Stability Gap in Continual Learning: The Role of Learning Rate

Paulina Sobocińska[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: Paulina Sobocińska
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Gido van de Ven, Alan Hanjalic

Abstract

Continual learning aims to enable neural networks to acquire new knowledge sequentially without forgetting what they have already learned. While many strategies have been developed to address catastrophic forgetting, a subtler challenge known as the stability gap—a temporary drop in performance immediately after switching tasks—remains insufficiently understood. Recent work suggests that the learning rate may influence this phenomenon by shaping the model's optimization trajectory. This paper systematically investigates how different constant learning rates affect the stability gap and whether dynamic learning rate scheduling can mitigate it. Experiments on Rotated MNIST with perfect replay show that smaller constant learning rates reduce the immediate drop but slow down recovery and convergence, while larger rates yield higher final accuracy but at the cost of a more severe gap. Scheduling methods, including CyclicLR and our custom IncreaseLROnPlateau, demonstrate potential for balancing this trade-off, but also introduce new challenges such as intra-task fluctuations. Overall, a carefully tuned constant learning rate provides the most robust trade-off in this setting. By isolating and quantifying these effects, this work offers insights for selecting and tuning learning rates in continual learning and lays the groundwork for future studies on more effective scheduling strategies. All code and experiments are publicly available at: https://github.com/wjssk/learning-rate-in-stability-gap.

1 Introduction

In traditional supervised learning, neural networks are trained under the assumption that all data is available at once. But in real-world applications, learning is often a continuous process, where tasks arrive sequentially, and previously seen data may no longer be accessible. This paradigm, known as **continual learning (CL)**, faces a significant challenge known as the **stability-plasticity dilemma** [7] - the model has to preserve its old knowledge (*stability*) while also trying to adapt to a new task (*plasticity*). High plasticity and low stability models suffered from **catastrophic forgetting** [2] - upon learning a new task, the model completely overwrote (forgot) previously acquired knowledge. Numerous methods have been developed to mitigate forgetting, such as replay-based strategies that reintroduce data from past tasks during training [10][12][8][9], or parameter regularization techniques, which constrain updates to important weights to preserve knowledge from previous tasks [5][17]. While these approaches have significantly reduced catastrophic forgetting, some limitations remain unsolved.

Related work Recent research conducted by De Lange et al. [1] has drawn attention to a subtler, yet potentially damaging issue: **the stability gap**. This term refers to a temporary but sudden drop in accuracy on previous tasks immediately after the model begins learning a new one - even when using replay-based methods, knowledge distillation or parameter regularization. Figure 1 is reproduced from De Lange et al. [1], and illustrates the stability gap.

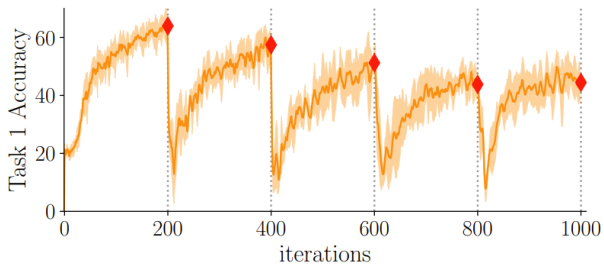


Figure 1: Stability gap example, reproduced from De Lange et al. [1]. The orange curve shows the accuracy for Task 1 throughout the whole training process. The vertical lines indicate the start of training on new tasks.

Although the model often recovers, the authors emphasize that this instability poses potential risks in real-world applications. In particular, they highlight concerns about maintaining reliable performance in safety-critical domains (e.g. the medical field), the possibility of adversaries exploiting momentary weaknesses, and the broader scientific importance of understanding why such gaps arise despite mitigation efforts.

Hess et al. [3] suggest that to mitigate the stability gap we should focus on the optimization trajectory instead of the optimization objective - i.e., is there a path in the parameter space that will lead us to the same endpoint, but without the stability gap? One of the main components shaping these trajectory paths is **the learning rate (LR)**. A large LR means the model takes bigger steps through the parameter space, which can help it escape local minima and converge faster. In contrast, a small LR results in smaller, more cautious steps, but can make the model more likely to get stuck in a local minimum.

Contributions To address the idea, this study performs a two-stage empirical investigation. First, a wide range of constant learning rates is tested to reveal general patterns in how step size affects the stability gap. The second stage explores whether carefully tuned learning rate schedulers can reduce the stability gap by adjusting the step size dynamically during training. By systematically isolating the effect of the learning rate in a domain-incremental scenario with perfect replay, this work aims to provide clearer insights into how the learning rate influences the stability gap and to guide more informed choices of this hyperparameter in continual learning models. The main research question of this study is therefore: **How does the learning rate influence the stability gap in continual learning, and can it be reduced through scheduling?**

Two key hypotheses are formulated.

H1 Smaller learning rates may lead to a smaller stability gap, as smaller steps might prevent sudden drops in accuracy at task transitions.

H2 A carefully tuned scheduled learning rate could further reduce the stability gap by using both a smaller and

larger LR to our advantage - smaller for stability, and higher for escaping local minima.

The remainder of this paper is organized as follows: section 2 details the experimental design, including the baseline model, custom metrics used to quantify the stability gap, and explanation of the chosen schedulers used in this study and their evaluation method; section 3 describes the experimental setup and presents the main results for both constant and scheduled learning rates; section 4 discusses the key insights, reflects on the study's limitations, and outlines recommendations for future research; section 5 answers the main research question, evaluates the proposed hypotheses, and highlights the practical implications; finally, section 6 addresses responsible research practices, ethical considerations, and the reproducibility of all experiments.

2 Methodology

This study aims to isolate and analyze the impact of the learning rate on the stability gap in continual learning. This section provides a detailed description of the baseline model architecture, the dataset and the learning scenario used in the experiments, the stability gap metrics used for the analysis, and the selected learning rate schedulers together with the motivation behind them, as well as the evaluation method for all approaches.

2.1 Baseline Model

Architecture The baseline model used in this study is a fully connected neural network with two hidden layers, each containing 400 units and using the ReLU activation function. The model operates on 28x28 grayscale images and performs multi-class classification over ten digit classes (0-9). No regularization techniques were used in the model to isolate the effect of the learning rate on performance.

Dataset The model is trained on the rotated MNIST benchmark [6], consisting of three tasks with input rotations of 0°, 80°and 160°. The rotations do not contain multiples of 90, to avoid the confusion of numbers 6 and 9.

Learning Scenario This study follows the **domain-incremental** learning scenario, as described by van de Ven et al. [15][14][16]. In domain-incremental learning, the model solves the same classification problem in each task (here, recognizing digits), but images in each task have some modifications. For example, in Rotated MNIST, all tasks use the same digit labels (0-9), but each task contains images rotated by a different angle.

We employ **incremental joint training**, meaning that at each task, the model is trained not only on the new task data, but also retrained on the full datasets of all previously encountered tasks. This simulates an idealized form of experience replay or "perfect regularization" and is known to still exhibit the stability gap [3], making it suitable for isolating the effect of learning rate on this phenomenon.

Accuracy Tracking Accuracy on all tasks is tracked throughout the whole training and reported in the plots. While the stability gap metrics are only computed for Task 1, visualizing the performance on Tasks 2 and 3 provides valuable context — for instance, a lack of stability gap might coincide with poor learning of the new task, which would not be apparent from Task 1 alone.

2.2 Stability Gap Metrics

To quantify the stability gap observed after each task switch, we introduce a set of intuitive metrics specifically designed to capture both the severity and duration of the temporary performance drop. Since there is currently no standard evaluation protocol for measuring the stability gap in continual learning, these metrics were developed based on visual inspection.

Each metric is computed based on the per-iteration accuracy curve of the first task. As illustrated in Figure 2, the main components include:

Final Accuracy (F-ACC) For each task, we record its final test accuracy at all relevant points during training: for Task 1, we report accuracy after completing Tasks 1, 2, and 3; for Task 2, after completing Tasks 2 and 3; for Task 3, after Task 3 only.

Height (H) Shows the vertical size of the gap; the difference between the final accuracy and the minimum accuracy reached immediately after the task switch (measured in percentage points).

Recovery time (REC-TIME) Shows the horizontal size of the gap; the number of iterations required for the model to recover from the drop and reach a specified percentage of its final accuracy, measured from the task switch to the recovery point. Rather than measuring a single threshold, we report multiple recovery times: the iteration count needed to return to 97%, 98%, 99%, and 100% of the final accuracy value before the task switch, respectively. This provides a more fine-grained view of the speed and shape of recovery.

Drop slope (DROP-SLOPE) The steepness of the accuracy drop, defined as the slope of a line of best fit over the accuracy values between the task switch and the minimum.

Knowledge loss based on the area (AUC-LOSS) Measures how much accuracy was lost on the previous task during the training of a new one. It compares the area under the actual accuracy curve $(actualArea)$ to the area that would exist if accuracy had stayed constant at its final value throughout the whole training period $(idealArea)$. The formula used is $1 - \frac{actualArea}{idealArea}$. This metric captures the severity of forgetting throughout the whole training period. A value closer to 0 indicates minimal forgetting, while values approaching 1 suggest a prolonged and severe loss of knowledge.

While these metrics are not based on prior work, they are grounded in observable behavior of the stability gap and aim to offer a more nuanced understanding of its dynamics.

All metrics are computed independently for each run and learning rate, then aggregated and reported as mean ± standard deviation.

2.3 Learning Rate Schedulers

In addition to evaluating a range of constant learning rates, this study also tests two dynamic learning rate schedulers to investigate whether adaptive step size adjustment can further

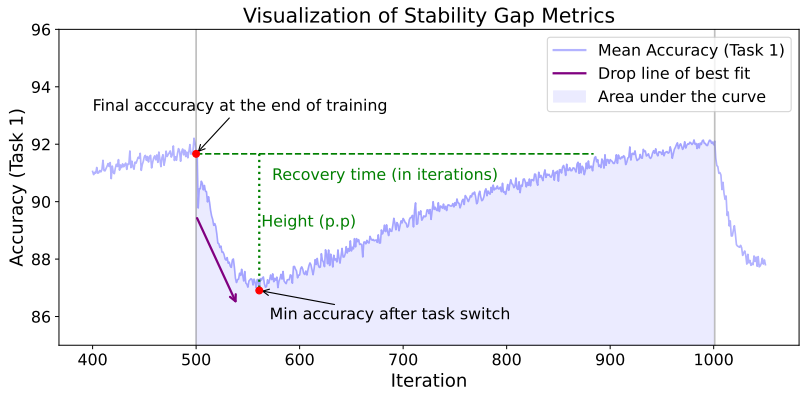


Figure 2: Visual illustration of the stability gap metrics defined in this section, including drop height, recovery time, drop slope, and gap area.

mitigate the stability gap. The core hypothesis motivating the choice of schedulers used is that smaller learning rates tend to produce smaller stability gaps but increase the risk of the model becoming trapped in suboptimal local minima. In contrast, larger learning rates can help escape poor local minima but can cause greater instability.

To balance these effects, both schedulers are designed to start with a low learning rate to limit the initial drop in accuracy and then increase the step size as training progresses, aiming to accelerate convergence and prevent the model from getting stuck in a local minimum. This approach is inspired by the general concept of **learning rate warmup**, which has been shown to reduce large gradient fluctuations at the beginning of training by gradually increasing the learning rate instead of setting it high from the start [4]. While warmup is commonly used to stabilize training dynamics in single-task scenarios, this study tests whether a similar principle can help smooth out the stability gap. Additionaly, since in realistic continual learning scenarios the number of tasks is typically unknown in advance, it is impractical to predefine how the scheduler should evolve across tasks. To reflect this, each scheduler is reset to its initial state at the beginning of every new task.

Cyclic LR Introduced by Smith [13], the Cyclic Learning Rate scheduler periodically oscillates the learning rate between a defined minimum and maximum value according to a chosen cycle length. This scheduler is available as a standard option in PyTorch's `torch.optim.lr_scheduler` module, making it straightforward to integrate into modern training pipelines. The scheduler has 3 modes, and all are tested in this study. They are also shown in Figure 3.

- **triangular** — a simple triangular cycle with constant amplitude.
- **triangular2** — a triangular cycle where the amplitude decreases by half each cycle, leading to progressively smaller oscillations.
- **exp_range** — a triangular cycle combined with an exponential decay factor, where the maximum learning rate gradually shrinks at each step, controlled by a parameter γ.

IncreaseLROnPlateau This custom scheduler, inspired by PyTorch's widely used `ReduceLROnPlateau`, operates in the opposite direction. While `ReduceLROnPlateau` decreases the learning rate by some predefined factor when the monitored metric (here: accuracy) stops improving, `IncreaseLROnPlateau` starts with a low learning rate and increases it by the factor once the metric plateaus.

Hyperparameter Tuning To determine the optimal hyperparameters for each learning rate scheduler, a grid search was performed. The objective function, defined in Equation 1, quantifies the trade-off between stability and final performance by combining two complementary metrics. The goal is to **maximize knowledge retention** (high final accuracy) while simultaneously **minimizing local instability** (low gap heights). In effect, the objective approximates a **worst-case accuracy** by penalizing temporary drops, and is constructed to be maximized to select the best-performing configuration.

$$\text{Objective} = \text{MeanFinalAcc} - \text{MeanHeight} \tag{1}$$

where: **MeanFinalAcc** is the mean final test accuracy of all three tasks measured after training on Task 3:

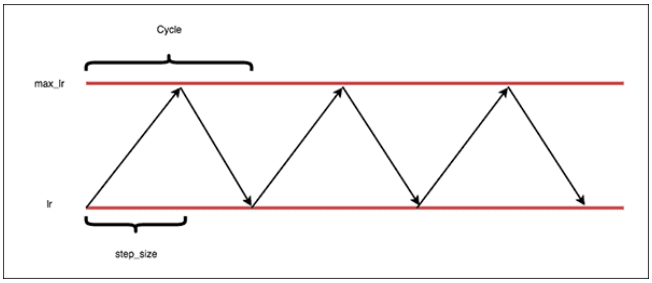
$$\text{MeanFinalAcc} = \frac{1}{3}(A_1 + A_2 + A_3) \tag{2}$$

MeanHeight is the average height of all observed stability gaps:

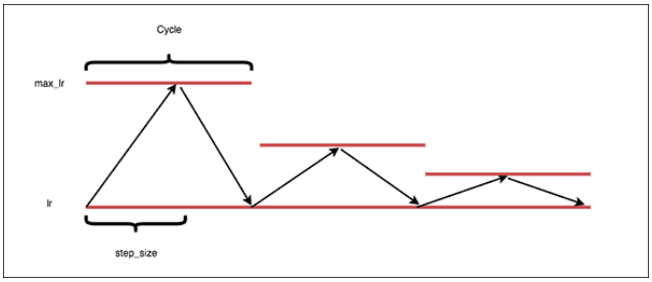
$$\text{MeanHeight} = \frac{1}{3}(H_{1,2} + H_{1,3} + H_{2,3}) \tag{3}$$

where $H_{1,2}$ denotes the height of the drop in Task 1 accuracy after starting Task 2, $H_{1,3}$ after starting Task 3, and $H_{2,3}$ the drop in Task 2 accuracy after starting Task 3.

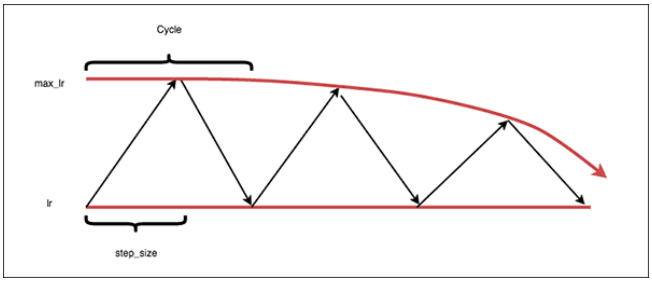
Smoothing the curves Since raw accuracy curves can exhibit high-frequency noise due to the stochastic nature of SGD, **Exponential Moving Average (EMA)** with $\alpha = 0.1$ is applied to smooth each curve before computing the objective value. This smoothing step ensures that the selected hyperparameter configuration reflects the true underlying pattern, rather than being overly influenced by random fluctuations.



(a) Mode "triangular". Keeps constant amplitude.



(b) Mode "triangular2". Amplitude decreases by half each cycle.



(c) Mode "exp_range". Amplitude decreases each cycle, the decrease rate is controlled by parameter γ.

Figure 3: All 3 modes of CyclicLR. Images reproduced from [11]

Evaluation of final configurations To ensure the reliability of the findings, the hyperparameter configurations identified as optimal during grid search were subsequently re-evaluated by performing 20 independent training runs for each configuration. Additionally, all constant learning rates were reassessed using the objective function defined in Equation 1, and the best-performing one was selected as the baseline. This enables a clear comparison to determine whether the schedulers provide any actual improvement over an optimally tuned fixed learning rate.

2.4 Evaluation procedure for constant and scheduled learning rates

The constant learning rate setup is evaluated using the full set of stability gap metrics defined in subsection 2.2. This enables systematic exploration of how varying the constant step size affects both the severity and recovery characteristics of the gap. The goal is to identify trends and trade-offs — for example, whether smaller learning rates consistently yield lower drop heights at the cost of slower recovery, or vice versa.

In contrast, for the scheduled learning rate experiments, the primary goal is not to exhaustively compute all stability gap metrics for each configuration but rather to identify the most promising schedulers using the objective function introduced in Equation 1. Once the top-performing configurations are selected based on this objective, they are further assessed through a combination of visual inspection and the metrics defined in Equation 2 and Equation 3 reported as mean ± standard deviation. This limited set of numerical results provides a consistent reference for comparison, while the detailed metrics are intentionally omitted for schedulers, as they are primarily designed to reveal systematic trends under fixed learning rates rather than to capture the more dynamic behavior introduced by adaptive scheduling.

3 Experimental Setup and Results

This section outlines the complete experimental setup and then presents the results, separately showing the effects of constant learning rates on the stability gap and the performance of the proposed learning rate schedulers.

3.1 Setup

The model is trained using stochastic gradient descent (SGD), with batch size increasing linearly with the number of seen tasks. All other experimental factors - such as architecture, task order, and optimizer - are held constant, so that the learning rate is the only controlled variable. To ensure reliability and account for stochasticity, each configuration was run 20 times with different random seeds. Key training settings are summarized in Appendix A.

Learning rate configurations Two sets of experiments were conducted:

Constant learning rates: 15 values were tested: 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5.

Scheduled learning rates: two schedulers tested as described in section 2. For each scheduler, a grid search over relevant hyperparameters was performed to identify the best configuration. All tested values of the hyperparameters are summarized in Appendix B. In total, 1200 configurations were evaluated for Cyclic LR and 3200 for IncreaseLROnPlateau.

Convergence adjustment For the three lowest constant learning rates $(0.001, 0.005, 0.01)$ the number of training iterations per task was increased from 500 to 2500, to ensure convergence to a stable accuracy level.

Metrics To quantify the stability gap, we compute all metrics based on the per-iteration test accuracy on the first task. This allows us to isolate and analyze the temporary performance drop that occurs when the model begins learning a new task.

Reproducibility A full implementation of the experimental pipeline is publicly available on GitHub[1].

[1] https://github.com/wjssk/learning-rate-in-stability-gap

3.2 Constant Learning Rate — Results

This section focuses on visual inspection of trends and patterns in the stability gap metrics, based on the mean performance across 20 runs. Detailed numerical results for each learning rate, including mean and standard deviation, are provided in Appendix D for reference. Visual representations are prioritized because they offer a clearer overview of the general dynamics and comparative behavior across different learning rates, which may be less apparent from raw tables alone. To further support the analysis, trend lines are overlaid on the plots. They illustrate general tendencies and are not intended as precise predictive models.

General Findings and Visual Inspection As shown in Figure 4, the model trained with a learning rate of 0.001 does not exhibit the typical immediate accuracy drop associated with a stability gap. However, the final accuracy remains low, suggesting that even 2500 training iterations may have been insufficient to reach convergence. In contrast, the configuration with a learning rate of 0.005, which also shows a relatively minor stability gap (Figure 5), achieves a much higher final accuracy despite its small learning rate. Additionally, this setting yields results with low variance across runs, indicating a stable and consistent learning trajectory.

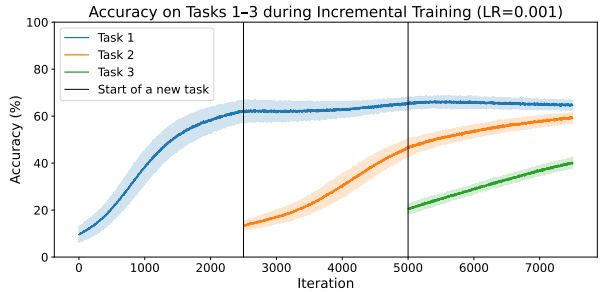


Figure 4: Mean accuracy (%) curves for Task 1 (blue), Task 2 (orange), and Task 3 (green) at LR = 0.001. Shaded areas indicate standard deviation across runs. This configuration exhibits no visible stability gap, but achieves lower overall accuracy compared to higher learning rates.

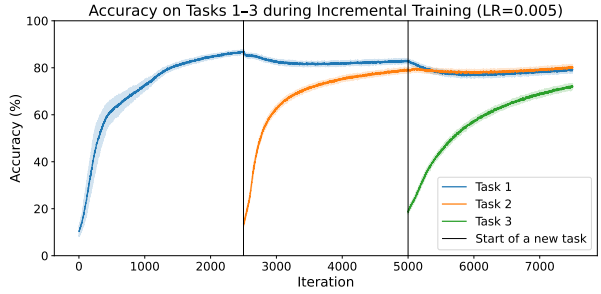
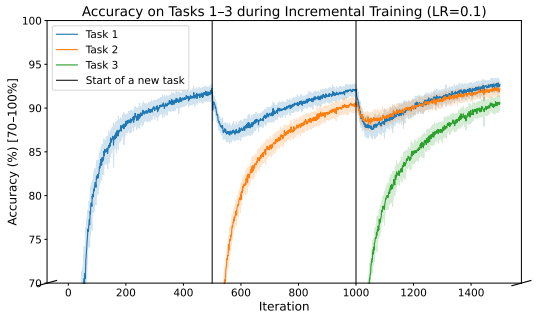
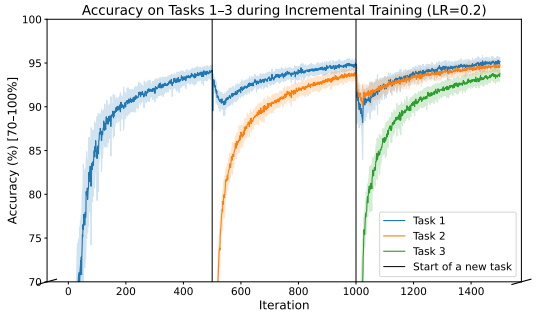


Figure 5: Mean accuracy (%) curves for Task 1 (blue), Task 2 (orange), and Task 3 (green) at LR = 0.005. Shaded areas indicate standard deviation across runs. This configuration exhibits only a minimal stability gap, which is smaller than for higher learning rates, but achieves slightly lower overall accuracy.

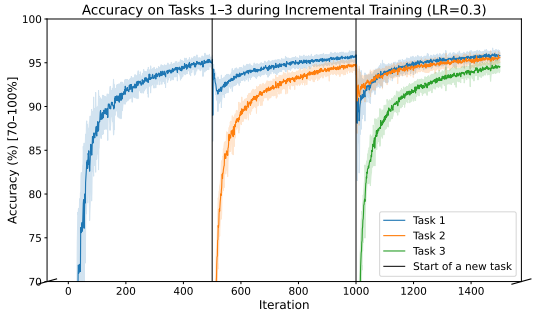
All higher learning rates demonstrate the presence of a stability gap, though its shape and severity vary systematically with the learning rate. As illustrated in Figure 6, increasing the learning rate alters the dynamics of the drop and subsequent recovery. Note that the plots in this figure use a restricted y-axis range (70–100) to better visualize these differences, in contrast to the full-scale range used in the other plots. The figure includes three representative learning rates (0.1, 0.2, 0.3), but plots for all values can be found in Appendix C.



(a) LR = 0.1



(b) LR = 0.2



(c) LR = 0.3

Figure 6: Mean accuracy (%) curves for Task 1 (blue), Task 2 (orange), and Task 3 (green) for selected learning rates. Shaded areas indicate standard deviation across runs. In general, lower learning rates produce a more gradual decline in accuracy with slower recovery, while higher learning rates cause a sharper initial drop but tend to enable faster recovery. Note that the plots in this figure use a restricted y-axis range (70–100).

In general, lower learning rates tend to result in a more gradual decline in accuracy, accompanied by slower recov-

ery. In contrast, higher learning rates cause a sharper initial drop but appear to facilitate faster recovery. These trends, which emerge from visual inspection, will be examined quantitatively in the subsequent sections to assess whether the numerical metrics align with the observed patterns.

It is also worth noting that the stability gap observed for Task 2 (orange curves) closely resembles that of Task 1 (blue curves), suggesting that the model exhibits similar behavior across tasks when it begins training on a new one.

Learning rate 1.5 was also tested, but the resulting accuracy curve displayed highly unstable and erratic behavior. Due to this instability, it is excluded from further quantitative analysis, as its results are not meaningful for interpreting the stability gap phenomena.

Final Accuracy (F-ACC) Figure 7 shows the mean final accuracy for Task 1, Task 2, and Task 3 after completing training on Task 3. Overall, learning rates in the range of 0.1–1.0 achieve consistently high final accuracy (typically above 95%) and demonstrate strong knowledge retention, recovering to the same or even higher accuracy after each task. In contrast, very low learning rates (e.g., 0.001, 0.005) tend to underperform and may fail to converge, especially for later tasks.

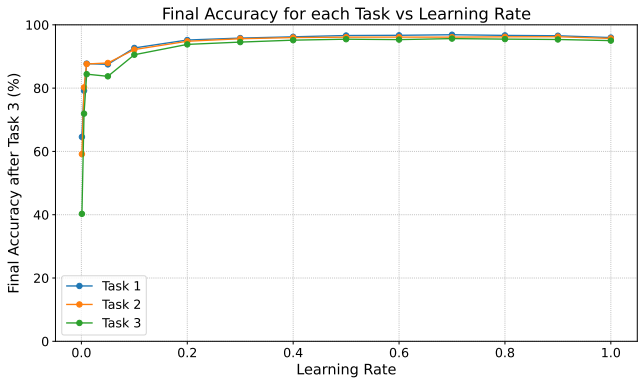


Figure 7: Final Accuracies (%) for all tasks after finishing training on Task 3. All constant LR > 0.1 keep a very similar level of final accuracies, generally above 95%.

Height (H) From this point onwards, all metrics will be calculated for Task 1 only - they will be reported separately for Gap 1 (after starting Task 2) and Gap 2 (after starting Task 3).

Figure 8 shows a clear upward trend: as the learning rate increases, the height of the gap also increases. The second gap tends to be more severe than the first.

Recovery Time (REC-TIME) Figure 9 visualizes the recovery time trends. To enable fair comparison between settings with different total training lengths, recovery times are normalized as a percentage of the full training duration. The curves indicate that recovery generally becomes faster as the learning rate increases, reaching its lowest point around LR values of 0.6–0.7. Beyond this range ($LR > 0.7$), the trend shows a slight upward tendency, suggesting that excessively high learning rates may again hinder the model's ability to regain its pre-drop accuracy quickly.

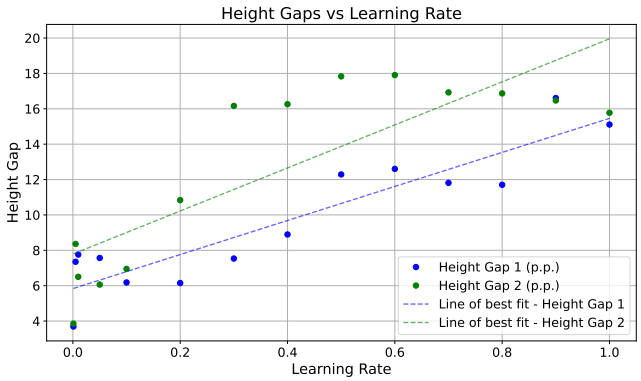


Figure 8: Lines of best fit for height for Gap 1 and Gap 2. A clear upward trend indicates that higher learning rates cause larger gaps, with Gap 2 generally showing greater severity than Gap 1.

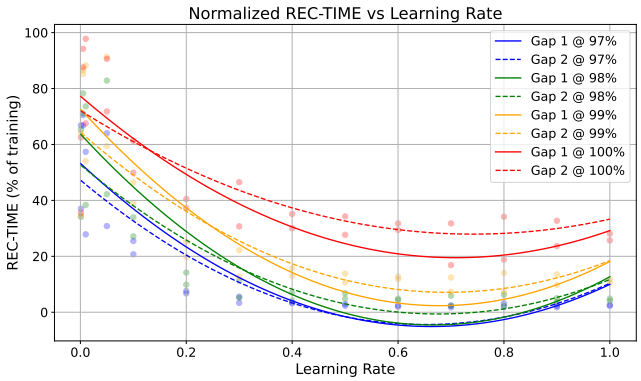


Figure 9: Normalized recovery times for Gap 1 (solid) and Gap 2 (dashed) across thresholds. The recovery speed is increasing with larger learning rates, with a slight decrease beyond 0.7.

Drop Slope (DROP-SLOPE) As illustrated in Figure 10, the descent towards the minimum becomes increasingly steep with larger learning rates.

Knowledge loss (AUC-LOSS) As shown in Figure 11, the overall knowledge loss, measured as the ratio of areas under the curve (AUC-LOSS), remains relatively low across most learning rates. Interestingly, a small peak is observed for learning rates between 0.005 and 0.05, where the loss gap reaches its highest values (around 0.05), indicating a modest degradation in performance. For higher learning rates, particularly in the range of 0.6 to 0.8, the knowledge loss reaches a minimum, suggesting that the model retains knowledge more effectively across task transitions.

Notably, for LR = 0.001, both loss gap values are slightly negative. This could indicate that the model was still in the process of converging and continued to improve even after switching tasks, rather than exhibiting forgetting. However, this effect is likely due to undertraining rather than true knowledge retention.

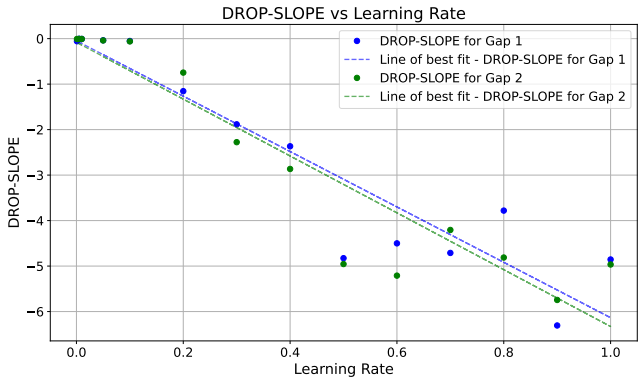


Figure 10: Lines of best fit for drop slopes for Gap 1 and Gap 2. Steeper slopes are observed at higher learning rates, indicating faster accuracy decline immediately after task switches.

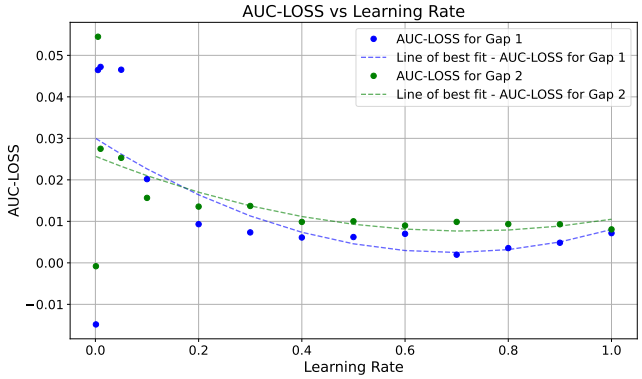


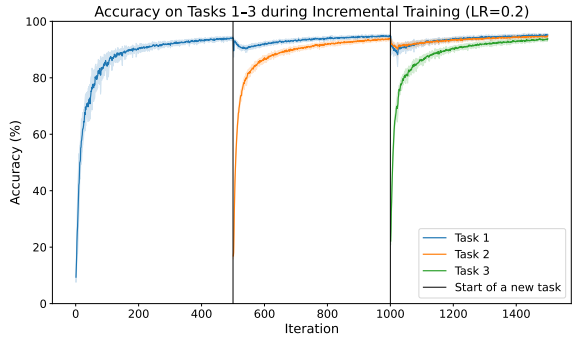
Figure 11: Lines of best fit for knowledge loss (AUC-LOSS) for Gap 1 and Gap 2. Loss remains low across most learning rates, with a slight peak at low LRs (0.005–0.05) and minimum values for moderate-to-high LRs (0.6–0.8).

3.3 Scheduled Learning Rates

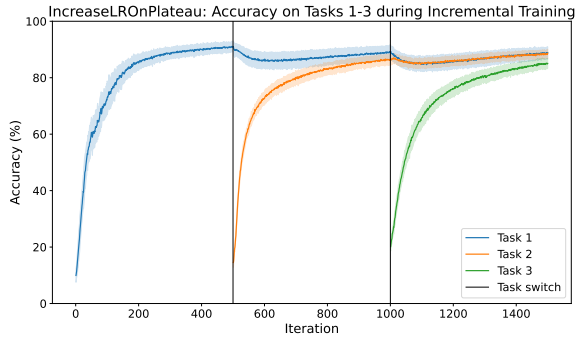
Figure 12 shows the mean accuracy curves for the baseline constant learning rate (LR $= 0.2$), the IncreaseLROnPlateau scheduler, and the best-performing CyclicLR configuration (Triangular2). All three represent the top configurations identified using the objective function defined in Equation 1 for each approach.

Although all CyclicLR modes (Triangular, Triangular2, Exp_Range) were tested separately, for all of them the optimal cycle length was consistently found to be 500 iterations (with an optimal step size of 250). This resulted in exactly one full cycle per task, which means that the specific mode had little impact on the resulting schedule shape — the main differences between modes, such as how they adjust the amplitude across multiple cycles, could not manifest with only a single cycle. Consequently, all CyclicLR modes produced nearly identical accuracy curves and objective values. For clarity, only the curve for Triangular2 (which achieved the highest objective value) is shown here.

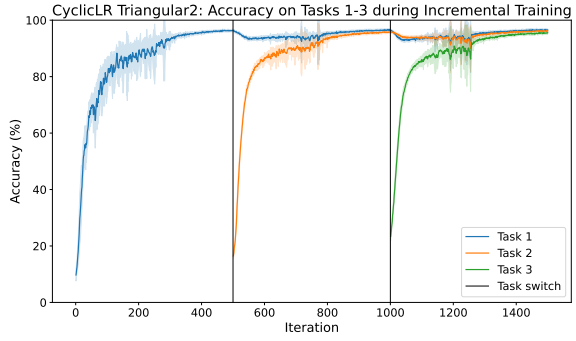
Visual inspection of Figure 12 reveals that while CyclicLR effectively maintains high overall accuracy, it introduces pronounced high-frequency fluctuations in the middle of each task's training. These oscillations do not represent the stability gap as defined earlier, which specifically refers to the immediate drop in accuracy occurring directly after switching tasks. Because this study focuses on analyzing the initial drop rather than later within-task oscillations, the reported gap height for CyclicLR in Table 1 was recalculated using only the first 100 iterations following each task switch, instead of the entire window. This narrower range isolates the part of the curve that truly represents the defined stability gap. However, it should be noted that these local fluctuations re-



(a) Baseline Constant LR = 0.2



(b) IncreaseLROnPlateau



(c) CyclicLR Triangular2

Figure 12: Mean accuracy (%) with standard deviation (shaded area) for Task 1 (blue), Task 2 (orange), and Task 3 (green). Each plot shows the configuration that maximized the objective function from Equation 1, selected via grid search for each method. Each configuration was evaluated over 20 independent runs.

main present and visible in the curves and could still affect the overall training stability in practice. Thus, while CyclicLR achieves the lowest measured drop height under this stricter definition, its tendency to produce internal fluctuations highlights a trade-off between minimizing the immediate gap and ensuring smooth convergence throughout training. Table 1 summarizes the key numerical results for each method, combining this refined height measurement with the final mean accuracy across tasks.

Table 1: Summary of the MeanFinalAcc defined in Equation 2 and MeanHeight defined in Equation 3 for each approach. For CyclicLR, the gap height is computed only over the first 100 iterations after each task switch, to better isolate the initial drop from the scheduler's further oscillations. Fluctuations occurring later are not included in the height value but are clearly visible in the accuracy plots. All values are reported as mean ± standard deviation across 20 runs. The best values are highlighted in bold.

Scheduler	MeanFinalAcc ↑	MeanHeight ↓
IncreaseLROnPlateau	87.54 ± 2.49	5.33 ± 0.70
CyclicLR Triangular2	**96.08 ± 0.57**	**4.39 ± 0.47**
Constant LR = 0.2	94.61 ± 0.72	7.75 ± 1.40

4 Discussion and Future Work

This study systematically examined how the learning rate influences the stability gap in continual learning and whether scheduling can help mitigate this effect. For constant learning rates, the results highlight a clear trade-off: smaller step sizes reduce the immediate drop in accuracy at task switches but slow down recovery, while larger step sizes speed up convergence but lead to a more pronounced gap. Furthermore, across nearly all learning rate values, the second gap for Task 1 (so after starting Task 3) tended to be more severe than the first, suggesting that cumulative effects may compound as more tasks are added.

For the tested schedulers, no configuration clearly outperformed the others in all respects. CyclicLR achieved the lowest measured gap height when the measurement was deliberately restricted to the first 100 iterations after each switch, isolating the true initial drop from later oscillations. However, visual inspection shows that CyclicLR introduces pronounced fluctuations within each cycle, which undermines its practical stability — especially when considering worst-case performance; this makes it less suitable for safety-critical applications (e.g., medical applications or self-driving cars) where this aspect is crucial. IncreaseLROnPlateau achieved a smaller average gap than the constant learning rate but did not reach comparable final accuracy, and it too showed some variability across runs.

Overall, a well-chosen constant learning rate ($\mathrm{LR} = 0.2$) offered the most robust trade-off in this controlled setting, combining high final accuracy with a moderate gap and no intra-task oscillations. This suggests that while adaptive schedulers hold promise in theory, their practical benefit may depend critically on careful tuning and sufficient training time, which should be factored into deployment decisions for real-world continual learning systems.

Practical Implications The findings of this study provide guidance for selecting learning rates in continual learning scenarios. For applications where worst-case performance must remain stable — such as medical systems or autonomous vehicles — it may be safer to favor lower learning rates that yield smaller and more predictable gap heights, even at the cost of slower recovery. In less critical domains, where fast adaptation is more valuable than strict stability, higher learning rates may be preferable. Additionally, practitioners with sufficient computational resources may benefit from experimenting with scheduler tuning to achieve more dynamic control over the stability gap.

Limitations This work is subject to some limitations. Due to computational constraints, both the training time per task (500 iterations) and the scheduler grid search size were relatively limited. In particular, the short cycle length may have prevented CyclicLR from fully stabilizing, exaggerating fluctuations near the maximum learning rate. Additionally, the custom gap metrics defined here, although motivated by clear visual patterns, are not yet standardized in the continual learning literature, which may limit their general applicability. Similarly, the objective function used to select the best scheduler configurations was based on these same custom metrics and does not follow any established framework, so it may not fully capture the best possible trade-offs between stability and performance.

Future Work Future research should apply longer training times, test a wider variety of scheduling strategies, and adopt more advanced hyperparameter optimization techniques to refine and validate these preliminary insights. Experiments involving more tasks could further clarify whether the trend of increasingly severe gaps continues and how quickly this effect grows with task count. Finally, verifying these observations on different datasets and model architectures would help establish whether the patterns found here generalize to more complex and practical continual learning scenarios.

5 Conclusion

This study set out to clarify how the learning rate affects the stability gap in continual learning and whether dynamic scheduling can help reduce it. The results confirm that smaller learning rates limit the immediate drop but slow the recovery and risk underperformance, while larger rates enable faster learning but at the cost of a more pronounced gap. Scheduling shows promise for improving this balance but requires careful tuning to avoid introducing new instability.

Together, these findings partially support both initial hypotheses and provide practical insights into the role of the learning rate in shaping the stability gap. By isolating and quantifying these effects, this work offers a solid foundation for future research on better-tuned schedulers and provides insights for making more informed choices when adjusting learning rates for stable continual learning.

6 Responsible Research

This study focuses on a fundamental aspect of continual learning and does not directly involve human participants,

sensitive data, or real-world deployment, so it does not pose any immediate ethical risks. However, understanding and mitigating the stability gap might have important implications for the security and reliability of AI systems in safety-critical applications. In domains such as autonomous driving or healthcare, sudden drops in model performance could lead to harmful decisions. By exploring how learning rate configurations affect this phenomenon, this research contributes to developing more robust and trustworthy continual learning algorithms.

All experiments were conducted using publicly available datasets (Rotated MNIST) under a permissive license. The complete implementation, including training scripts, hyperparameter configurations, and the code for computing the stability gap metrics, is openly shared on GitHub to support full reproducibility. This ensures that other researchers can replicate the results, verify the findings, and build upon this work to advance the field responsibly.

Use of Large Language Models (LLMs) To support the writing process, a Large Language Model (ChatGPT by OpenAI) was used as a tool for brainstorming and rephrasing sentences for clarity and coherence. All content generated by the LLM was critically reviewed, verified, and revised to ensure correctness and appropriateness for this project. Example prompts and interactions with the LLM are provided in Appendix F.

References

[1] Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. *arXiv preprint arXiv:2205.13452*, 2022.

[2] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[3] Timm Hess, Tinne Tuytelaars, and Gido M van de Ven. Two complementary perspectives to continual learning: Ask not only what to optimize, but also how. *arXiv preprint arXiv:2311.04898*, 2023.

[4] Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37:111760–111801, 2024.

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

[6] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[7] 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.

[8] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[9] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[10] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[11] Adrian Rosebrock. Cyclical learning rates with keras and deep learning. https://pyimagesearch.com/2019/07/29/cyclical-learning-rates-with-keras-and-deep-learning/, 2019. Accessed: 2024-06-17.

[12] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

[13] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[14] Gido M Van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

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

[16] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

[17] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

A Key training settings used for all experiments

Table 2: Training configuration used for all experiments

Parameter	Value
Dataset	Rotated MNIST
Tasks	3 (0°, 80°, 160°)
Learning scenario	Domain-incremental
Training scheme	Incremental joint training
Optimizer	SGD with no momentum
Batch size	128× number of seen tasks
Iterations per task	500 (2500 for constant LR = 0.001, 0.005, 0.01)
Evaluation frequency	After every training iteration
Test set size	2000 examples

B Search Space for Grid Search for Scheduled LR

This appendix provides a detailed overview of the hyperparameters and their tested ranges for each learning rate scheduler evaluated in this study. The chosen hyperparameters represent the most fundamental factors that directly control the core behavior of each scheduler.

B.1 CyclicLR (Triangular, Triangular2, Exp_Range)

The CyclicLR scheduler was tested in all three standard modes available in PyTorch: `triangular`, `triangular2`, and `exp_range`. In all cases, momentum cycling was explicitly disabled (`cycle_momentum = False`), and all other parameters were kept at PyTorch's default values.

Table 3: Grid search parameters for CyclicLR (all modes)

Parameter	Values
`min_lr`	{0.001, 0.005, 0.01, 0.05}
`max_lr`	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
`step_size`	{250, 125, 100, 50, 1}
`gamma` (Exp_Range only)	{0.9999, 0.9995, 0.99, 0.98}

B.2 IncreaseLROnPlateau

The custom `IncreaseLROnPlateau` scheduler was designed analogously to PyTorch's standard `ReduceLROnPlateau`, but operates in the opposite direction by increasing the learning rate when the monitored loss plateaus. The grid search varied the increase factor, patience, and threshold for plateau detection, together with the minimum and maximum learning rates.

Table 4: Grid search parameters for IncreaseLROnPlateau

Parameter	Values
`min_lr`	{0.001, 0.005, 0.01, 0.05}
`max_lr`	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
`factor`	{1.1, 1.2, 1.3, 1.4, 1.5}
`patience`	{5, 10, 15, 20}
`threshold`	{1e-2, 1e-3, 1e-4, 1e-5}

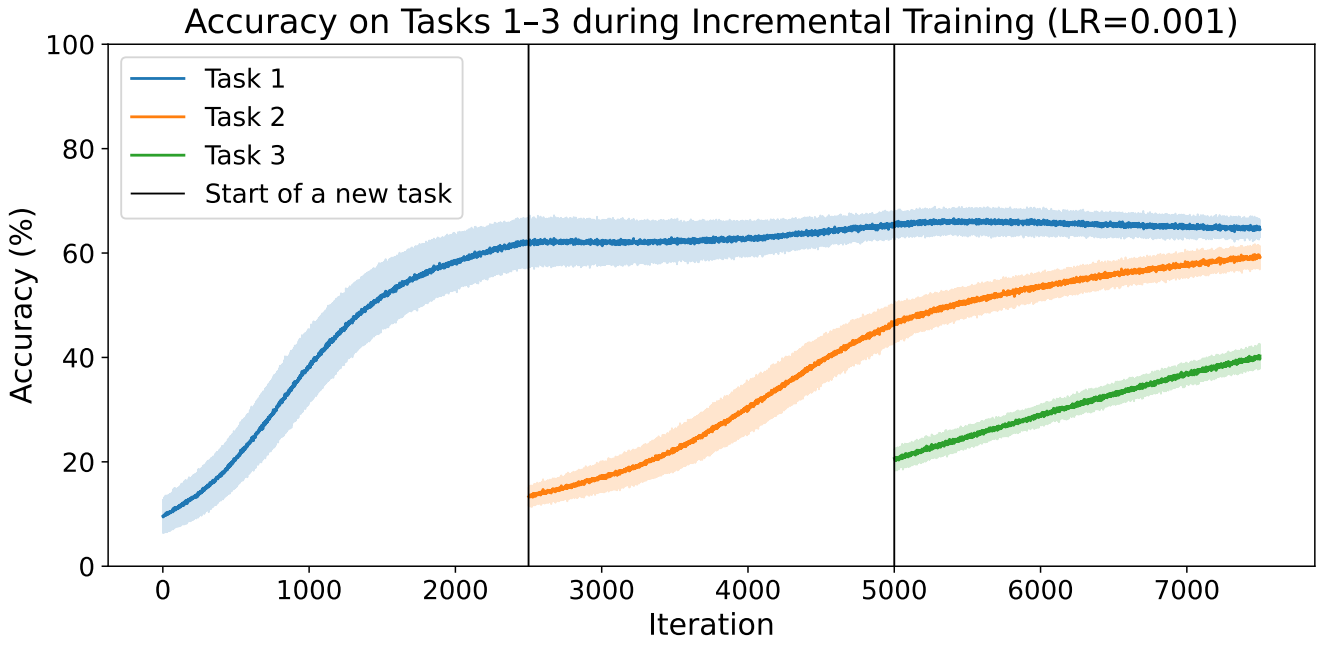
B.3 Constant Learning Rates (Baseline)

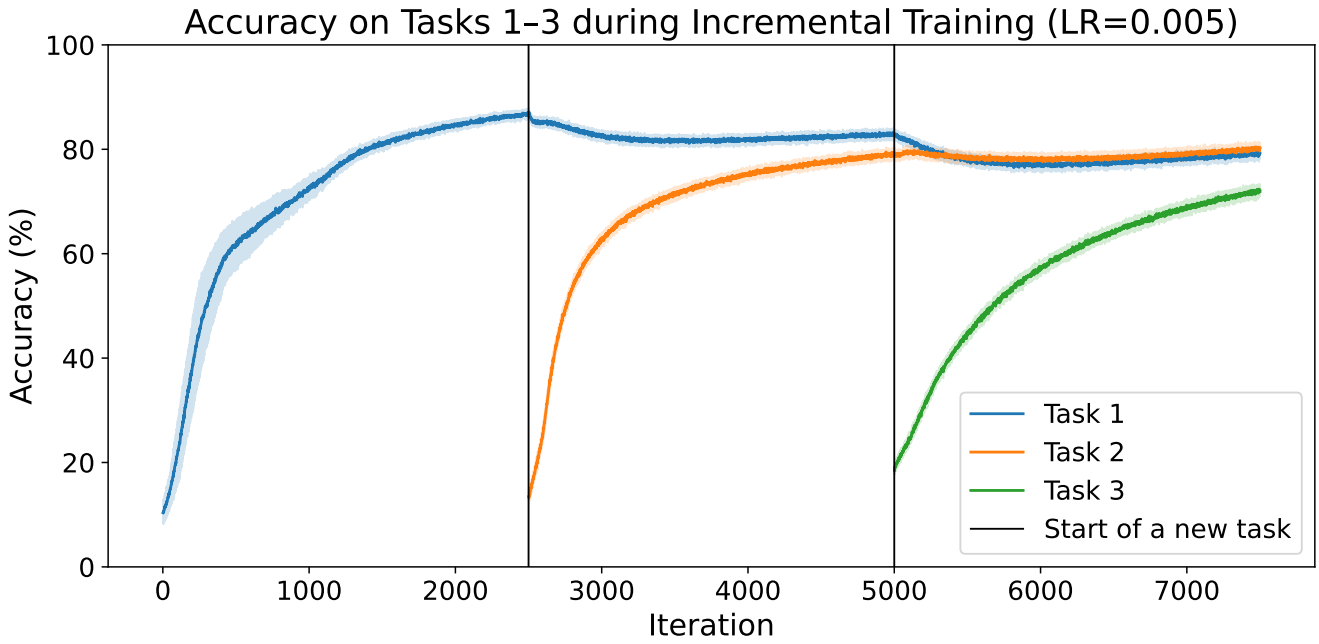
For reference, the following set of constant learning rates was evaluated as a baseline. These values span a wide range from very small to large step sizes to fully map their influence on the stability gap:

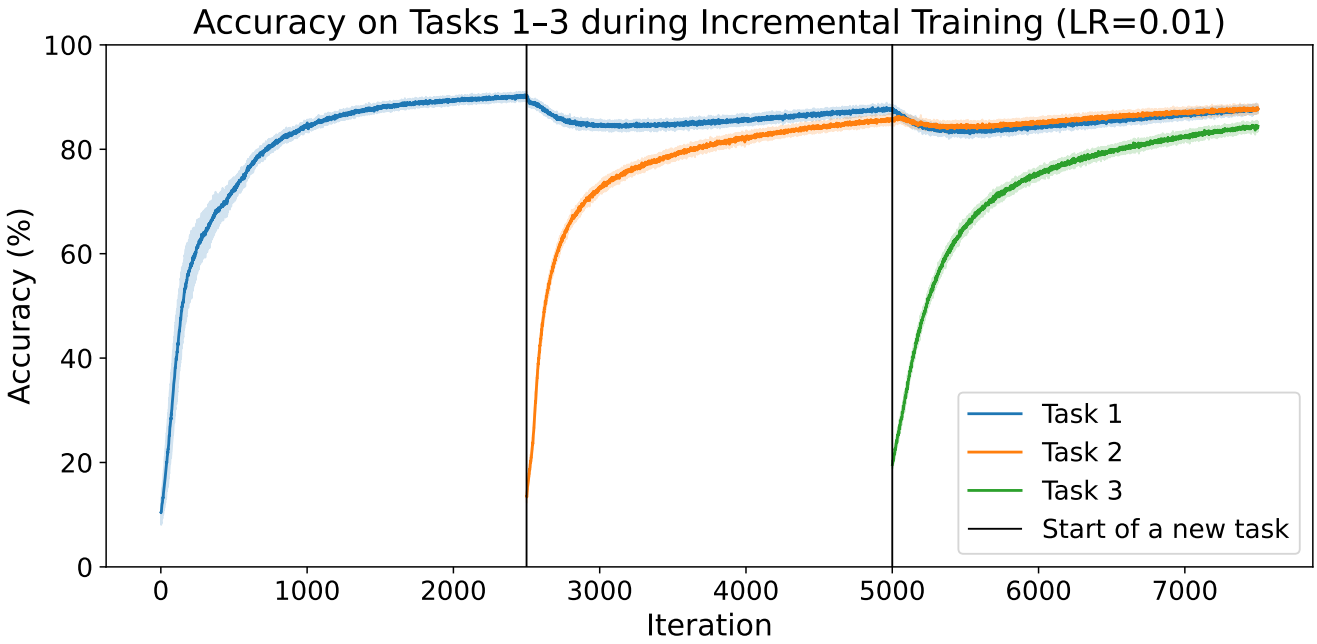
$$\{0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5\}.$$

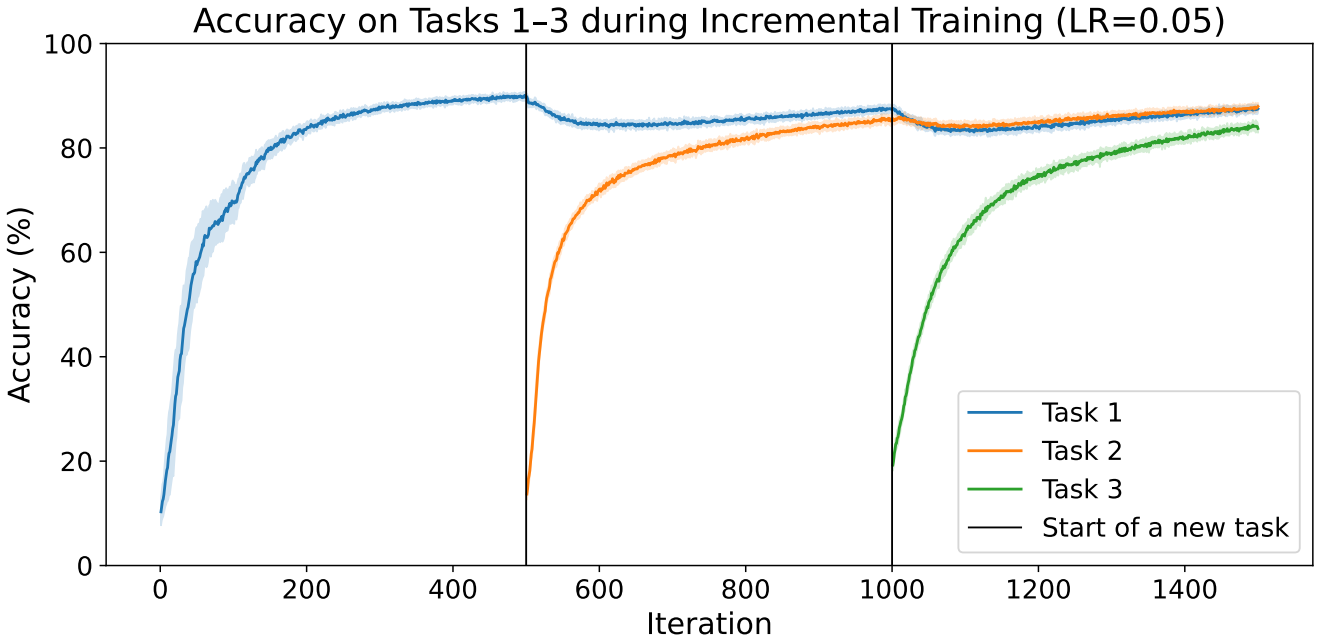
Motivation Only the core hyperparameters directly responsible for controlling the fundamental cycle shape, amplitude, and frequency (for CyclicLR) and the increase pattern (for IncreaseLROnPlateau) were included in the search. This design choice isolates their principal effect on training dynamics while keeping the search computationally feasible and the results interpretable.

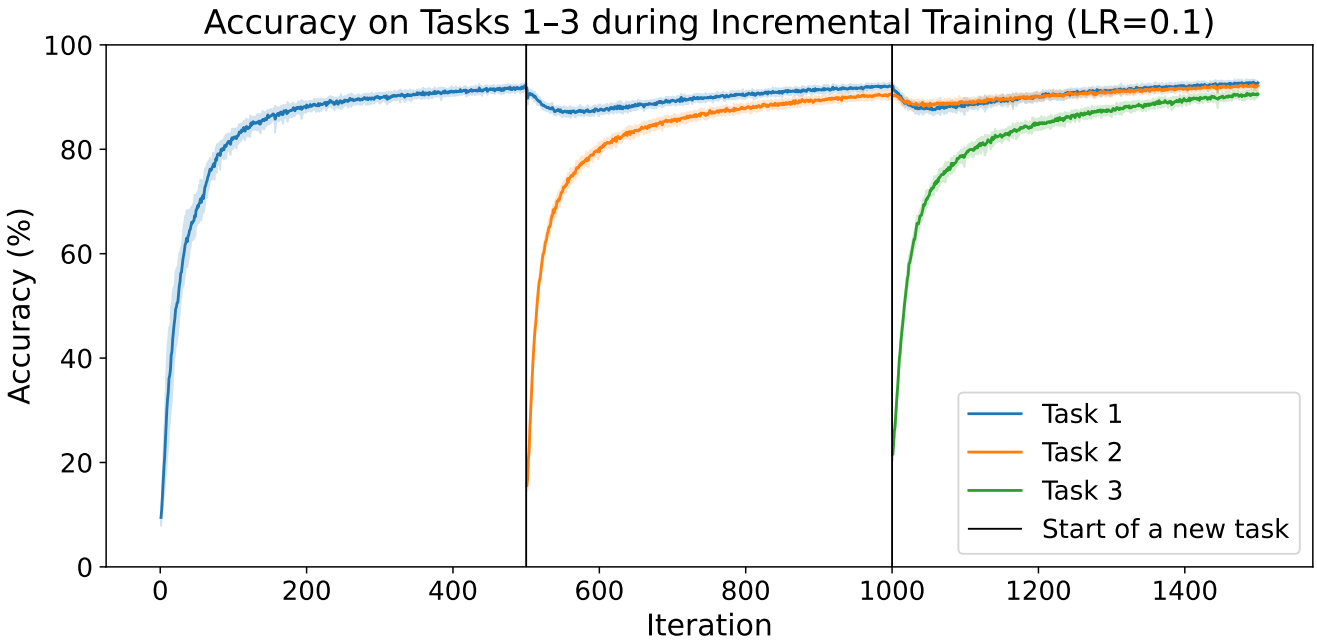
C All accuracy plots for Constant Learning Rates

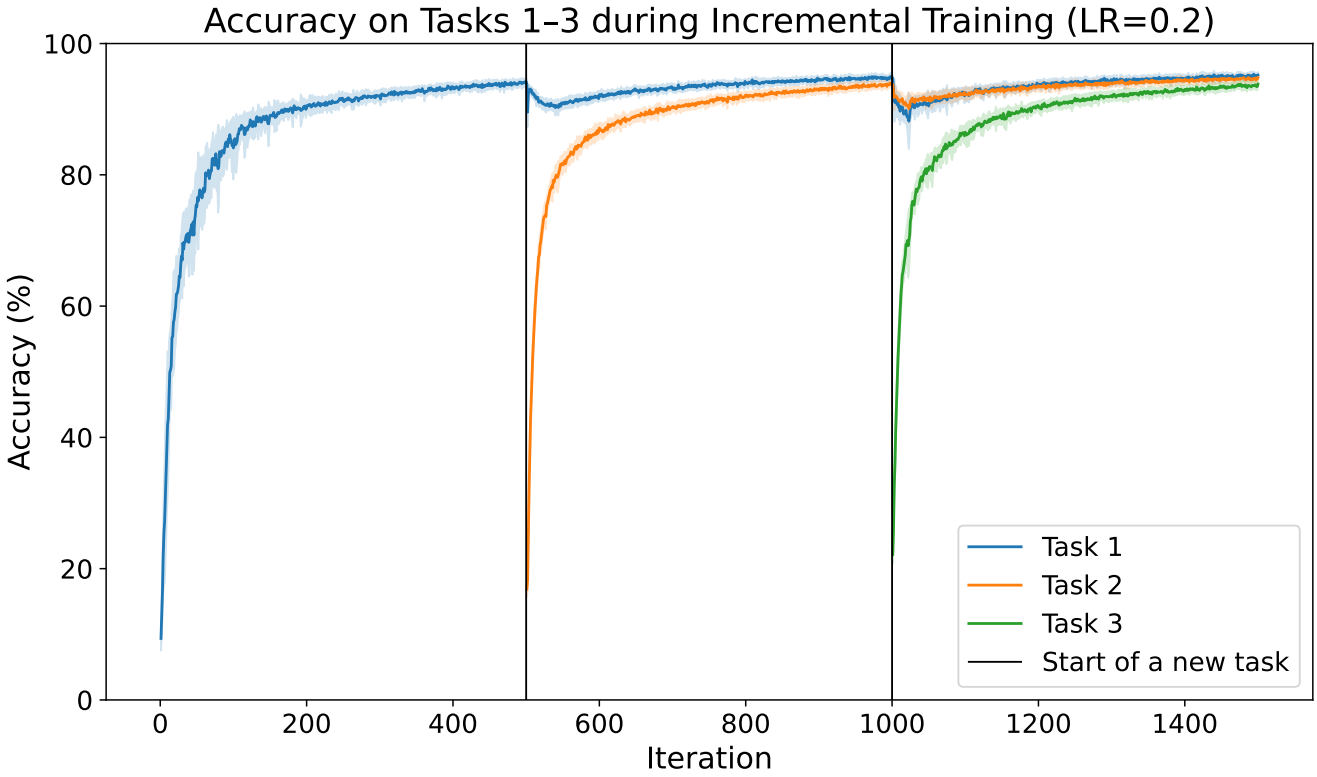


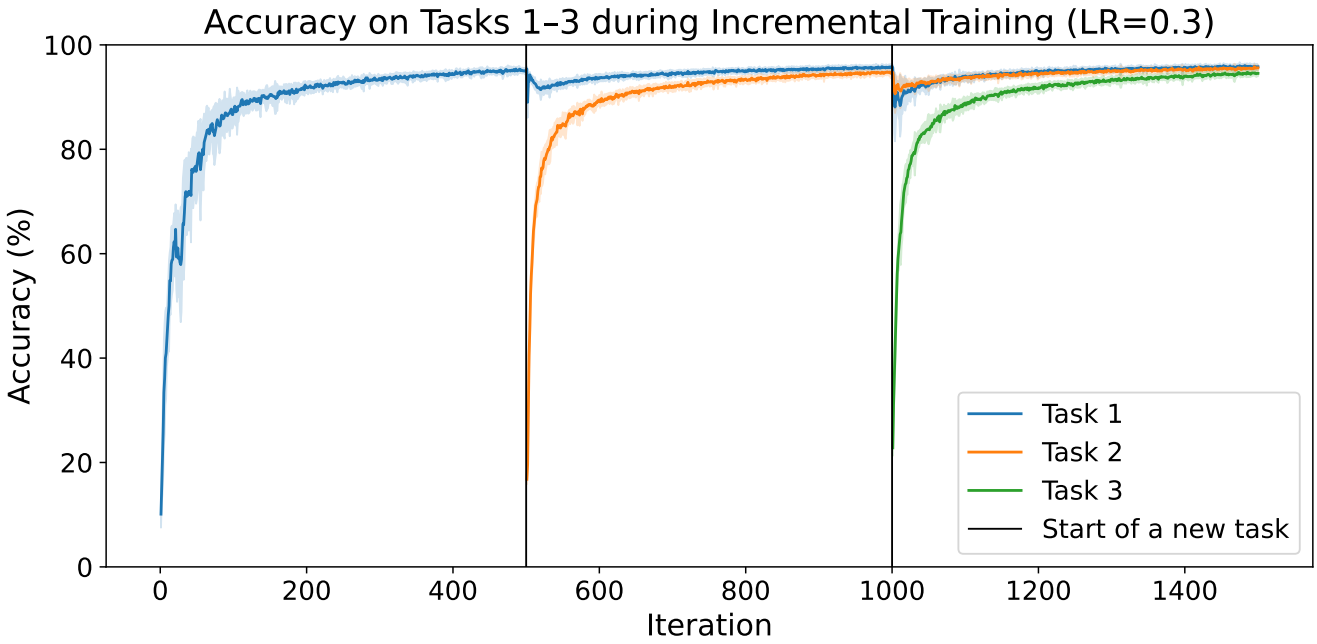


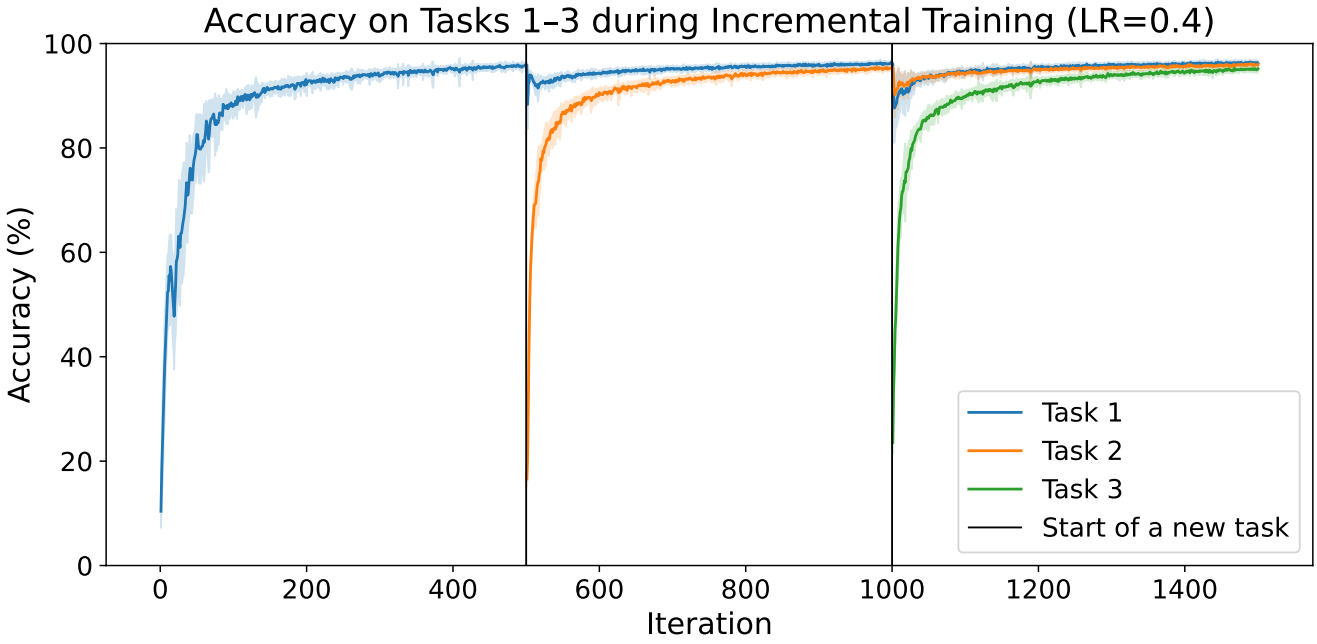

Accuracy on Tasks 1–3 during Incremental Training (LR=0.01)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
6000
7000
Task 1
Task 2
Task 3
Start of a new task

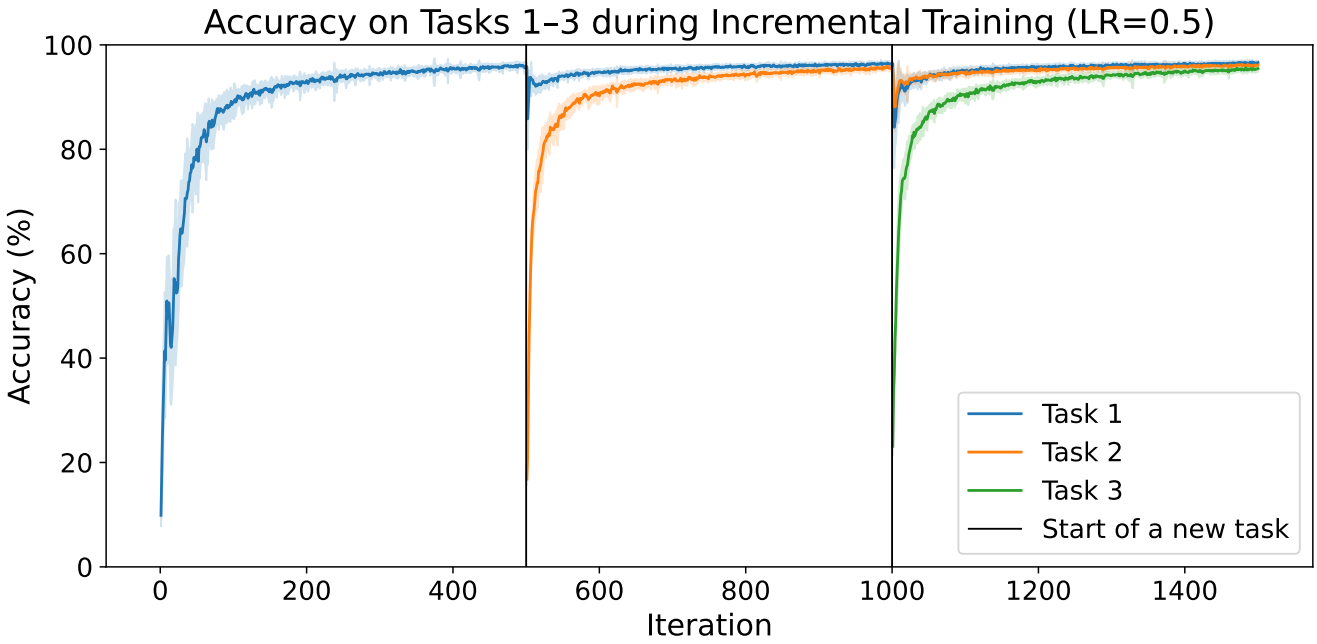

Accuracy on Tasks 1–3 during Incremental Training (LR=0.05)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

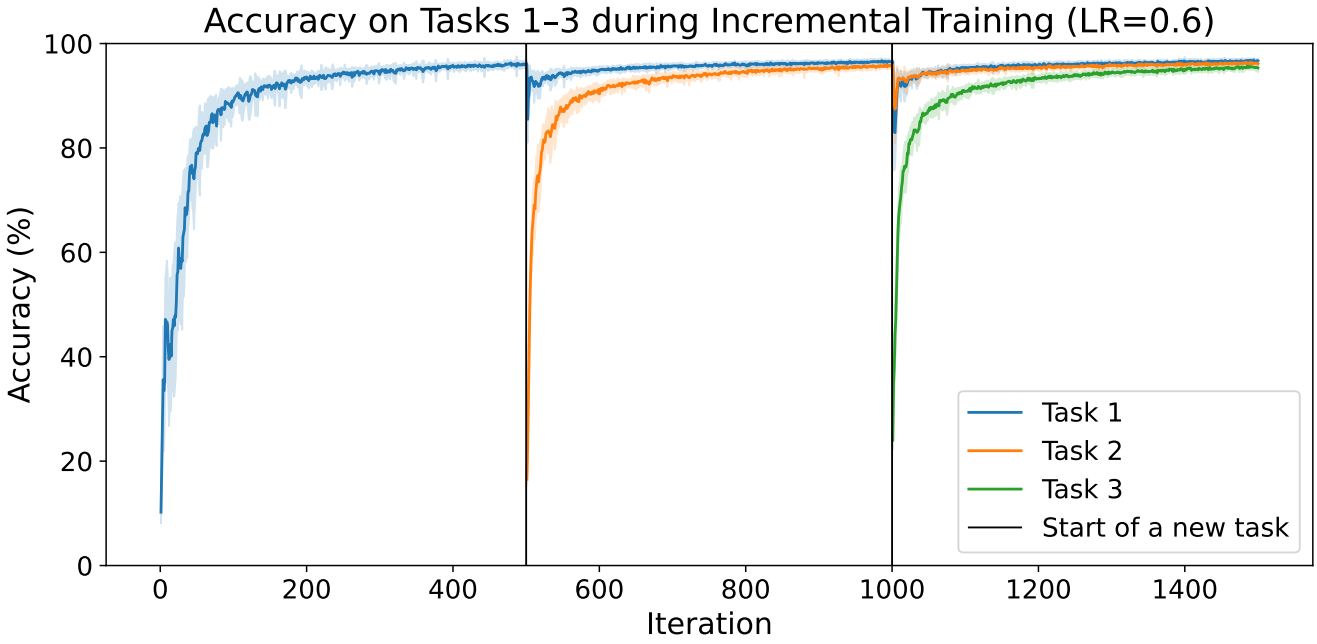

Accuracy on Tasks 1–3 during Incremental Training (LR=0.1)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

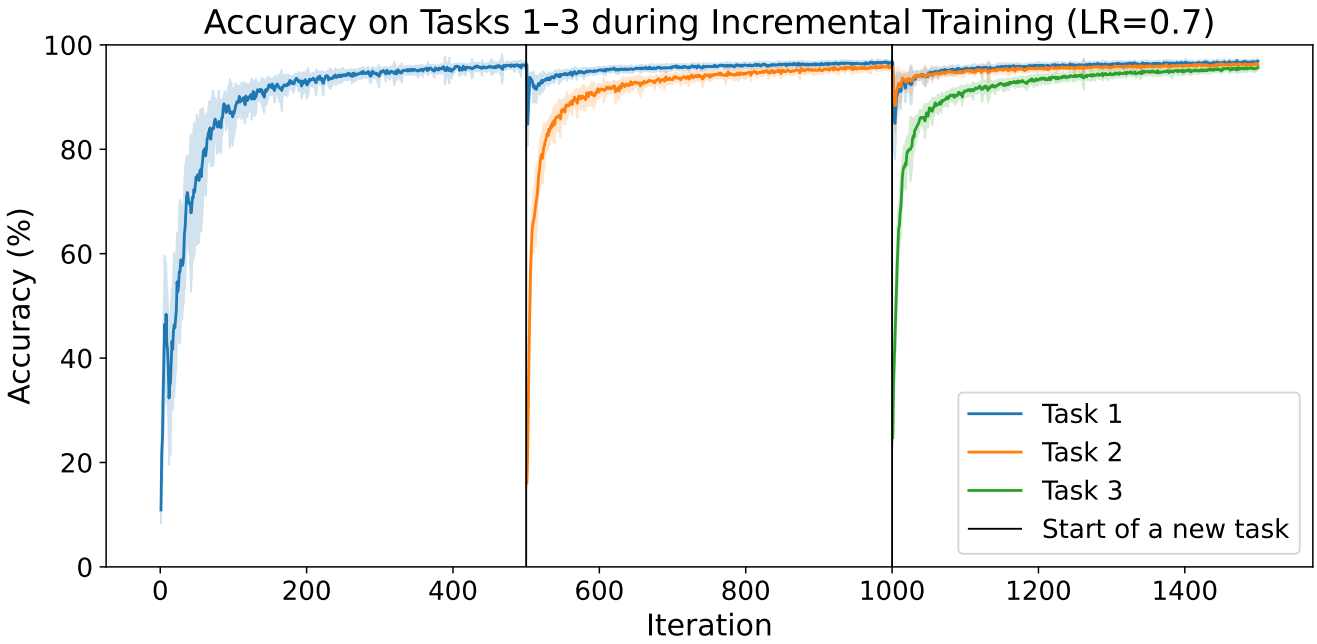

Accuracy on Tasks 1–3 during Incremental Training (LR=0.2)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

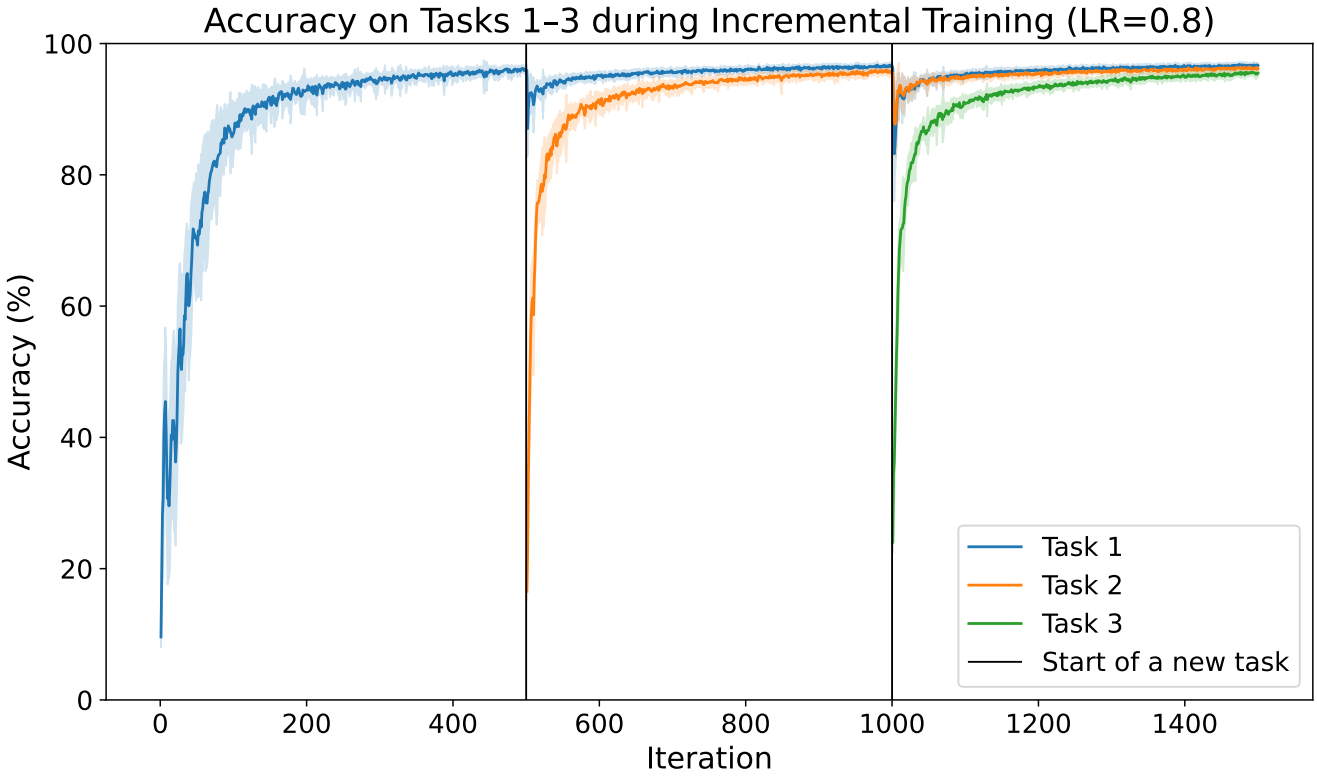

Accuracy on Tasks 1–3 during Incremental Training (LR=0.3)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

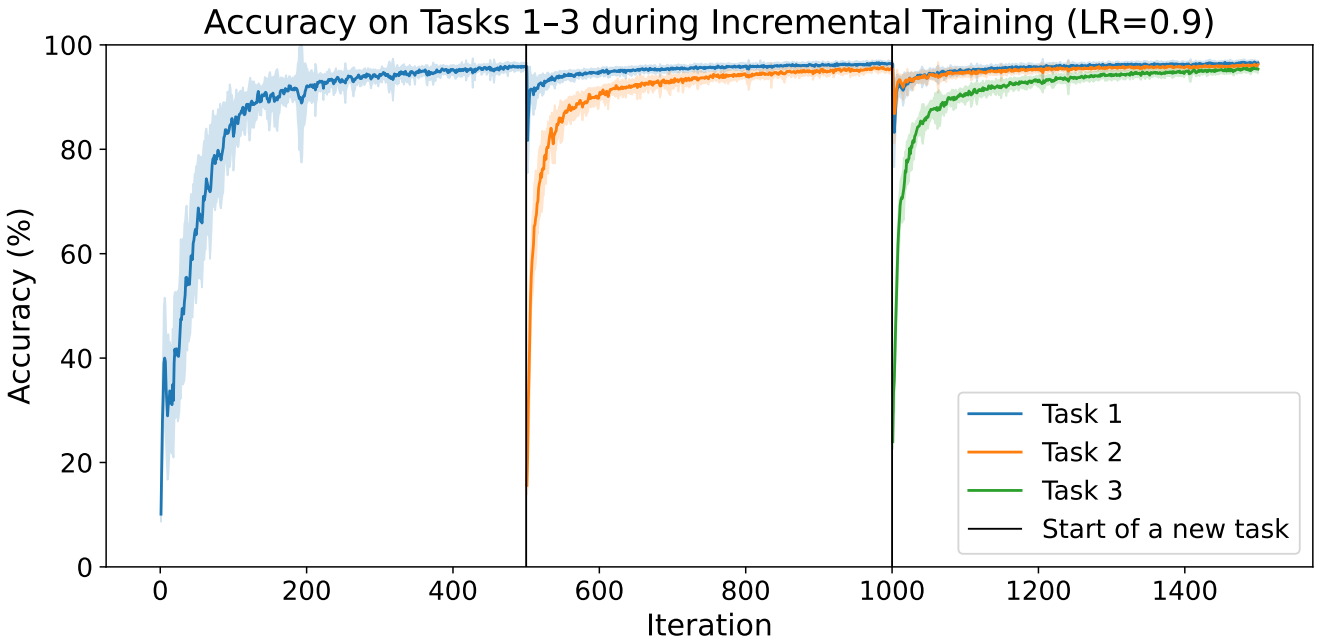

Accuracy on Tasks 1–3 during Incremental Training (LR=0.4)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

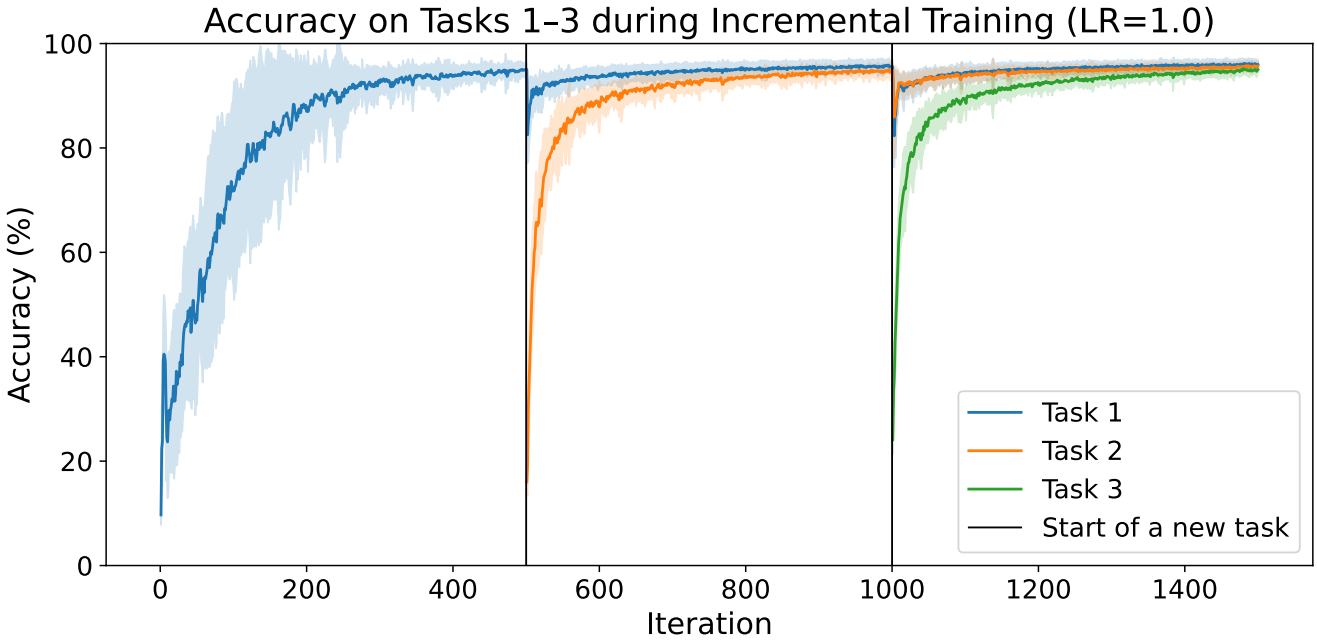

Accuracy on Tasks 1–3 during Incremental Training (LR=0.5)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task

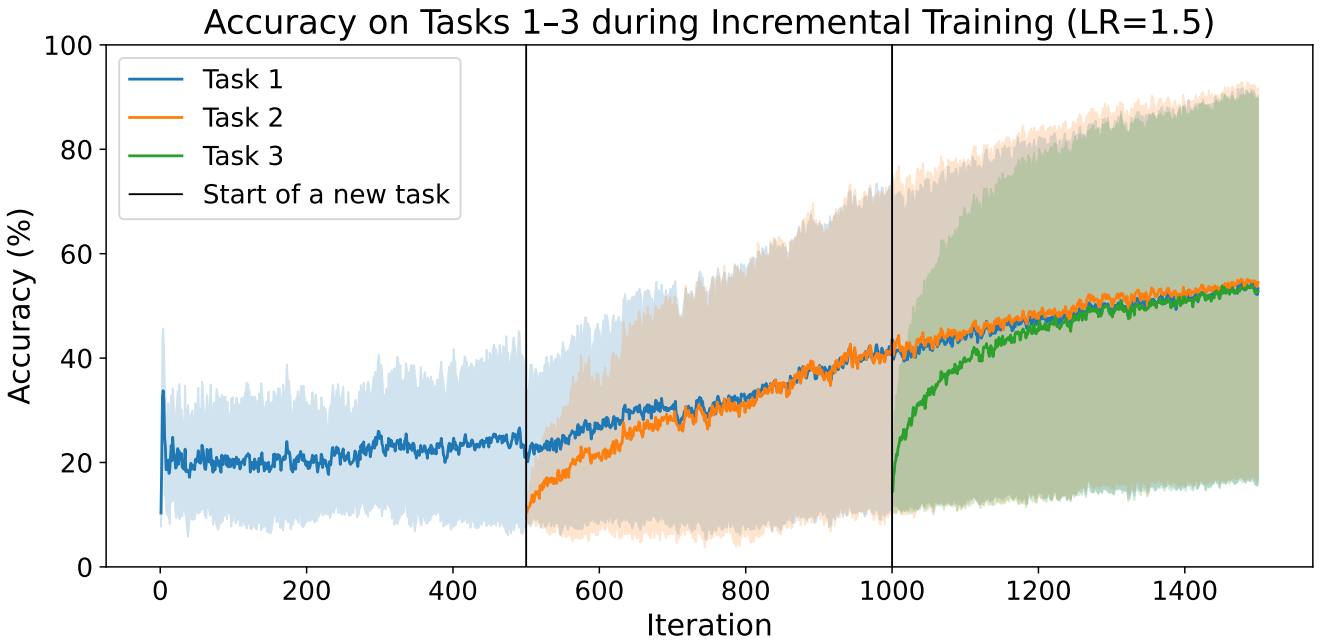

Accuracy on Tasks 1–3 during Incremental Training (LR=0.6)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task


Accuracy on Tasks 1–3 during Incremental Training (LR=0.7)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task


Accuracy on Tasks 1–3 during Incremental Training (LR=0.8)
Accuracy (%)
Iteration
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400
Task 1
Task 2
Task 3
Start of a new task


Accuracy on Tasks 1–3 during Incremental Training (LR=0.9)
Accuracy (%)
Iteration
Task 1
Task 2
Task 3
Start of a new task


Accuracy on Tasks 1–3 during Incremental Training (LR=1.0)
Accuracy (%)
Iteration
Task 1
Task 2
Task 3
Start of a new task



D Constant LR Metrics - All Tables with Raw Values

Final Accuracy (F-ACC)

Table 5: Final accuracy (%) for each task measured after subsequent task training. Values are reported as mean ± standard deviation across 20 runs.

F-ACC of Task		Task 1			Task 2		Task 3
After task	T1	T2	T3	T2	T3	T3
Learning rate						
0.001	62.1 ± 4.0	65.5 ± 1.8	64.6 ± 1.5	46.8 ± 3.0	59.2 ± 1.7	40.3 ± 1.9
0.005	86.5 ± 0.7	82.7 ± 1.1	79.2 ± 0.9	79.3 ± 1.1	80.2 ± 0.6	71.9 ± 0.9
0.01	90.1 ± 0.6	87.7 ± 0.7	87.7 ± 0.6	85.5 ± 1.0	87.6 ± 0.7	84.4 ± 0.8
0.05	89.9 ± 0.6	87.4 ± 0.7	87.5 ± 0.8	85.2 ± 0.6	87.9 ± 0.7	83.7 ± 0.8
0.1	91.7 ± 0.7	92.0 ± 0.7	92.7 ± 0.3	90.4 ± 0.6	92.2 ± 0.4	90.5 ± 0.6
0.2	94.2 ± 0.6	94.9 ± 0.5	95.2 ± 0.5	93.7 ± 0.5	94.8 ± 0.4	93.8 ± 0.4
0.3	95.2 ± 0.5	95.9 ± 0.5	95.8 ± 0.3	94.8 ± 0.5	95.6 ± 0.6	94.5 ± 0.5
0.4	95.6 ± 0.7	96.1 ± 0.5	96.2 ± 0.6	95.4 ± 0.4	96.0 ± 0.5	95.2 ± 0.4
0.5	95.9 ± 0.6	96.4 ± 0.4	96.6 ± 0.4	95.5 ± 0.6	96.1 ± 0.4	95.5 ± 0.7
0.6	96.2 ± 0.5	96.5 ± 0.4	96.7 ± 0.4	95.9 ± 0.6	96.1 ± 0.4	95.3 ± 0.8
0.7	95.8 ± 0.7	96.7 ± 0.4	96.9 ± 0.4	95.9 ± 0.5	96.1 ± 0.3	95.6 ± 0.5
0.8	95.9 ± 0.5	96.6 ± 0.4	96.7 ± 0.3	95.5 ± 0.8	96.2 ± 0.4	95.5 ± 0.6
0.9	95.8 ± 0.6	96.4 ± 0.5	96.6 ± 0.4	95.6 ± 0.5	96.2 ± 0.4	95.4 ± 0.8
1.0	95.1 ± 1.5	95.7 ± 1.1	96.0 ± 0.9	94.7 ± 1.5	95.7 ± 0.7	95.0 ± 0.7

Height (H)

Table 6: Mean heights for Gap 1 and Gap 2 reported as mean ± standard deviation across 20 runs.

Learning rate	Height Gap 1	Height Gap 2
0.001	3.7 ± 1.8	3.9 ± 1.3
0.005	7.4 ± 0.8	8.4 ± 1.3
0.01	7.8 ± 0.6	6.5 ± 0.7
0.05	7.6 ± 0.8	6.1 ± 0.7
0.1	6.2 ± 0.8	7.0 ± 1.5
0.2	6.1 ± 1.2	10.8 ± 3.1
0.3	7.5 ± 2.8	16.2 ± 5.2
0.4	8.9 ± 3.8	16.3 ± 4.7
0.5	12.3 ± 4.8	17.8 ± 5.7
0.6	12.6 ± 3.7	17.9 ± 5.8
0.7	11.8 ± 3.6	16.9 ± 4.9
0.8	11.7 ± 3.8	16.9 ± 5.9
0.9	16.6 ± 5.7	16.5 ± 4.7
1.0	15.1 ± 4.6	15.8 ± 6.3

Recovery Time (REC-TIME)

Table 7: Number of iterations needed to recover for the selected thresholds for Gap 1. Values are reported as mean ± standard deviation across 20 runs. Boldface highlights rows corresponding to learning rates of 0.001, 0.005, and 0.01, which were trained for 2500 iterations instead of 500, to clarify the higher recovery counts. A dash ("–") indicates that fewer than 85% of runs (i.e., fewer than 17 out of 20) reached the specified threshold, and thus the value is not reported.

Threshold Learning rate	REC-TIME 100%	REC-TIME 99%	REC-TIME 98%	REC-TIME 97%
0.001	**889.9 ± 720.4**	**864.8 ± 688.7**	**852.8 ± 675.1**	**926.2 ± 743.9**
0.005	-	-	-	**1669.3 ± 503.4**
0.01	-	-	**1840.5 ± 292.0**	**1434.5 ± 339.7**
0.05	-	-	414.3 ± 67.8	320.2 ± 92.1
0.1	306.4 ± 75.2	232.2 ± 63.2	169.6 ± 42.3	127.4 ± 31.7
0.2	186.2 ± 91.8	98.9 ± 51.9	49.4 ± 36.0	38.0 ± 26.1
0.3	153.6 ± 69.4	63.8 ± 45.6	25.8 ± 24.8	16.8 ± 12.9
0.4	150.2 ± 99.8	64.7 ± 52.0	21.0 ± 16.4	16.1 ± 12.1
0.5	138.4 ± 107.5	53.3 ± 42.5	24.2 ± 22.6	11.6 ± 16.3
0.6	158.8 ± 75.5	63.8 ± 36.6	21.6 ± 16.5	10.2 ± 11.4
0.7	84.2 ± 53.1	36.5 ± 21.2	11.3 ± 8.2	8.7 ± 6.5
0.8	94.1 ± 53.1	36.5 ± 21.9	11.8 ± 7.7	9.4 ± 7.0
0.9	118.3 ± 82.9	48.8 ± 34.1	18.4 ± 13.1	9.0 ± 5.7
1.0	140.6 ± 72.1	58.8 ± 38.5	24.4 ± 15.2	12.8 ± 9.3

Table 8: Number of iterations needed to recover for the selected thresholds for Gap 2. Values are reported as mean ± standard deviation across 20 runs. Boldface highlights rows corresponding to learning rates of 0.001, 0.005, and 0.01, which were trained for 2500 iterations instead of 500, to clarify the higher recovery counts. A dash ("–") indicates that fewer than 85% of runs (i.e., fewer than 17 out of 20) reached the specified threshold, and thus the value is not reported.

Threshold Learning rate	REC-TIME 100%	REC-TIME 99%	REC-TIME 98%	REC-TIME 97%
0.001	**1561.5 ± 888.2**	**1599.8 ± 881.5**	**1632.0 ± 868.2**	**1671.2 ± 864.9**
0.005	-	-	-	**1762.4 ± 535.0**
0.01	**1688.0 ± 327.8**	**1351.7 ± 324.0**	**959.2 ± 271.8**	**696.1 ± 182.3**
0.05	359.1 ± 55.7	297.1 ± 60.5	211.0 ± 56.2	154.0 ± 47.8
0.1	249.5 ± 62.0	195.2 ± 47.9	135.6 ± 48.7	103.6 ± 39.4
0.2	202.3 ± 55.0	126.3 ± 31.2	70.9 ± 16.5	33.7 ± 20.0
0.3	232.5 ± 110.6	110.8 ± 55.3	56.8 ± 18.7	27.8 ± 17.3
0.4	175.6 ± 85.0	78.4 ± 40.1	35.2 ± 17.1	17.4 ± 9.1
0.5	171.4 ± 96.7	69.2 ± 33.5	34.0 ± 18.1	14.6 ± 10.0
0.6	147.2 ± 59.9	59.2 ± 34.7	24.4 ± 14.8	11.6 ± 7.0
0.7	158.9 ± 57.9	62.2 ± 23.7	29.4 ± 13.5	12.3 ± 6.2
0.8	170.8 ± 110.7	70.2 ± 33.0	31.2 ± 21.3	16.0 ± 15.1
0.9	163.6 ± 80.7	67.6 ± 33.1	25.1 ± 13.6	11.6 ± 7.1
1.0	128.4 ± 58.7	52.6 ± 29.8	20.2 ± 11.4	11.5 ± 6.7

Drop Slope (DROP-SLOPE)

Table 9: DROP-SLOPE for Gap 1 and Gap 2 respectively, reported as mean ± standard deviation across 20 runs.

Learning rate	DROP-SLOPE Gap 1	DROP-SLOPE Gap 2
0.001	-0.06 ± 0.24	-0.01 ± 0.03
0.005	-0.0 ± 0.0	-0.01 ± 0.0
0.01	-0.01 ± 0.0	-0.01 ± 0.0
0.05	-0.04 ± 0.02	-0.04 ± 0.02
0.1	-0.05 ± 0.02	-0.06 ± 0.05
0.2	-1.15 ± 1.71	-0.75 ± 1.13
0.3	-1.88 ± 2.35	-2.28 ± 2.34
0.4	-2.37 ± 3.06	-2.87 ± 2.41
0.5	-4.83 ± 3.37	-4.96 ± 2.75
0.6	-4.5 ± 3.16	-5.21 ± 2.87
0.7	-4.71 ± 3.04	-4.21 ± 2.75
0.8	-3.78 ± 2.61	-4.81 ± 2.97
0.9	-6.31 ± 3.98	-5.74 ± 2.79
1.0	-4.86 ± 3.6	-4.97 ± 2.28

Knowledge Loss (AUC-LOSS)

Table 10: AUC-LOSS for Gap 1 and Gap 2 respectively, reported as mean ± standard deviation across 20 runs.

Learning rate	AUC-LOSS Gap 1	AUC-LOSS Gap 2
0.001	-0.01 ± 0.04	-0.0 ± 0.02
0.005	0.05 ± 0.01	0.05 ± 0.01
0.01	0.05 ± 0.01	0.03 ± 0.01
0.05	0.05 ± 0.01	0.03 ± 0.01
0.1	0.02 ± 0.01	0.02 ± 0.01
0.2	0.01 ± 0.01	0.01 ± 0.0
0.3	0.01 ± 0.01	0.01 ± 0.01
0.4	0.01 ± 0.01	0.01 ± 0.01
0.5	0.01 ± 0.01	0.01 ± 0.0
0.6	0.01 ± 0.01	0.01 ± 0.0
0.7	0.0 ± 0.01	0.01 ± 0.0
0.8	0.0 ± 0.01	0.01 ± 0.01
0.9	0.0 ± 0.01	0.01 ± 0.0
1.0	0.01 ± 0.01	0.01 ± 0.0

E Grid Search - Best configurations found for IncreaseLROnPlateau and CyclicLR

This appendix lists the hyperparameter configurations selected as optimal for each scheduler based on the objective function defined in Equation 1. These configurations were used for the final experiments and figures shown in the main paper.

E.1 Best Configuration for IncreaseLROnPlateau

Table 11: Best hyperparameters found for IncreaseLROnPlateau

Parameter	Selected Value
`min_lr`	0.05
`max_lr`	0.9
`factor`	1.5
`patience`	5
`threshold`	1e-5

E.2 Best Configuration for CyclicLR

Table 12: Best hyperparameters found for CyclicLR (Triangular2 mode)

Parameter	Selected Value
`mode`	Triangular2
`min_lr`	0.05
`max_lr`	0.8
`step_size`	250
`cycle_momentum`	False

Note All other non-listed parameters were kept at their PyTorch default values.

F Use of LLMs

During the preparation of this paper, ChatGPT (by OpenAI) was used as a support tool for improving the academic style and ensuring a clear and logical structure of the text. Typical prompts included:

- *"Could you rephrase this paragraph in a more academic style?"*
- *"Does this section flow logically? How could I reorganize it?"*
- *"Could you suggest a clearer way to express this idea?"*

- *“Can you help rewrite this sentence to sound more formal?”*

All generated suggestions were carefully reviewed and edited by the author to ensure correctness and appropriateness for the final report.