Sharpness-Aware Optimization for Stability Gap Reduction

Ksenia Sycheva [1]

Supervisor(s): 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: Ksenia Sycheva
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Gido Van de Ven, Alan Hanjalic

Abstract

One of the problems in continual learning, where models are trained sequentially on tasks, is a sudden drop in performance after switching to a new task, called *stability gap*. The presence of stability gap likely indicates that training is not done optimally. In this work we aim to address stability gap problem by using sharpness-aware optimization that biases convergence to flat minima. While flat minima are known to mitigate forgetting, their role in ensuring stable learning *during* task transitions remains unexplored. Through systematic analysis of two Entropy-SGD and C-Flat, we demonstrate that sharpness-aware optimizers produce smoother learning trajectories with reduced instability after task switch. Furthermore, we show that C-Flat's second-order curvature approximation provides additional stabilization, suggesting that efficient Hessian-aware methods offer advantages for continual learning. The source code is available at Stability-Gap-SAM.

1 Introduction

Continual learning (CL) is a machine learning paradigm where models are trained on a sequence of tasks, with the goal of maintaining strong performance on all tasks over time. In this paper, we focus on the *offline* setting of continual learning, where data is encountered in discrete, well-defined tasks (e.g., Task 1, Task 2, etc.), and the model can train on each task for multiple epochs before transitioning to the next. Specifically, we consider *domain-incremental* learning, where tasks share the same output space (e.g., the same set of classes) but exhibit shifting input distributions—meaning old tasks remain relevant in the data while new tasks are incrementally introduced. This differs from *task-incremental* or *class-incremental* learning, where task identities or class sets change over time [van de Ven and Tolias, 2019]. While continual learning is highly applicable to real-world scenarios, it faces two key challenges:

1. **Catastrophic forgetting**: Models tend to abruptly forget previously learned information after switching to a new task [McCloskey and Cohen, 1989].
2. **Stability gap**: Even when catastrophic forgetting is mitigated, models often exhibit temporary but significant performance drops on prior tasks after learning new ones. Though performance may eventually recover, this instability is inefficient and poses risks in safety-critical applications (e.g., autonomous systems or medical diagnostics).

Standard approaches address these through regularization [Kirkpatrick et al., 2017], architectural modifications, or replay mechanisms [Yoo et al., 2024]. While effective, these methods often treat symptoms rather than underlying causes. Recent work suggests that optimization geometry plays a crucial role - models converging to flat minima demonstrate improved generalization and forgetting resistance [Chaudhari et al., 2017].

While these methods have demonstrated effectiveness in finding flat minima, two critical aspects remain underexplored: (1) the specific impact of sharpness-aware optimization on reducing the stability gap in continual learning, and (2) the potential of second-order optimization approaches for more robust flat minima discovery.

This gap motivates two key hypotheses:

- **H1:** Sharpness-aware optimization directly contributes to stability gap reduction in continual learning systems.
- **H2:** Incorporating second-order information into sharpness-aware optimizers yields additional improvements in stability gap mitigation.

A tradeoff arises in addressing these issues: many methods that reduce forgetting or stability gaps inadvertently degrade the model's ability to learn new tasks efficiently. Our work explicitly targets **both** objectives.

Our work makes the following contributions:

- We analyze how sharpness-aware optimization impacts training dynamics in CL, specifically after task transitions.
- We provide the first empirical evidence that second-order curvature information directly reduces stability gaps.

The remainder of this paper is organized as follows. Section 2 discusses flat minima optimization and existing CL approaches. Section 3 gives more details on how second-order information can be incorporated in training objective. Section 4 details experiments and reports results with their interpretation. At the end, we discuss limitations and possible future directions of work.

2 Background

We briefly outline why sharpness-aware optimization can mitigate the stability gap by favoring flat regions of the loss landscape.

Stability Gap

The stability gap, first formalized by Lange et al. [2023], refers to the transient performance degradation observed after task transitions in continual learning scenarios. Various mitigation strategies have emerged to address this phenomenon, including soft targets, low-rank adaptation techniques that selectively constrain hidden layer plasticity (e.g., LoRA [Hu et al., 2021]), and output layer freezing mechanisms [Harun and Kanan, 2024]. While these approaches demonstrate efficacy in gap reduction, they inherently limit model plasticity and may consequently impair overall performance. Interestingly, certain replay-based methods like those proposed by Drusvyatskiy [2017] - though not explicitly evaluated for stability gap mitigation - show promising stability characteristics in preserving prior knowledge, suggesting their potential applicability to this challenge.

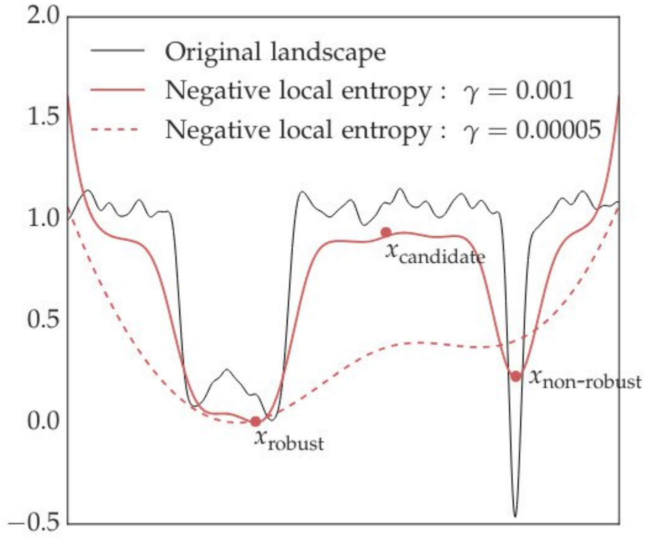


Figure 1: Local entropy concentrates on wide valleys in the energy landscape. Image is taken from Chaudhari et al. [2017].

Sharpness-Aware Optimization

The geometry of local minima plays a crucial role in model generalization. Of particular interest are *flat minima* - regions of parameter space where the loss function remains relatively insensitive to small perturbations. This flatness property can be formally characterized through the spectral properties of the Hessian matrix, where flat minima correspond to those with many small-magnitude eigenvalues [Chaudhari et al., 2017]. The connection between flat minima and improved generalization has motivated the development of optimization methods specifically designed to converge to such regions - sharpness-aware optimizers. We hypothesize that flat minima may offer particular advantages in continual learning: when optimization converges to a flat region during the first task, the broader basin of low loss is more likely to contain shared solutions that remain valid after task switches, thereby reducing stability gap. Below we describe two sharpness-aware optimizers considered in our work.

Entropy-SGD. Chaudhari et al. [2017] introduced one of the first sharpness-aware optimization methods by reformulating the optimization objective using local entropy:

$$F(\theta,\gamma) = \log \int_{\theta' \in \mathbb{R}^n} \exp\left(-f(\theta') - \frac{\gamma}{2}\|\theta - \theta'\|_2^2\right) d\theta' \quad (1)$$

where f is loss function, θ represents the model weights and θ' denotes perturbed weight configurations within a neighborhood of θ.

As illustrated in Figure 1, this formulation induces a smoothed loss landscape by integrating over a Gaussian-weighted region centered at θ. The hyperparameter γ governs the trade-off between local loss minimization and neighborhood flatness. By maximizing this local entropy measure, the optimizer preferentially converges to broad, flat minima that demonstrate better generalization properties, while avoiding narrow, sharp minima that typically overfit to training data. The objective is approximated via stochastic gradient Langevin dynamics (SGLD) [Welling and Teh, 2011], which introduces hyperparameters analyzed in Appendix B. This framework is optimizer-agnostic: we term the SGD-based implementation *Entropy-SGD* and the Adam variant *Entropy-Adam*, collectively referring to the approach as *Entropy-Regularized Optimization* when discussing general principles.

C-Flat. While Entropy-SGD provides a general approach for finding flat minima, Bian et al. [2024] developed C-Flat specifically to address catastrophic forgetting in continual learning. Their key insight is that improved generalization at each training stage naturally mitigates forgetting. C-Flat achieves this by combining zeroth-order and second-order information approximated using gradient. The size of the considered neighborhood is controlled by hyperparamters ρ, while contribution of second-order information is additionally weighted by ϕ. The authors demonstrate that targeted flatness optimization leads to more stable performance across sequential tasks while maintaining competitive accuracy on current tasks.

3 First vs. Second-Order Flatness Optimization

The main difference between Entropy-regularized optimizers and C-Flat is that the former uses zero-order information only to guide optimization towards flat minima, while the latter in addition to zero-order values utilizes loss gradients. These gradients enable efficient estimation of second-order geometric properties, bypassing explicit Hessian computation while still capturing the curvature information most relevant to stability gap reduction. This section discusses why gradient-based curvature approximation is theoretically well-founded for stability preservation in continual learning.

Approach

Hessian of flat minima has distinct properties. As discussed in Section 2, flat minima have most of its eigenvalues with low magnitude. In Entropy-regularized optimization this property is not taken into account explicitly: training is regularized by averaging loss values in the neighborhood around current weights. In contrast to this, C-Flat targets this property directly by using gradients to regularize training objective:

$$\rho \cdot \max\{||\nabla \mathcal{L}(\theta')|| : \theta' \in B(\theta,\rho)\} \quad (2)$$

where $B(\theta,\rho)$ is a ball centered at θ with radius ρ.

According to Bian et al. [2024], Equation (2) is not enough to converge to local minima. Therefore, they propose to use it in combination with zero-order sharpness. The final objective is:

$$\begin{gathered}\max\{\mathcal{L}(\theta') : \theta' \in B(\theta,\rho)\} \\ +\phi \cdot \rho \cdot \max\{||\nabla \mathcal{L}(\theta')|| : \theta' \in B(\theta,\rho)\}\end{gathered} \quad (3)$$

This method is orthogonal to existing continual learning techniques - it can be combined with replay-based, regularization-based, or architectural approaches without conflict. For implementation details, see Algorithm 1. Below, we formalize the connection to Hessian properties.

Diagonal Approximation of Hessian

Although second-order methods provide superior convergence properties through exact Hessian information, their computational and memory requirements scale quadratically with parameter count. This limitation has driven the development of efficient approximations of Hessian, like diagonal [Becker and Lecun, 1989], or Kronecker-factored methods [Martens and Grosse, 2020]. As discussed in Elsayed et al. [2024], high quality of diagonal approximations suggest that Hessian matrices have dominant diagonal entries. According to Gershgorin Theorem [Gershgorin, 1931], all eigenvalues of Hessian must lie close to diagonal entries. Specifically, for a Hessian matrix H with $|H_{ii}| >> \sum_{j\neq i}|H_{ij}|$, each eigenvalue λ satisfies $|\lambda - H_{ii}| \leq R_i$ where R_i (the Gershgorin radius) is small relative to $|H_{ii}|$. Thus, diagonal entries closely approximate the eigenvalues, and consequently, Equation (3) effectively constrains the spectral properties of the Hessian through these computationally tractable diagonal approximations.

Algorithm 1 Second-Order Flat Minima Optimization

```
Require: Model parameters $\theta$, loss function $\mathcal{L}$, neighborhood radius $\rho$, second-order penalty weight $\phi$
1: Compute $\mathcal{L}(\theta)$                                   // Forward pass
2: $\theta'_0 \leftarrow \theta + \frac{\nabla\mathcal{L}(\theta)}{||\mathcal{L}(\theta)||_2}$   // Perturb weights $\theta$ to approximate zero-order flatness term
3: Compute $\nabla\mathcal{L}(\theta'_0)$
4: $\theta'_1 \leftarrow \theta + \frac{\nabla||\nabla\mathcal{L}(\theta)||_2}{||\nabla||\mathcal{L}(\theta)||_2||_2}$   // Perturb weights $\theta$ to approximate second-order flatness term
5: Compute $\nabla\mathcal{L}(\theta'_1)$
6: Aggregate gradients
7: Perform optimization step with base optimizer
```

4 Experimental Setup and Results

We evaluate all methods on a rotated MNIST benchmark [Deng, 2012] using three distinct rotation angles (0°, 80°, 160°) in a domain-incremental learning scenario, with primary results reported for the fixed task order (0°→ 160°→ 80°) and additional validation performed using the reversed order (0°→ 80°→ 160°) to verify consistency (see Appendix A).

The base architecture consists of a fully-connected neural network with two hidden layers (400 units each) and ReLU [Agarap, 2019] activations. In all experiments batch size was set to 128, and number of training steps for each task to 1000. We implement two optimization baselines: (1) SGD and (2) Adam, with their entropy-regularized variants (Entropy-SGD, Entropy-Adam) and flatness-aware versions (C-Flat). For main experiments we set number of SGLD iterations to 15, γ to 0.001 in Entropy-SGD/Adam, and $\rho = 0.05, \phi = 2.0$ in C-Flat. These values were selected based on preliminary ablation studies (detailed in Appendix B). To ensure statistical reliability all experiments are repeated across five random seeds. Reported metrics represent averages over all runs.

Metrics

To evaluate sharpness-aware optimizers against the baseline, we compute several metrics, which are described below.

Maximum drop (MD) measures the maximum accuracy decline between the point immediately before a task switch and the point before the next switch, quantifying forgetting severity during transitions.
Recovery steps (RS) counts the iterations needed to recover from the worst forgetting event, identified by first locating the iteration with maximum stability gap depth (lowest accuracy), then measuring number of steps until accuracy returns to or exceeds its last pre-drop level, ensuring measurement from the lowest point with sustained improvement.
Accuracy on all tasks evaluates final model performance across all tasks after training completion, verifying that stability improvements don't compromise overall task performance.
All metrics are first computed per random seed, then averaged across seeds (N=5) with standard errors noted. Pseudocode for stability gap metrics is provided in Appendix C.

Results

We report results on different optimizers with fixed hyperparameters on five seeds. In addition to quantitative results, we provide qualitative analysis of the performance of different optimizers.

Quantitative Results

In experiments with both optimizers (SGD and Adam), we can see that training dynamics is affected by incorporating sharpness-aware regularization.

SGD-based optimizers optimizers demonstrate consistent improvements in stability while maintaining task performance compared to SGD baseline. As visible in Figure 2, both Entropy-SGD and C-Flat (SGD) significantly reduce post-switch accuracy drops compared to vanilla SGD across both task transitions (see Table 3). Entropy-SGD shows particularly strong recovery properties, achieving 16% (first switch) and 36% (second switch) smaller accuracy drop on average. While C-Flat (SGD) exhibits slightly slower initial recovery after the first transition, it outperforms vanilla SGD by 8% in recovery speed after the second transition, and has the smallest accuracy drop after first task switch. Final task accuracies remain competitive, with Entropy-SGD matching baseline SGD performance and C-Flat (SGD) showing only marginal degradation (0.75 percentage points across tasks), as shown in Table 1.
Adam-based optimizers exhibit distinct stability patterns (see Figure 3) compared to their SGD counterparts (Table 4). While all variants maintain competitive final accuracies (within 0.29 percentage points of baseline, see Table 2), C-Flat (Adam) demonstrates consistent stability improvements: recovery times decrease by 36% (first switch:

101.2 vs 157.2 iterations) and 82% (second switch: 59.2 vs 330.2 iterations), while accuracy drops reduce by 23% (first switch: 12.18 vs 15.81 points) and 14% (second switch: 3.87 vs 4.48 points) compared to vanilla Adam. This strong performance contrasts sharply with Entropy-Adam, which increases instability (23.26 vs 15.81 drop) - reversing Entropy-SGD's effects. The dramatic instability increase with Entropy-Adam reveals an incompatibility between entropy regularization and adaptive gradients, whereas C-Flat's second-order flatness minimization remains robust in Adam-based optimization.

Optimizer	**Task 1**	**Task 2**	**Task 3**
SGD	$\underline{95.42 \pm 0.44}$	$\underline{95.19 \pm 0.48}$	$\underline{94.18 \pm 0.55}$
Entropy-SGD	$\mathbf{96.05 \pm 0.31}$	$\mathbf{96.04 \pm 0.30}$	$\mathbf{94.94 \pm 0.16}$
C-Flat (SGD)	95.04 ± 0.44	94.89 ± 0.49	93.43 ± 0.26

Table 1: Comparison of test accuracy (%) for SGD-based optimizers. Entropy-SGD achieves the highest performance on Tasks 2 and 3, while SGD performs best on Task 1. C-Flat (SGD) lags slightly behind in all tasks. n all experiments base optimizer learning rate was set to 0.1. Bold and underlined values denote the top two results per task.

Optimizer	**Task 1**	**Task 2**	**Task 3**
Adam	96.36 ± 0.35	95.95 ± 0.40	$\underline{95.47 \pm 0.26}$
Entropy-Adam	$\mathbf{97.24 \pm 0.34}$	$\mathbf{97.24 \pm 0.15}$	$\mathbf{96.33 \pm 0.14}$
C-Flat (Adam)	$\underline{96.69 \pm 0.33}$	$\underline{96.14 \pm 0.37}$	95.18 ± 0.50

Table 2: Comparison of test accuracy (%) for Adam-based optimizers. Both Entropy-Adam and C-Flat with Adam as base optimizer outperform Adam on the first two tasks. In these experiments learning rate of base optimizer was set to 0.01.

Qualitative Results

Additionally, we conduct several experiments with C-Flat by varying ϕ values, which control the effect of second-order information on optimization. We experiments with $\phi \in \{0, 1.0, 2.0\}$ and use SGD as base optimizer. The rest of the setup is the same as described above. As shown in Figure 4, increasing ϕ leads to smaller MD without noticeable drop in accuracy. This relationship between ϕ and stability gap provide empirical evidence that second-order curvature offers precise control over transient stability gaps, beyond what pure zero-order optimization can achieve.

5 Discussion and Future Work

Analysis. Our study demonstrates that sharpness-aware optimization significantly stabilizes training dynamics in continual learning. We evaluate two existing sharpness-aware techniques: Entropy-Regularized optimization and C-Flat. Both methods applied to SGD outperform vanilla SGD. Notably, while C-Flat maintains comparable performance when applied to Adam, Entropy-Adam exhibits an interesting tradeoff: it reduces recovery steps but increases gap depth. These results reveal that flat minima convergence provides a principled approach to stability gap reduction, with second-order curvature information playing a particularly crucial role. This observation aligns with results obtained by Bian et al. [2024] for catastrophic forgetting.

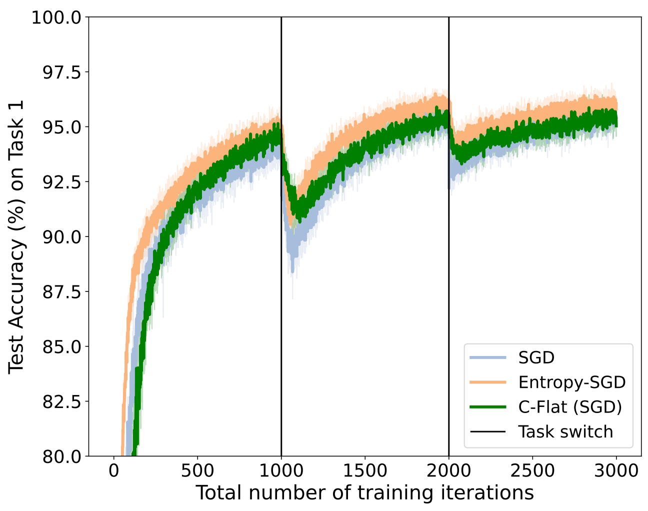


Figure 2: Task 1 accuracy trajectories demonstrating stability gap characteristics of SGD-based optimizers during incremental training (shaded regions indicate ±1 standard error across runs). Both Entropy-SGD and C-Flat exhibit faster recovery from post-switch accuracy drops and better stability preservation compared to vanilla SGD, while simultaneously maintaining competitive downstream task performance.

Limitations. Our work has several limitations. The most significant limitation is the high standard error in recovery step (RS) measurements, suggesting the need for more robust stability metrics. In addition to that, all experiments were conducted on MNIST in a domain-incremental setup, which provides initial validation but may not fully represent challenges in more complex scenarios like CIFAR [Krizhevsky, 2012] or ImageNet [Deng et al., 2009]. The study also focuses exclusively on three-task sequences, while scalability to longer task streams was not tested. While the current implementation uses efficient small-scale models, scaling to larger architectures may reveal additional computational constraints - particularly since sharpness-aware optimizers introduce computational overhead that could become significant at scale.

Future work. This study suggests several promising research directions. First, addressing the current limitations through evaluation on more complex benchmarks (e.g., CIFAR, ImageNet) and longer task sequences would strengthen the findings. Second, investigating other sharpness-aware optimizers could provide deeper insights into how different loss landscape properties affect stability gaps. Most importantly, while our empirical results demonstrate the value of second-order information for stability gap mitigation, formal theoretical analysis of this relationship remains unstudied.

First Task Switch			Second Task Switch		
Optimizer	**MD** $\downarrow$	**RS** $\downarrow$	**Optimizer**	**MD** $\downarrow$	**RS** $\downarrow$
SGD	6.29 ± 1.0	$\underline{361.8 \pm 68.96}$	SGD	4.21 ± 0.62	384.4 ± 89.43
Entropy-SGD	$\underline{5.25 \pm 0.33}$	$\mathbf{339.6 \pm 60.01}$	Entropy-SGD	$\mathbf{2.78 \pm 0.39}$	$\mathbf{252.4 \pm 149.36}$
C-Flat (SGD)	$\mathbf{4.61 \pm 0.57}$	408.2 ± 82.35	C-Flat (SGD)	$\underline{3.08 \pm 0.32}$	$\underline{355.4 \pm 134.59}$

Table 3: Stability gap analysis across task transitions for SGD-based optimizers, measuring both accuracy drop (percentage points) and recovery time (iterations). Entropy-SGD shows the fastest recovery after the first transition, while both modified optimizers (Entropy-SGD and C-Flat) achieve significantly smaller accuracy drops than vanilla SGD at both transitions.

First Task Switch			Second Task Switch		
Optimizer	**MD** $\downarrow$	**RS** $\downarrow$	**Optimizer**	**MD** $\downarrow$	**RS** $\downarrow$
Adam	$\underline{15.81 \pm 1.58}$	157.2 ± 97.94	Adam	$\underline{4.48 \pm 0.69}$	330.2 ± 345.56
Entropy-Adam	23.26 ± 0.98	$\mathbf{82.6 \pm 29.80}$	Entropy-Adam	6.68 ± 0.90	$\underline{87.8 \pm 40.91}$
C-Flat (Adam)	$\mathbf{12.18 \pm 1.82}$	$\underline{101.2 \pm 21.90}$	C-Flat (Adam)	$\mathbf{3.87 \pm 1.02}$	$\mathbf{59.2 \pm 28.02}$

Table 4: Stability gap analysis for Adam-based optimizers reveals distinct behaviors: C-Flat achieves both the smallest accuracy drops and fastest recovery times across transitions, while Entropy-Adam underperforms with larger accuracy drops.

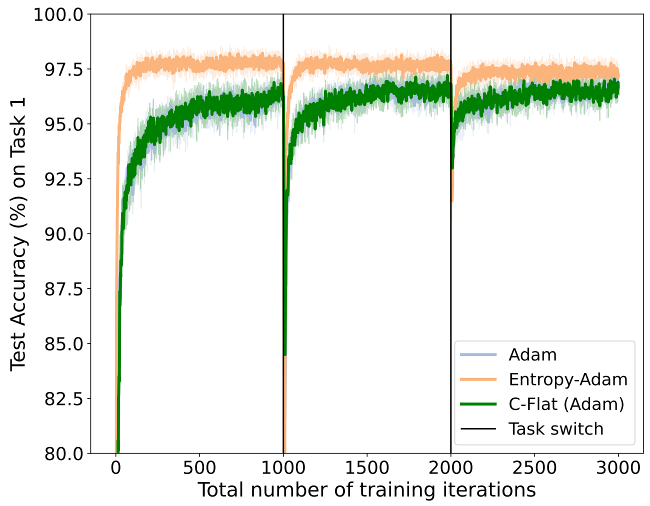


Figure 3: Task 1 accuracy trajectories for Adam-based optimizers, revealing distinct stability gap behaviors. While C-Flat demonstrates faster recovery from task switches, Entropy-Adam shows notably degraded performance compared to both its SGD counterpart and baseline Adam, suggesting the entropy regularization approach may be less suitable in continual learning setting.

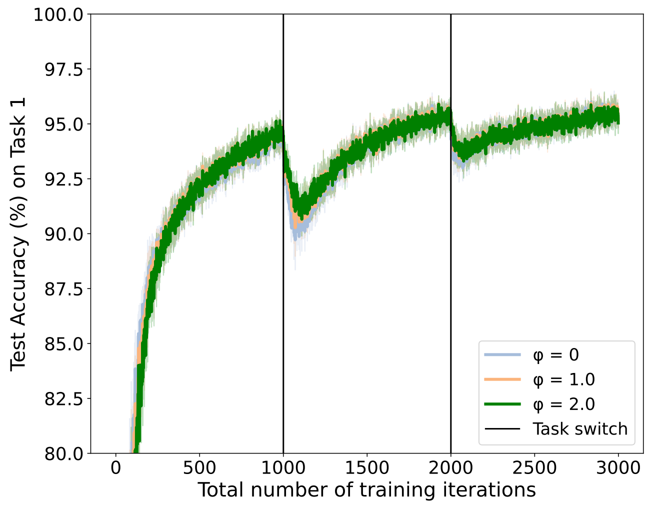


Figure 4: Task 1 accuracy trajectories of C-Flat (SGD) optimizers with different ϕ values during incremental training with different. Larger ϕ values reduce MD more without sacrificing performance significantly.

6 Conclusions

In this work we investigated the effect of sharpness-aware optimization on stability gap in continual learning setting. Such optimizers are biased towards flat minima. We hypothesize that flat minima reduce stability gaps by increasing the likelihood of shared optimal solutions across tasks—their broad basins maintain low loss even when the input distribution shifts. In particular, we examined two methods—Entropy-regularized optimization and C-Flat—with the latter explicitly incorporating second-order curvature information to study its distinct impact on stability gap reduction.

Our results demonstrate that sharpness-aware optimization effectively reduces stability gaps, with second-order methods delivering more consistent improvements, without affecting negatively model's performance on other tasks. Our experiments show that choice of hyperparameters that control effect of the neighborhood points should be done properly, a critical consideration when implementing this approach.

7 Responsible Research

This work adheres to ethical research practices in machine learning. All experiments were conducted using the publicly available MNIST dataset, which contains anonymized handwritten digits and poses no privacy concerns. The model was implemented using the open-source PyTorch framework, ensuring transparency and reproducibility.

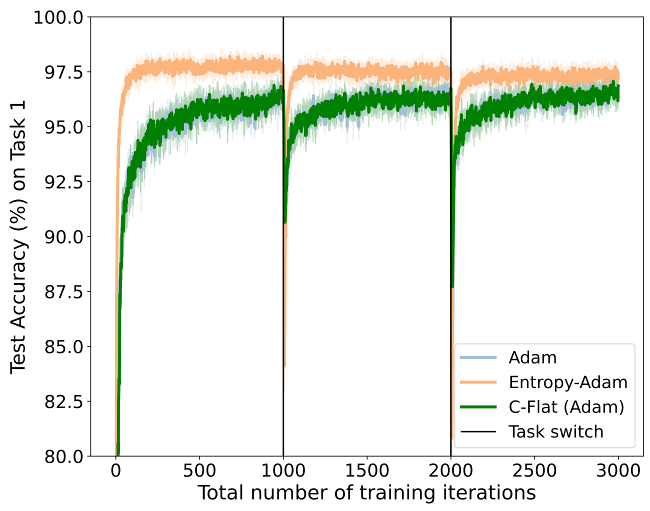


Figure 5: Task 1 accuracy trajectories for Adam-based optimizers on task sequence 0°→ 80°→ 160°. The behavior observed on this task sequence is similar to the one on 0°→ 160°→ 80°, used in main experiments.

To minimize environmental impact, experiments were performed on a single NVIDIA RTX A2000 GPU, optimizing computational efficiency. The full code will be open-sourced upon publication to facilitate reproducibility and further research in continual learning.
Since this study uses benchmark data without human-subject risks, formal ethics approval was not required. However, we acknowledge the broader responsibility in deploying continual learning systems and encourage future work to assess fairness and bias when adapting to real-world streaming data.

Appendix

A Permuted Order

To ensure that results are not dependent on specific task order, we repeated experiments with Adam using another permutation of tasks: 0°→ 80°→ 160°. The rest of the setup coincides with the one described in Section 4. From Figure 5 it is visible that the overall trend is similar to the one observed before. Numerical results in Table 5 confirm that the relative performance of methods is not altered. Interestingly, the absolute values of MD are different from values reported in Table 4: in Table 4 drop after first switch is much larger than drop after second switch, while in Table 5 these values are similar in all experiments. This suggests that order of tasks has significant effect on model's training dynamics.

B Hyperparameters

We experiment with hyperparameters specific for sharpness-aware optimizers discussed in the paper. Here we provide a more detailed description of these hyperparameters to motivate choice of their values in the main experiments.

For **Entropy-SGD** we consider two hyperparameters: number of SGLD steps L and scale γ. SGLD [Welling and Teh, 2011] is a Markov chain Monte-Carlo (MCMC) algorithm designed to draw samples from a Bayesian posterior. In Entropy-SGD number of SGLD steps L determines the precision of approximation of the objective in Equation (1). We experimented with $L \in \{5, 15, 25\}$, keeping $\gamma = 0.001$. As demonstrated in Figure 6a, increasing L leads to higher accuracies across all tasks and reduced stability gap. Since training time grows with L, for main experiments we choose $L = 15$ to balance between time and performance. In contrast to L, Entropy-SGD does not show sensitivity to scope γ, providing consistent results for a wide range of γ, as shown in Figure 6b.
C-Flat has two hyperparameters: ρ that defines the width of the considered neighborhood, and ϕ that controls contribution of second-order curvature. Results of experiments with ϕ are analyzed in Section 4. Here we additionally present results for $\rho \in \{0.0, 0.05, 0.1\}$ with fixed $\phi = 2.0$. From results in Figure 7 we can see that larger values of ρ lead to decrease in stability gap width. However, changing this values results additionally in decrease in accuracy.

C Stability Gap Metrics

Our evaluation quantifies continual learning stability through two metrics computed from the accuracy trajectory $A(t)$. As formalized in Algorithm 2 and Algorithm 3, both measures operate at the iteration level, requiring: (1) task switch points t_i, and (2) per-iteration accuracy values. The metrics capture complementary aspects of transient instability—*magnitude* and *duration*.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019. URL https://arxiv.org/abs/1803.08375.

Suzanna Becker and Yann Lecun. Improving the convergence of back-propagation learning with second-order methods. 01 1989.

Ang Bian, Wei Li, Hangjie Yuan, Chengrong Yu, Mang Wang, Zixiang Zhao, Aojun Lu, Pengliang Ji, and Tao Feng. Make continual learning stronger via c-flat, 2024. URL https://arxiv.org/abs/2404.00986.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys, 2017. URL https://arxiv.org/abs/1611.01838.

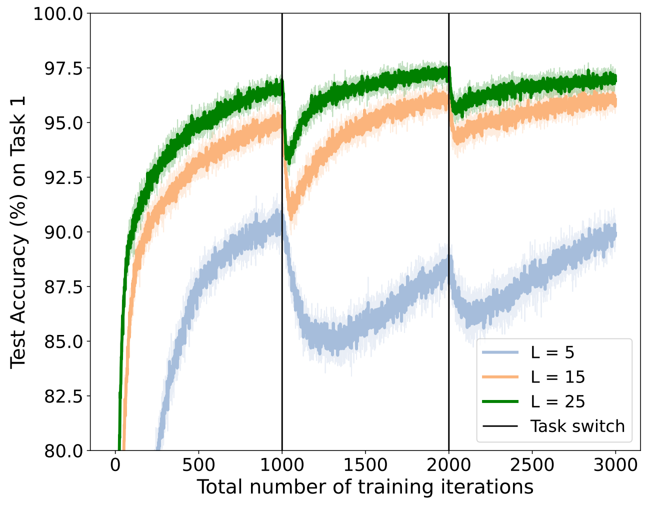
Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

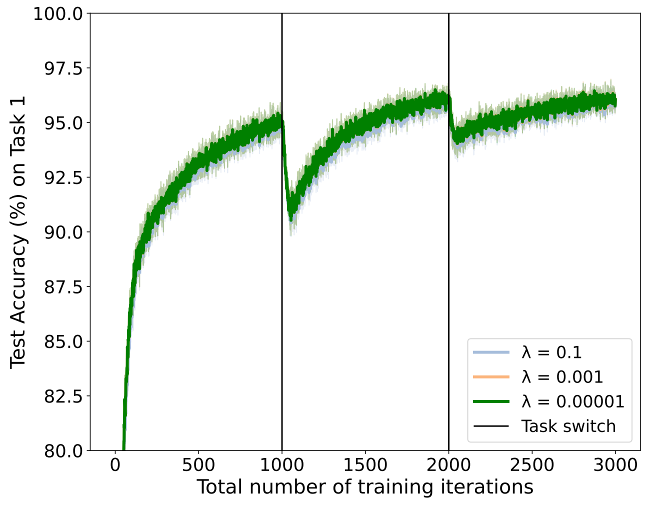
Dmitriy Drusvyatskiy. The proximal point method revisited, 2017. URL https://arxiv.org/abs/1712.06038.

First Task Switch			Second Task Switch		
Optimizer	**MD ↓**	**RS ↓**	**Optimizer**	**MD ↓**	**RS ↓**
Adam	11.26 ± 1.09	174.0 ± 181.04	Adam	11.58 ± 0.85	176.0 ± 71.89
Entropy-Adam	17.30 ± 1.53	**74.20 ± 17.42**	Entropy-Adam	17.01 ± 1.84	83.0 ± 25.01
C-Flat (Adam)	**9.02 ± 2.04**	114.4 ± 54.35	C-Flat (Adam)	**9.47 ± 2.02**	**225.8 ± 176.50**

Table 5: Robustness analysis with permuted task order (Tasks 2 & 3 swapped). While absolute metric values vary between switches, C-Flat (Adam) consistently achieves the lowest maximum drops (MD) across both transitions (9.02 vs 11.26 and 9.47 vs 11.58 pp), confirming its task-order invariance.



(a) Varying SGLD steps (L)



(b) Varying scope (γ)

Figure 6: Task 1 accuracy trajectories for Entropy-SGD under different hyperparameter configurations. **(a)** Larger SGLD steps (L) improve both final accuracy and reduce post-switch drops. **(b)** In contrast, the stability gap shows minimal sensitivity to the scope parameter (γ), suggesting that local entropy range has limited impact on transient stability.

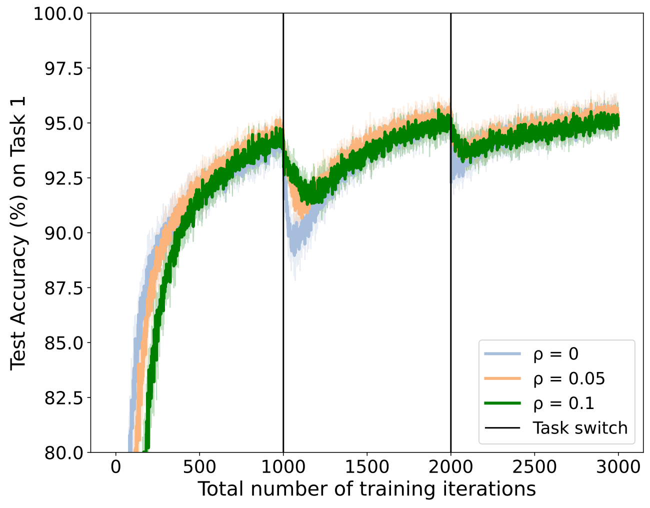


Figure 7: Task 1 accuracy trajectories for C-Flat with varying ρ. Our results show that stability gap decreases with larger ρ values, while introducing slight decrease in accuracy.

Mohamed Elsayed, Homayoon Farrahi, Felix Dangel, and A. Rupam Mahmood. Revisiting scalable hessian diagonal approximations for applications in reinforcement learning, 2024. URL https://arxiv.org/abs/2406.03276.

S. Gershgorin. On the bounding of eigenvalues of a matrix. *Izvestiya Akademii Nauk SSSR, Seriya Matematicheskaya*, 7:749–754, 1931. Translated title from Russian.

Md Yousuf Harun and Christopher Kanan. Overcoming the stability gap in continual learning, 2024. URL https://arxiv.org/abs/2306.01904.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

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), March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Algorithm 2 Stability Gap Magnitude (ΔA_{max})

```
Require: Task switch $t_i$, accuracy $A(t)$
1: $A_{pre} \leftarrow A(t_i - 1)$
2: $\Delta A \leftarrow A_{pre} - \min_{t \in [t_i, t_{i+1}]} A(t)$
3: return $\Delta A$
```

Algorithm 3 Recovery Time (T_{rec})

```
Require: Task switch $t_i$, accuracy $A(t)$
1: $A_{pre} \leftarrow A(t_i - 1)$
2: $t_{min} \leftarrow \text{argmin}_{t \in [t_i, t_{i+1}]} A(t)$ // Get index of last such value
3: $T \leftarrow \min\{t > t_{min} | A(t) \geq A_{pre}\}$
4: return $T$
```

Figure 8: Algorithms for computing stability gap metrics: a) maximum accuracy drop b) time to recover. t_i is the number of iteration when i-th task switch happened; $A(t)$ is a function that returns accuracy for a given iteration. In all experiments, evaluation was run after every training iteration.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Matthias De Lange, Gido van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap, 2023. URL https://arxiv.org/abs/2205.13452.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2020. URL https://arxiv.org/abs/1503.05671.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. URL https://api.semanticscholar.org/CorpusID:61019113.

Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning, 2019. URL https://arxiv.org/abs/1904.07734.

Max Welling and Yee Teh. Bayesian learning via stochastic gradient langevin dynamics. pages 681–688, 01 2011.

Jason Yoo, Yunpeng Liu, Frank Wood, and Geoff Pleiss. Layerwise proximal replay: A proximal point method for online continual learning, 2024. URL https://arxiv.org/abs/2402.09542.