategies. While LPR is evaluated in conjunction with incremental joint training in this study, investigating its performance with more advanced or adaptive replay buffer management techniques could yield further improvements. Additionally, future research could focus on theoretically understanding the optimal ω range and how it scales with network depth, batch size, and or even introduce an adaptive ω similar to research regarding learning rates[16].

Another direction involves evaluating LPR on more complex and diverse continual learning benchmarks, including class-incremental or task-incremental scenarios with larger datasets and more varied task distributions. This would help assess the scalability and generality of LPR beyond the Rotated MNIST setting. Finally, exploring the computational overhead associated with the periodic recalculation of preconditioners and investigating methods to optimise this process for online applications would be valuable.

7 Conclusion

Our experiments, conducted on a domain-incremental Rotated MNIST benchmark, demonstrate that LPR consistently reduces the stability gap, particularly at higher learning rates where instability is typically more pronounced. Statistical analysis confirmed these reductions are significant for learning rates of 0.1, 0.5, 1.0, and borderline for 0.01. A key finding is the existence of an optimal range for the stability coefficient ω, which balances short-term performance and long-term retention. Our results establish a critical trade-off: while small to moderate ω values are beneficial, excessively large values can lead to over-regularisation, hindering adaptation to new tasks and causing underfitting.

LPR achieves these stability improvements without necessitating a reduction in the learning rate, preserving the model's responsiveness to novel inputs. This balance makes LPR a practical method for continual learning scenarios. Furthermore, although LPR generally achieves lower final accuracies, it demonstrates superior long-term retention compared to the baseline, maintaining significantly lower variance in final accuracy on earlier tasks at the final iteration of training.

Future work can explore the interaction of LPR with various experience replay strategies, theoretically and empirically investigate the optimal ω range and its interaction with learning rate, and evaluate LPR on more complex and diverse continual learning benchmarks. Additionally, optimising the computational overhead associated with preconditioner recalculation for online applications would be valuable. Overall, LPR offers an effective solution for improving stability in continual learning.

8 Responsible Research

8.1 Reproducibility and Open Science

All experiments described in this paper were conducted using publicly available datasets and openly shared code. To ensure full replicability, the training scripts, models, and evaluation tools, including our implementation of Layerwise Proximal Replay (LPR), are publicly accessible on the author's GitHub repository: `https://github.com/kingossi`. This reposi-

tory includes instructions for reproducing all experimental results, including hyperparameter configurations and logging outputs.

8.2 Use of Large Language Models

Large Language Models (LLMs), specifically ChatGPT, were used to assist in drafting some parts of the final paper, such as text refinement, LaTeX formatting, and summarisation. The technical content, code, and experimental results were independently designed and implemented by the author but often improved and/or debugged with the help of ChatGPT. Examples of LLM-assisted prompts and outputs are provided in the appendix B for transparency.

8.3 Data and Licensing

All experiments use the Rotated MNIST benchmark derived from the original MNIST dataset [7], which is freely available for academic research. No proprietary data, restricted software, or paid frameworks were used in this project.

8.4 Ethical and Societal Considerations

This work investigates algorithmic improvements in continual learning using a digit classification task. As such, it does not involve human subjects, biometric data, or decision-making systems deployed in sensitive domains. While this work constitutes fundamental research in machine learning, we acknowledge that continual learning techniques may eventually find application in real-world systems. We explicitly distance ourselves from any use of this research in the development of weaponry, surveillance infrastructure, or technologies that contribute to environmental harm.

References

[1] Hengjie Cao, Yifeng Yang, Mengyi Chen, Ruijun Huang, Fang Dong, Jixian Zhou, Mingzhi Dong, Yujiang Wang, Dongsheng Li, David A. Clifton, Robert P. Dick, Qin Lv, Fan Yang, Tun Lu, Ning Gu, and Li Shang. Large learning rates without the agonizing pain: Dispelling the curse of singularities in deep neural networks, 2024.

[2] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

[3] Matthias De Lange, Gido M. van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. In *International Conference on Learning Representations (ICLR)*, 2023.

[4] Sujan Harun, Kashif Zafar, Kamran Siddique, Arif Rahman, and Shah Hossain. How efficient are today's continual learning algorithms? In *ResearchGate*, 2023.

[5] Tim Hess, Tinne Tuytelaars, and Gido M. van de Ven. Two complementary perspectives to continual learning: Ask not only what to optimize, but also how. In *Proceedings of*

the 1st ContinualAI Unconference, volume 249 of *Proceedings of Machine Learning Research*, pages 37–61. PMLR, 2023.

[6] Byungmin Kim, Kwangmin Jeong, Seungwoo Ryu, Jaehong Park, Hwalsuk Kim, Youngmin Lim, Changbin Hwang, and Kwangjun Lee. Energy-efficient and timeliness-aware continual learning management system. *Energies*, 16(24):8018, 2023.

[7] 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.

[8] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[9] 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.

[10] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[11] Andrea Sorrenti, Matthias De Lange, Rahaf Aljundi, and Tinne Tuytelaars. Selective freezing for efficient continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 4119–4127, 2023.

[12] Gido M. van de Ven. Stabilitygap – continual learning repository. `https://github.com/GMvandeVen/continual-learning/tree/master/StabilityGap`, 2023.

[13] Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022.

[14] Peng Wang, Chengyu Fan, Feng Zheng, Jing Jiang, Xinlong Pan, and Xudong Liang. Cost-efficient continual learning with sufficient exemplar memory. *arXiv preprint arXiv:2502.07274*, 2025.

[15] Jason Yoo, Yunpeng Liu, Frank Wood, and Geoff Pleiss. Layerwise proximal replay: A proximal point method for online continual learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. PMLR, 2024.

[16] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

A Appendix A. Original LPR Algorithm

For completeness, we include Algorithm 1 as proposed by Yoo et al. [15], which serves as the conceptual basis for our implementation. Note that our actual procedure differs in several respects, as described in Section 4.2.

Algorithm 2 Layerwise Proximal Replay (reproduced from Yoo et al. [15])

Input: network parameters θ, learning rate η, per batch gradient steps count S, preconditioner update interval T, layerwsie stability hyperparameters ω_0^ℓ
Output: trained parameters θ
Initialise replay buffer: $\mathcal{M} \leftarrow \{\}$
Initialise preconditioner inverses $\Lambda_\ell \leftarrow I$ for all ℓ
for $\tau \in \{1, \ldots, \infty\}$ **do**
    Obtain batch $\mathcal{D}_\tau$
    for $s \in \{1, \ldots, S\}$ **do**
        Compute loss $\mathcal{L}(\theta)$ using $\mathcal{D}_\tau$ and $\mathcal{M}$
        Compute gradient $\nabla\mathcal{L}(\theta)$
        for each layer $\ell \in \{1, \ldots, L\}$ **do**
            $\Theta^{(\ell)} \leftarrow \Theta^{(\ell)} - \eta\Lambda_\ell \nabla_{\Theta^{(\ell)}}\mathcal{L}(\theta)$
        end for
    end for
    Update $\mathcal{M}$ with $\mathcal{D}_\tau$
    if $\tau \bmod T = 0$ **then**
        Obtain feature tensor X_{mem} from $\mathcal{M}$
        Set $Z^{(1)} \leftarrow X_{\text{mem}}$, $n \leftarrow$ batch size
        for each layer index $\ell \in \{1, \ldots, L\}$ **do**
            $\omega^\ell \leftarrow \omega_0^\ell / n$
            $\Lambda_\ell \leftarrow \left(\omega^\ell Z^{(\ell)\top} Z^{(\ell)} + I\right)^{-1}$
            $Z^{(\ell+1)} \leftarrow \phi^{(\ell)}(Z^{(\ell)}\Theta^{(\ell)})$
        end for
    end if
end for

B Appendix B. Example Prompts (LLM Assistance Disclosure)

In line with responsible research practices, this project made limited use of a large language model (ChatGPT by OpenAI) to assist with technical debugging, visualisation improvements, and formatting. The following examples illustrate typical prompts used during development:

Prompt 1: Visualising Standard Deviation as Shaded Area

> I have 5 runs of accuracy over time for each task. How can I plot the mean with a shaded area showing standard deviation in matplotlib?

Prompt 2: Fixing a Shape Mismatch in a Matrix Operation

> I'm trying to precondition gradients using matrix multiplication in PyTorch. I'm getting a shape mismatch between the preconditioner and the gradient. How do I fix this error: *Full error statement*

Prompt 3: Improving Sentence Clarity and Checking Grammar

> Can you check the grammar of this sentence from my paper? It's too long and hard to read. How can I split it into two clearer sentences while keeping the meaning accurate?

Prompt 4: LaTeX Table Alignment

> My table columns with SEM values aren't aligned in LaTeX and the boldness isn't showing. How do I make them look clean and centered and have the last column in bold?

C Appendix C. Full Stability Gap Results

To complement the summary statistics presented in the main text (Table 2), we report the full stability gap results across all tested configurations. This includes both the baseline and all combinations of learning rate and regularisation coefficient ω for Layerwise Proximal Replay (LPR). Each value is computed over $n = 15$ gap measurements (3 task switches x 5 runs). Bold values indicate the best (i.e., lowest) mean gap per learning rate.

Table 5: Stability gap (mean ± std) for all ω values at each learning rate. Bold values indicate the lowest gap per group. All results are averaged over $n = 15$ measurements (3 task switches × 5 runs).

(a) Learning rate $\eta = 0.01$

ω	Mean ± Std (%)
–	0.22 ± 0.28
0.01	0.14 ± 0.20
0.10	0.10 ± 0.18
0.50	0.27 ± 0.41
1.00	**0.09 ± 0.20**
10.0	0.15 ± 0.20

(b) Learning rate $\eta = 0.10$

ω	Mean ± Std (%)
–	14.71 ± 17.64
0.01	**1.38 ± 1.64**
0.10	1.55 ± 1.86
0.50	1.20 ± 1.74
1.00	1.86 ± 2.25
10.0	1.79 ± 2.29

(c) Learning rate $\eta = 0.50$

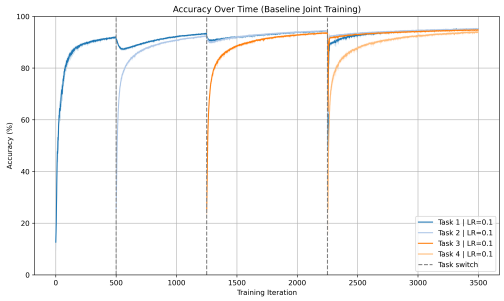
ω	Mean ± Std (%)
–	48.12 ± 21.75
0.01	**0.65 ± 0.55**
0.10	0.96 ± 0.63
0.50	1.00 ± 0.90
1.00	1.20 ± 0.94
10.0	1.49 ± 0.94

(d) Learning rate $\eta = 1.00$

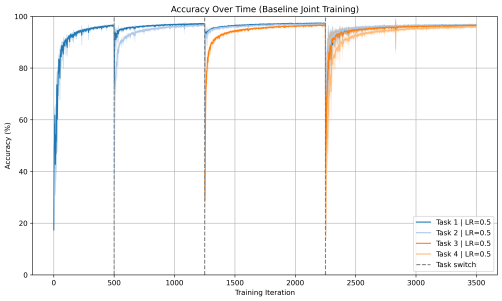
ω	Mean ± Std (%)
–	46.37 ± 25.19
0.01	**0.63 ± 0.61**
0.10	1.44 ± 0.97
0.50	3.05 ± 4.49
1.00	3.36 ± 3.95
10.0	5.23 ± 7.59

Additionally we have added Figure 5 which shows task-wise test accuracy over time under the baseline incremental joint training setup for four different learning rates.

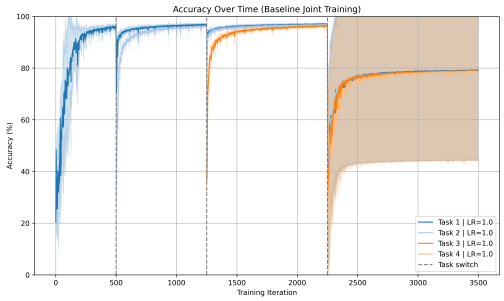
(a) Baseline accuracy, $\eta = 0.01$



(b) Baseline accuracy, $\eta = 0.10$



(c) Baseline accuracy, $\eta = 0.50$



(d) Baseline accuracy, $\eta = 1.00$

Figure 5: Task 1 accuracy over time under baseline incremental joint training across four learning rates. Each line shows mean accuracy per task; shaded regions indicate standard deviation over five runs.

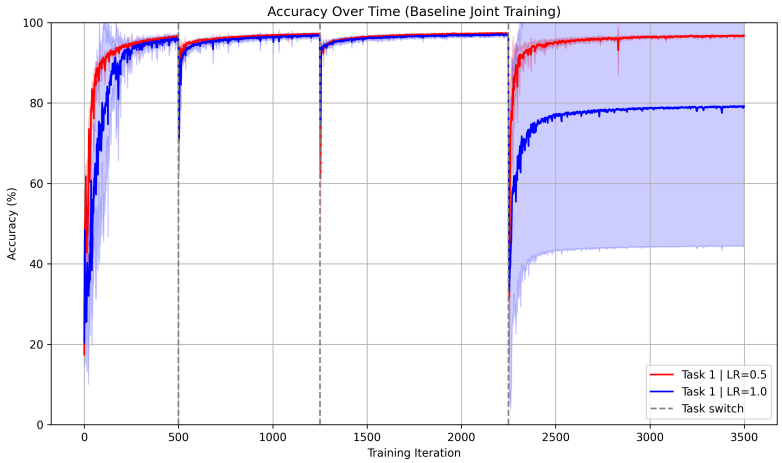


Figure 6: Task 1 accuracy over time under baseline joint training for two learning rates ($\eta = 0.5$ and $\eta = 1.0$). The higher learning rates exhibit high volatility, particularly following the final task switch, where performance degrades sharply and recovers slowly. Shaded regions indicate standard deviation across runs.

D Appendix D. Stability Gap Algorithm

Algorithm 3 Stability Gap Calculation Procedure

for each learning rate lr **do**
  for each run r **do**
    Load test accuracy history for all tasks
    for each task switch t (e.g., after Task 1 and 2) **do**
      $acc_{\text{before}} \leftarrow$ accuracy at iteration before switch
      $acc_{\text{after}} \leftarrow$ local minimum after switch within 10 iterations
      gap $\leftarrow acc_{\text{before}} - acc_{\text{after}}$
      Store gap in list for this lr
    end for
  end for
end for
Compute mean and standard deviation of gaps for each lr