Does Catastrophic Forgetting Happen in Tiny Subspaces?

Rufat Asadli Armin Begic Jan Schlegel Philemon Thalmann

Abstract

Catastrophic forgetting remains a significant challenge in continual learning, where adapting to new tasks often disrupts previously acquired knowledge. Recent studies on neural network optimization indicate that learning in a non-continual framework primarily occurs within the bulk subspace of the loss Hessian, which is associated with small eigenvalues of the latter. However, the role of the bulk subspace in a continual learning setting, particularly in relation to forgetting, is not well understood. In this work, we investigate how constraining gradient updates to either the bulk or dominant subspace affects learning and forgetting. Through experiments on Permuted MNIST, Split-CIFAR10, and Split-CIFAR100, we confirm that task-specific learning occurs in the bulk subspace of the loss Hessian. Additionally, there is evidence suggesting that forgetting may also predominantly occur within the bulk subspace, although further large-scale experiments are needed to validate this. Our findings provide promising avenues for efficient implementations of algorithms that counter catastrophic forgetting.[1]

1. Introduction

Continual learning is a machine learning subfield in which a model is tasked with learning from sequentially emerging information while retaining previously gained knowledge (Wang et al., 2024b). However, as first observed in McCloskey & Cohen (1989), a neural network's adaptations to new tasks generally deteriorate its performance on the tasks learned earlier, a phenomenon known as *catastrophic forgetting* (CF). The prevalence of CF has since led to the introduction of a plethora of continual learning algorithms, many of which aim to modify the optimization program in a way that steers parameter updates toward directions that preserve previously acquired knowledge. Frequently, this involves restricting parameter updates to subspaces that are deemed less likely to interfere with prior tasks.[2] Therefore, understanding which subspaces are critical for learning, which subspaces are associated with forgetting, and how these subspaces interact with one another is of significant interest. Theoretical research (Lanzillotta et al., 2024) demonstrates that when certain conditions on the solutions of prior tasks are satisfied, aligning the parameter updates for new tasks with the null space of the average Hessian of previous tasks can effectively prevent forgetting. However, recent findings on neural network optimization by Song et al. (2024) show that within a non-continual framework, learning happens in the so-called *bulk subspace*, determined by eigenvectors corresponding to small eigenvalues (as opposed to the *dominant subspace*, cf. Section 2). Because this partitioning into bulk and dominant subspace has yet to be explored from a continual learning perspective, it remains unclear whether these low eigenvalues are similarly significant in a continual setting. Therefore, this project aims to fill that gap by investigating the effect of restricting parameter updates to either the *bulk subspace* or the *dominant subspace* on learning and forgetting. We hypothesize that the findings of Song et al. (2024) extend to continual learning scenarios, with each task being learned within its respective bulk subspace. Consequently, since task-specific learning occurs in these bulk subspaces, we further hypothesize that this leads to interference between tasks and, ultimately, to forgetting also happening primarily in the bulk subspaces.

2. Methods and Models

More formally, we henceforth adapt the notation of Song et al. (2024) to the continual learning setting and consider for some task $\tau \in \{1, \dots, \mathcal{T}\}$ a training loss function $\mathcal{L}_\tau : \mathbb{R}^p \to \mathbb{R}, \theta_\tau \mapsto \mathcal{L}_\tau(\theta_\tau)$ and its Hessian $\nabla^2 \mathcal{L}_\tau(\theta_\tau) \in \mathbb{R}^{p \times p}$. Moreover, denote with $\lambda_1(\theta_\tau), \lambda_2(\theta_\tau), \dots, \lambda_p(\theta_\tau)$ the eigenvalues of $\nabla^2 \mathcal{L}_\tau(\theta_\tau)$ in decreasing order and with $u_1(\theta_\tau), u_2(\theta_\tau), \dots, u_p(\theta_\tau)$ the corresponding eigenvectors. We define the top-k *dominant subspace* for task τ at θ_τ as

$$S_k(\theta_\tau) := \operatorname{span}\{u_i(\theta_\tau) : i \in [k]\},$$

and the *bulk subspace* for task τ at θ_τ as its orthogonal complement

$$S_k^\perp(\theta_\tau) := \operatorname{span}\{\omega \in \mathbb{R}^p : \omega^\top u_i(\theta_\tau) = 0 \quad \forall : i \in [k]\}.$$

[1] The code is available under https://github.com/JHSchlegel/cf-tiny-subspaces

[2] Compare, e.g., Wang et al. (2024a) for a survey, especially Section 4.3 for a review of optimization-based algorithms.

Furthermore, as $\{u_1(\theta_\tau), \ldots, u_k(\theta_\tau)\}$ form an orthonormal basis for $S_k(\theta_\tau)$, the projection matrix onto $S_k(\theta_\tau)$ can be defined as

$$P_k(\theta_\tau) := \sum_{j=1}^{k} u_j(\theta_\tau) u_j(\theta_\tau)^\top \in \mathbb{R}^{p \times p}$$

and the projection matrix onto $S_k^\perp(\theta_\tau)$ as $P_k^\perp(\theta_\tau) := \mathbf{I}_p - P_k(\theta_\tau)$. Hence, any vector $\omega \in \mathbb{R}^p$ can be decomposed into $\omega = P_k(\theta_\tau)\omega + P_k^\perp(\theta_\tau)\omega$, where in previous work, k is generally chosen to correspond to the number of classes in the dataset. How this choice of k translates to a continual learning setting is investigated in Section 3.

Finally, for analyzing subspace dynamics and quantifying the overlap between the subspaces of different tasks, we adopt the overlap metric introduced in Gur-Ari et al. (2018). Accordingly, let $S_k^{(t)}(\theta_\tau)$ be the top-k *dominant subspace* for task τ at training step $t \in \mathbb{N}_0$ spanned by $u_1^{(t)}(\theta_\tau), \ldots, u_k^{(t)}(\theta_\tau)$, and $P_k^{(t)}(\theta_\tau)$ the corresponding projection matrix onto $S_k^{(t)}(\theta_\tau)$. The overlap $O(\cdot, \cdot)$ between the subspace $S_k^{(t)}(\theta_\tau)$ and $S_k^{(t')}(\theta_{\tau'})$ at training step $t' \geq t$ and task $\tau' \geq \tau$ is defined as

$$O\left(S_k^{(t)}(\theta_\tau), S_k^{(t')}(\theta_{\tau'})\right) := \frac{\mathrm{Tr}\left(P_k^{(t)}(\theta_\tau) P_k^{(t')}(\theta_{\tau'})\right)}{\sqrt{\mathrm{Tr}\left(P_k^{(t)}(\theta_\tau)\right) \mathrm{Tr}\left(P_k^{(t')}(\theta_{\tau'})\right)}} = \frac{1}{k} \sum_{j=1}^{k} \left\| P_k^{(t)}(\theta_\tau) u_j^{(t')}(\theta_{\tau'}) \right\|_2^2 \in [0, 1],$$

where each element of the sum specifies the proportion of $u_j^{(t')}(\theta_{\tau'})$ that belongs to $S_k^{(t)}(\theta_\tau)$ (Gur-Ari et al., 2018). Hence, the overlap between two identical subspaces is 1 and it is 0 between two orthogonal subspaces.

Concretely, inspired by Song et al. (2024), we expect that for *every task*, learning happens primarily in its respective *bulk subspace*. Therefore, as described in Algorithm 1 in Appendix A, we commence by performing standard SGD updates to train our network on data $\mathcal{D}_1$ from the first task. For all subsequent task datasets $\mathcal{D}_2, \ldots, \mathcal{D}_\mathcal{T}$, we perform parameter updates according to Dom-SGD or Bulk-SGD algorithms proposed in Song et al. (2024), resulting in what we define as CL-Dom-SGD and CL-Bulk-SGD, respectively. Specifically, at every training step t, we obtain the top-k eigenvectors in a scalable manner by leveraging Lanczos' method and Hessian-vector products as is implemented in Noah Golmant (2018). To update the parameters, we then project the SGD gradient vectors onto $S_k(\theta_\tau)$ in CL-Dom-SGD and onto $S_k^\perp(\theta_\tau)$ in CL-Bulk-SGD (where we omit the step index t for ease of notation).

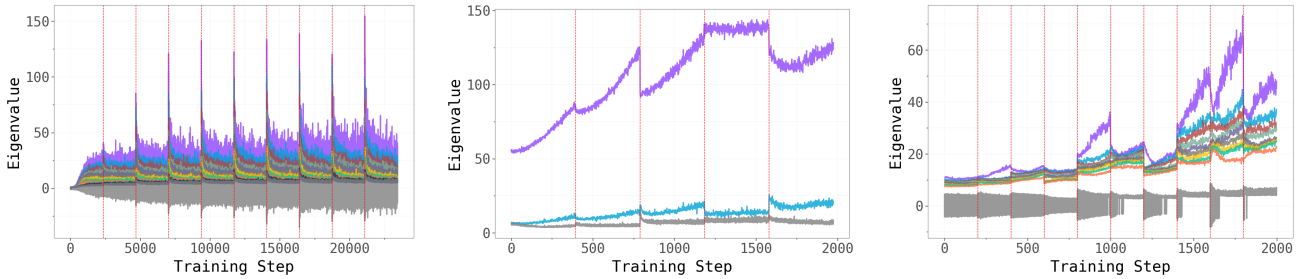
3. Results

To investigate the relationship between Hessian subspaces and the dynamics of learning and forgetting, we compare CL-Dom-SGD and CL-Bulk-SGD with a baseline model trained sequentially using standard SGD in terms of the metrics mentioned in Appendix B.3. To be consistent with the existing continual learning literature (cf., e.g., Van de Ven et al., 2022), we employ datasets for both domain (Permuted MNIST) and task (Split-CIFAR10 and Split-CIFAR100) incremental learning. The specific details of each setting including dataset configurations and task definitions are described in Appendix B.2.

To guide the selection of the choice of k used in CL-Bulk-SGD and CL-Dom-SGD, we examine the eigenvalue behavior when training with standard SGD. Figure 1 displays the progression of the top-k and next-top-k eigenvalues over all training steps for all three datasets. We observe that the relationship proposed by Gur-Ari et al. (2018), where k corresponds to the number of classes in the dataset, does not translate directly to the continual learning setting. Instead, when choosing k such that the top-k eigenvalues are notably larger than the remaining ones, we obtain $k = 10$ for Permuted MNIST and $k = 2$ for Split-CIFAR10, aligning with the number of classes per task. For Split-CIFAR100, however, $k = 9$ appears to provide a better fit.

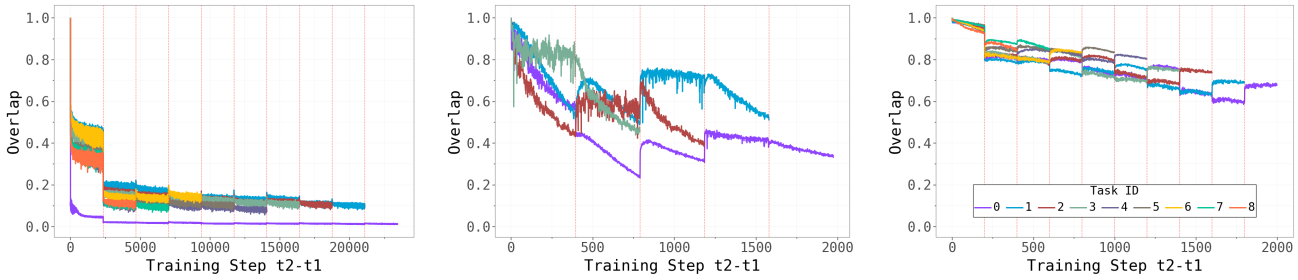
Furthermore, we investigate subspace behavior by examining the overlap of the top-k dominant subspaces across training steps, as shown in Figure 2. Each trajectory illustrates how the top-k dominant subspace at the beginning of a task overlaps with the top-k dominant subspace at a later training step. Concretely, for two given training steps t_1 and t_2, with $t_2 \geq t_1$, the curve depicts $O\left(S_k^{(t_1)}(\theta_\tau), S_k^{(t_2)}(\theta_{\tau'})\right)$, where $\forall \tau, \tau' \in \{1, \ldots, \mathcal{T}\} : \tau' \geq \tau$.

Across all datasets, we observe abrupt changes in top-k dominant subspace overlaps at task transitions, indicating significant shifts in the loss landscape between tasks. In particular, for Permuted MNIST, the dominant subspace overlap drops to nearly zero when transitioning into the second task. In contrast, for Split-CIFAR10 and Split-CIFAR100, the overlap either considerably increases or decreases at task boundaries. We hypothesize that this differing task boundary behavior stems from the distinct data distributions of each dataset. Specifically, there are substantial shifts in the input distribution for Permuted MNIST, as each task applies a unique permutation to the dataset. In comparison, the inputs for Split-CIFAR10 and Split-CIFAR100 exhibit higher similarity (the latter even includes superclasses with subclasses that share similar characteristics). This similarity likely causes the overlap of the top-k dominant subspaces to increase or remain stable at task boundaries for the Split-CIFAR datasets, whereas it quickly vanishes for Permuted MNIST. Additionally, the model's use of separate classification heads for each task in the Task-IL setting possibly amplifies this effect. However, further investigation is required to validate these hypotheses.



(a) Permuted MNIST (b) Split-CIFAR10 (c) Split-CIFAR100

Figure 1. **Evolution of the largest eigenvalues of the loss Hessian**. The plots illustrate the top-$2k$ eigenvalues of the loss Hessian for the three different learning scenarios — Permuted MNIST (left), Split-CIFAR10 (center), and Split-CIFAR100 (right) — each trained using standard SGD. Each colored curve represents one of the top-k eigenvalues, while the light grey lines depict the subsequent top-k eigenvalues. The red-dashed lines delineate the distinct tasks.



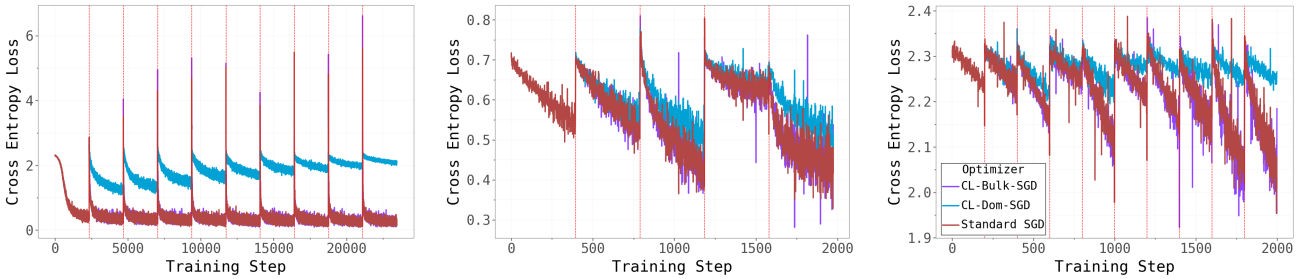
(a) Permuted MNIST (b) Split-CIFAR10 (c) Split-CIFAR100

Figure 2. **Overlap of top-k dominant subspaces $S_k^{(t_1)}(\theta_\tau)$ and $S_k^{(t_2)}(\theta_{\tau'})$**, with $\tau' \geq \tau$. The plots illustrate how the top-k dominant subspaces evolve over the course of three different learning scenarios — Permuted MNIST (left), Split-CIFAR10 (center), and Split-CIFAR100 (right) — each trained using standard SGD. Each colored trajectory shows how the top-k dominant subspace from the beginning of one task continues to overlap with the top-k dominant subspace at subsequent training steps, extending to the end of training. To enhance clarity, only trajectories spanning across multiple tasks are shown. The red-dashed lines delineate the distinct tasks.

Also, within Split-CIFAR10, we observe considerable variability in the magnitude of the dominant subspace overlap within individual tasks. This observation aligns with the findings of Gur-Ari et al. (2018), who report similar behavior when utilizing a simple CNN on the non-sequential CIFAR10 dataset. In contrast, both Permuted MNIST and Split-CIFAR100 demonstrate notably stable within-task overlaps, with Split-CIFAR100 maintaining consistently high overlap values across all tasks and timesteps.

To explore where subspace learning and forgetting occur, we analyze the effectiveness of the CL-Bulk-SGD and CL-Dom-SGD learning algorithms. Table 1 displays a systematic evaluation of the effectiveness of the CL-Bulk-SGD and CL-Dom-SGD learning algorithms in terms of average accuracy and average maximum forgetting, as defined in Appendix B.3. For a fair comparison, no experience replay mechanisms are incorporated, and parameter updates strictly omit momentum and weight decay. The joint-multitask baseline corresponds to an oracle procedure trained using standard SGD with simultaneous access to all tasks' data, serving as an upper bound on achievable performance. Additionally, to precisely attribute learning and forgetting to the respective subspaces, we include a "standard SGD" baseline trained incrementally on all tasks using standard SGD updates. Further details regarding the experimental setup can be found in Appendix B.

Our analysis reveals patterns consistent with our hypothesis that learning for each task predominantly occurs within the designated bulk subspace. Specifically, the average accuracy metrics for both standard SGD and CL-Bulk-SGD closely align across all continual learning scenarios considered. Additionally, we conduct a comprehensive ablation study of the CL-Bulk-SGD algorithm in Appendix D, demonstrating relatively consistent learning behavior across most of the hyperparameters under consideration. Similarly, the training loss trajectories for standard SGD and CL-Bulk-SGD, illustrated in Figure 3, are virtually indistinguishable. Furthermore, it is evident that for CL-Dom-SGD, the loss trajectories are higher and the accuracy figures are considerably lower for each of the experiments we conduct. For the



(a) Permuted MNIST (b) Split-CIFAR10 (c) Split-CIFAR100

Figure 3. **Training loss under different optimizers**. The plots illustrate the evolution of the train cross-entropy loss over three different learning scenarios — Permuted MNIST (left), Split-CIFAR10 (center), and Split-CIFAR100 (right) — with red-dashed lines marking task boundaries. The first task was trained using standard SGD for all methods, after which training proceeded with either standard SGD, CL-Bulk-SGD, or CL-Dom-SGD.

Table 1. **Average accuracy (%) and average maximum forgetting (%) on continual learning benchmarks**. Joint-multitask refers to an oracle baseline that can access all data of all tasks simultaneously.

METHOD	ACCURACY	FORGETTING
PERMUTED MNIST (DOMAIN-IL):		
JOINT-MULTITASK	96.598	0
STANDARD SGD	64.075	31.309
CL-BULK-SGD	62.090	33.500
CL-DOM-SGD	45.721	8.011
SPLIT-CIFAR10 (TASK-IL)		
JOINT-MULTITASK	79.470	0
STANDARD SGD	76.200	0.600
CL-BULK-SGD	76.230	0.475
CL-DOM-SGD	71.590	1.237
SPLIT-CIFAR100 (TASK-IL)		
JOINT-MULTITASK	46.600	0
STANDARD SGD	23.940	1.289
CL-BULK-SGD	24.150	1.533
CL-DOM-SGD	21.770	0.322

Domain-IL setting, the results are in line with our hypothesis that forgetting primarily manifests in the bulk subspace. In the Task-IL setting, we do not observe a clear forgetting pattern. Rather, we obtain very small forgetting combined with small accuracies on both Split-CIFAR datasets.

4. Discussion

The low accuracy and forgetting on the Split-CIFAR tasks is likely due to the use of a small network, necessitated by computational constraints, which may limit both representational capacity and the reliability of observed forgetting patterns. Also, the accuracy is limited by using standard SGD as an optimizer instead of a more sophisticated optimizer involving adaptive learning rates (e.g., ADAM by Kingma, 2014) combined with relatively few optimization steps. However, we also note that previous research found similar average accuracy on 10-task Split-CIFAR100 when not including experience replay (Janson et al., 2023). Interestingly, while their accuracy figures are comparably low, they observe higher values of forgetting. This discrepancy is possibly due to the use of different, more complex network architectures.

Hence, future work could examine whether our findings generalize to more heavily parametrized networks such as the ResNet-18 architecture, which is widely used in continual learning literature. Understanding whether our findings extend to large-scale networks could advance the development of continual learning algorithms. One promising approach could involve projecting the gradient vectors onto the bulk subspace of the current task while maintaining orthogonality with the bulk subspaces of previous tasks.

5. Summary

This work investigates the role of Hessian subspaces in the context of continual learning. Through experiments on Permuted MNIST, Split-CIFAR10, and Split-CIFAR100, we show that recent findings on neural network optimization extend to the continual learning setting, with task-specific learning predominantly occurring in the bulk subspace of the task specific loss Hessian. In the investigated Domain-IL scenario, both learning and forgetting are concentrated in the bulk subspace, providing strong support for the hypothesis that this subspace is central to task-specific updates and interference. While for Task-IL the task-specific learning is also concentrated in the bulk subspace, no clear pattern emerges regarding forgetting, highlighting the need for further investigation and larger scale experiments. These findings emphasize the dual role of bulk subspaces in enabling learning and contributing to task interference in continual learning frameworks.

References

Chaudhry, A., Khan, N., Dokania, P. K., and Torr, P. H. S. Continual learning in low-rank orthogonal subspaces, 2020. URL https://arxiv.org/abs/2010.11635.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability, 2022. URL https://arxiv.org/abs/2103.00065.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Graldi, J., Lanzillotta, G., Noci, L., Grewe, B. F., and Hofmann, T. To learn or not to learn: Exploring the limits of feature learning in continual learning. In *NeurIPS 2024 Workshop on Scalable Continual Learning for Lifelong Foundation Models*, 2024. URL https://openreview.net/forum?id=TYPBYgWyw8.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace, 2018. URL https://arxiv.org/abs/1812.04754.

Janson, P., Zhang, W., Aljundi, R., and Elhoseiny, M. A simple baseline that questions the use of pretrained-models in continual learning, 2023. URL https://arxiv.org/abs/2210.04428.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lanzillotta, G., Singh, S. P., Grewe, B. F., and Hofmann, T. Local vs global continual learning, 2024. URL https://arxiv.org/abs/2407.16611.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Noah Golmant, Zhewei Yao, A. G. M. M. J. G. pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition, October 2018. URL https://github.com/noahgolmant/pytorch-hessian-eigenthings.

Song, M., Ahn, K., and Yun, C. Does sgd really happen in tiny subspaces?, 2024. URL https://arxiv.org/abs/2405.16002.

Van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application, 2024a. URL https://arxiv.org/abs/2302.00487.

Wang, Z., Li, Y., Shen, L., and Huang, H. A unified and general framework for continual learning, 2024b. URL https://arxiv.org/abs/2403.13249.

A. Algorithm

Algorithm 1 presents the procedure for performing Bulk-SGD (cf. Song et al., 2024) parameter updates in the continual learning framework. The algorithm for Dom-SGD is identical apart from projecting onto the dominant subspace instead of the bulk subspace.

Algorithm 1 CL-Bulk-SGD

Require: Task sequence $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\mathcal{T}\}$, initial parameters θ_0, learning rate η

Train without restriction on $\mathcal{D}_1$ to obtain parameters θ_1
for $\tau = 2$ to $\mathcal{T}$ **do**
  $\theta_\tau \leftarrow \theta_{\tau-1}$
  for $n = 1$ to `num_epochs` **do**
    $B \leftarrow$ `len(trainloader(`$\mathcal{D}_\tau$`))`
    for $b = 1$ to B **do**
      Obtain top-k eigenvectors of $\nabla^2 \mathcal{L}_{\tau,b}(\theta_\tau)$: $u_1(\theta_\tau), \ldots, u_k(\theta_\tau)$ via Lanczos method & Hessian-vector products
      Compute projection $P_k^\perp(\theta_\tau)$ of $S_k^\perp(\theta_\tau)$ using $u_1(\theta_\tau), \ldots, u_k(\theta_\tau)$
      Update task parameters $\theta_\tau \leftarrow \theta_\tau - \eta P_k^\perp(\theta_\tau) \nabla \mathcal{L}_{\tau,b}(\theta_\tau)$
    end for
  end for
end for

B. Experimental Details

B.1. Data

In accordance with existing continual learning literature (Lanzillotta et al., 2024, Chaudhry et al., 2020), we employ the Split-CIFAR10 and Split-CIFAR100 datasets in a task incremental context. Additionally, we employ the Permuted MNIST dataset in a Domain-IL context. While we are aware that the latter is sometimes considered to be too artificial (Van de Ven et al., 2022), we nevertheless include it to have a dataset with less similarity between tasks.

Permuted MNIST

We use the Permuted MNIST dataset, a variant of the original MNIST handwritten digits. Permuted MNIST increases task complexity by applying a unique, fixed random permutation to the pixel sequences for each task (Goodfellow et al., 2013). Our experiments consist of 10 tasks, each leveraging the full MNIST dataset with its own consistent permutation applied to all images within the task.

Split-CIFAR10 and Split-CIFAR100

CIFAR10 and CIFAR100 each contain color images of 10 and 100 classes, respectively (Krizhevsky et al., 2009). For our experimental continual learning setup, we split CIFAR10 into 5 tasks and CIFAR100 into 10 tasks, ensuring that classes do not overlap between tasks.

B.2. Experimental Setup

In continual learning, three primary scenarios are used to address non-stationary data streams (a detailed description is available in Van de Ven et al., 2022). In task-incremental learning (*Task-IL*), the goal is to sequentially solve a series of distinct tasks, where the task identity is explicitly available (also at evaluation time). As a result, the mapping involves both the input and the context to produce task-specific outputs. In domain-incremental learning (*Domain-IL*), the problem structure remains the same, but there is a distribution shift in the input across tasks. In this scenario, the mapping is independent of task labels and focuses solely on the input-output relationship. In class-incremental learning (*Class-IL*), the challenge lies in distinguishing between an expanding set of classes over time without relying on explicit task labels. This requires the model to map inputs to outputs in a unified class space (Van de Ven et al., 2022).

For a given dataset, the continual learning problem can often be framed within one of these three scenarios. For instance, Permuted MNIST aligns naturally with Domain-IL due to its input distribution shifts. For Split-CIFAR10 and Split-CIFAR100, we follow prior research (e.g., Graldi et al., 2024) and frame them as Task-IL problems.

The implemented model varies depending on the CL scenario. Specifically, in the Domain-IL setting we use an MLP with two hidden layers, each consisting of 100 ReLU units. For Split-CIFAR10 and Split-CIFAR100 under Task-IL, we use the two-layer CNN, with 32 output channels in each layer, presented in Cohen et al. (2022) with a separate classification head (a fully connected layer atop the convolutional layers) for each task (see, Graldi et al., 2024 or Van de Ven et al., 2022). This distinction between the convolutional "body", shared across tasks, and the task-specific classification heads, which are isolated to prevent interference, directly impacts how gradient projections are performed in CL-Dom-SGD and CL-Bulk-SGD. In our implementation, which we validated experimentally, we calculate the Hessian with respect to all model parameters. However, for Task-IL, only the gradients of the convolutional layers are projected onto the dominant or bulk subspace of the Hessian, which ensures that task-specific components remain unaffected.

We train all our models for 5 epochs with a batch size of 128. Due to computational limitations, we subsample 2,000 observations of the current task data loader to approximate the loss Hessian. This approach yields sufficiently accurate estimates of the top-k eigenvalues and eigenvectors, which are critical for our algorithms that strongly depend on the resulting subspaces.

B.3. Metrics

After completing the continual learning sequence, we assess model performance using two metrics: average accuracy and average maximum forgetting (Chaudhry et al., 2020). The average accuracy metric captures the model's overall performance across all tasks $\tau \in \{1, \ldots, \mathcal{T}\}$ after training has concluded

$$\text{Accuracy} = \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} a_{\mathcal{T},\tau},$$

where $a_{i,\tau}$ represents the test accuracy achieved on task τ after completing training on task i.

To quantify the degree of forgetting, we employ the above mentioned average maximum forgetting metric, which measures the average decrease in performance from peak accuracy across tasks

$$\text{Forgetting} = \frac{1}{\mathcal{T}-1} \sum_{\tau=1}^{\mathcal{T}-1} \max_{l \in \{1,\ldots,\mathcal{T}-1\}} (a_{l,\tau} - a_{\mathcal{T},\tau}).$$

This metric reflects how much knowledge of earlier tasks is lost during the sequential learning process by comparing each task's peak performance to its final performance after training on all tasks.

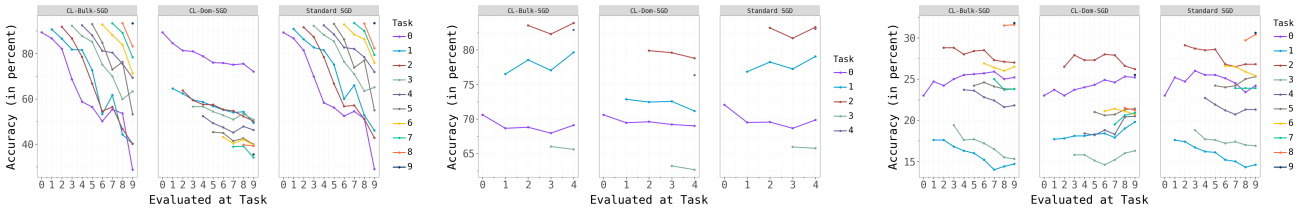
C. Complementary Visualizations

Drawing from our analysis, we visualize how test accuracy evolves across sequential tasks in Figure 4, where each scenario compares three training approaches: CL-Bulk-SGD (left), CL-Dom-SGD (center), and standard SGD (right). The plots reveal a stark contrast between Domain-IL and Task-IL scenarios. In Domain-IL (Permuted MNIST), we observe substantial degradation in accuracy for earlier tasks as new tasks are being learned, emphasizing catastrophic forgetting to a noticeable extent. In contrast, Task-IL scenarios (Split-CIFAR10 and Split-CIFAR100) exhibit remarkably stable performance such that once a task is learned, its accuracy remains largely preserved over the model's learning process across the subsequent tasks.

To further investigate the underlying dynamics, in Figure 5, we examine the evolution of the largest eigenvalue during training with CL-Bulk-SGD (left), CL-Dom-SGD (center), and standard SGD (right). Most notably, both CL-Bulk-SGD and standard SGD yield the largest eigenvalue to be of a greater scale and volatility, particularly pronounced in the Permuted MNIST scenario where we observe regular spikes corresponding to task transitions. On the other hand, CL-Dom-SGD consistently shows lower eigenvalue magnitudes across all datasets.

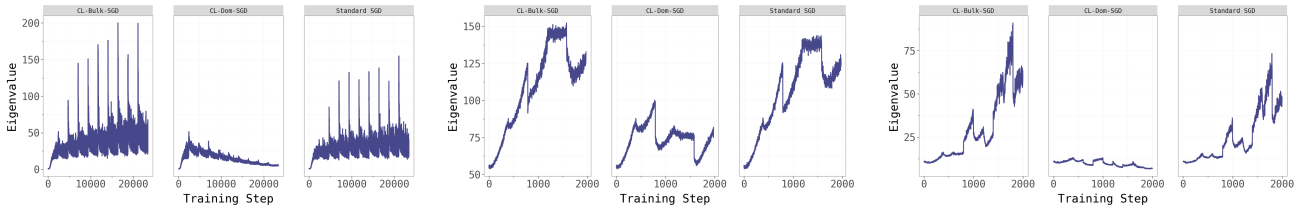
D. Bulk Subspace Ablations

We conduct an ablation study focusing on the batch size, learning rate, and model width to investigate the impact of different hyperparameter settings on the loss surface when using the CL-Bulk-SGD algorithm presented in Algorithm 1. These variations are evaluated across the Permuted MNIST (Domain-IL), Split-CIFAR10 (Task-IL), and Split-CIFAR100 (Task-IL) datasets.



(a) Permuted MNIST (b) Split-CIFAR10 (c) Split-CIFAR100

Figure 4. **Test accuracy across sequential tasks**. Each curve traces how that task's accuracy changes as new tasks are learned using CL-Bulk-SGD, CL-Dom-SGD and standard SGD stratified by different learning scenarios — Permuted MNIST (left), Split-CIFAR10 (center), and Split-CIFAR100 (right).



(a) Permuted MNIST (b) Split-CIFAR10 (c) Split-CIFAR100

Figure 5. **Evolution of the largest eigenvalue during training**. This figure illustrates the evolution of the largest eigenvalue when employing CL-Bulk-SGD, CL-Dom-SGD and standard SGD across the three different learning scenarios — Permuted MNIST (left), Split-CIFAR10 (center), and Split-CIFAR100 (right).

Model Architectures: For Permuted MNIST (Domain-IL), we utilize an MLP with two hidden layers with ReLU units, varying the hidden dimension during width ablations. For Split-CIFAR10 (Task-IL) and Split-CIFAR100 (Task-IL), we employ the same two-layer CNN as for the other experiments, which is based on the one described in Cohen et al. (2022), with task-specific classification heads. Therein, we modify the number of output channels of the two convolutional layers to assess width effects.

Hyperparameter Variations: Each ablation is performed by altering one hyperparameter at a time while keeping the others fixed. More precisely, for Permuted MNIST (Domain-IL), we test hidden dimensions in $\{50, 100, 200\}$, batch sizes in $\{128, 256, 512\}$, and learning rates in $\{0.001, 0.01, 0.1\}$. For Split-CIFAR10 (Task-IL) and Split-CIFAR100 (Task-IL), on the other hand, we experiment with $\{16, 32, 64\}$ output channels of every convolutional layer, batch sizes in $\{128, 256, 512\}$ and learning rates in $\{0.0001, 0.001, 0.01\}$. Finally, we dynamically adjust the number of epochs to keep the number of optimization steps constant across different batch sizes, where, for the default batch size of 128, we train every task for five epochs across all datasets.

Results: Table 2 reports average accuracy and average maximum forgetting under varying model width, batch size, and learning rate for our proposed CL-Bulk-SGD algorithm on Permuted MNIST (Domain-IL), Split-CIFAR10 (Task-IL) and Split-CIFAR100 (Task-IL). In general, adjusting model width and batch size have a relatively mild effect on performance. However, deviating from the default learning rate causes sharp decreases in accuracy and noticeable increases in forgetting.

In the course of our experiments, we observe that the batch size and the number of subsamples used to compute the top-k eigenvalues via Hessian-vector products and the Lanczos method also significantly influence performance, with some parameter settings even causing convergence issues in the eigendecomposition. We employ 2,000 subsamples consistently across all experiments on all three datasets and find that, within our computation limitations, stable and satisfactory performance is achieved when using a batch size of 500 for both Split-CIFAR10 and Split-CIFAR100, and a batch size of 32 for Permuted MNIST. Moreover, we are constrained to utilizing a single batch per Lanczos iteration instead of processing the entire subsampled task training dataloader. Therefore, a more comprehensive investigation into the parameter configuration for the Hessian-vector product approximation would be natural, given sufficient computational resources.

In addition, our ablation study reveals that, across most of the hyperparameter configurations we examine, the observed dominant subspace overlap patterns remain largely consistent. As shown in Figure 6 (Permuted MNIST), Figure 7 (Split-

Table 2. **CL-Bulk-SGD ablation study results**. Average accuracy (%) and average maximal forgetting (%) across different hyperparameter combinations for Permuted MNIST (Domain-IL), Split-CIFAR10 (Task-IL), and Split-CIFAR100 (Task-IL). Each dataset's first row gives the **default** hyperparameters. Subsequent rows show performance for the two modified values.

DATASET	HYPERPARAMETER	VALUE	ACCURACY	FORGETTING
PERMUTED MNIST	**DEFAULT: HIDDENDIM=100, BATCHSIZE=128, LR=0.01**		62.090	33.500
	HIDDEN DIM.	50	60.065	35.348
		200	65.449	30.052
	BATCH SIZE	256	62.535	33.030
		512	61.978	33.678
	LEARNING RATE	0.001	53.596	17.126
		0.1	15.045	79.106
SPLIT-CIFAR10	**DEFAULT: #OUTCHAN=32, BATCHSIZE=128, LR=0.001**		76.230	0.475
	# OUT CHANNELS	16	73.320	0.125
		64	79.380	0.325
	BATCH SIZE	256	76.050	0.400
		512	76.440	0.375
	LEARNING RATE	0.0001	58.640	0.325
		0.01	69.200	19.438
SPLIT-CIFAR100	**DEFAULT: #OUTCHAN=32, BATCHSIZE=128, LR=0.001**		24.150	1.533
	# OUT CHANNELS	16	19.830	0.900
		64	29.630	0.778
	BATCH SIZE	256	24.350	1.244
		512	24.380	1.322
	LEARNING RATE	0.0001	10.260	0.200
		0.01	28.980	3.756

CIFAR10), and Figure 8 (Split-CIFAR100), the learning rate seems to most noticeably influence the dominant subspace overlap trajectory, with higher learning rates producing lower overlap magnitudes and sharper drop-offs. Overall, the top-k dominant subspace overlap behavior from CL-Bulk-SGD ablations strongly resembles that displayed in Figure 2 under standard SGD.

Finally, Figures 9, 10, and 11 display the evolution for the top-k largest eigenvalues when training on Permuted MNIST, Split-CIFAR10, and Split-CIFAR100 with CL-Bulk-SGD for various hyperparameter configurations. We observe that large fluctuations in the top-k eigenvalues arise primarily when the learning rate is increased, causing instability in the eigenvalue estimates. In the case of Permuted MNIST, the fluctuations when adjusting the learning rate are especially pronounced, likely due to the small batch size of 32 used for the Hessian-vector products in each Lanczos step.

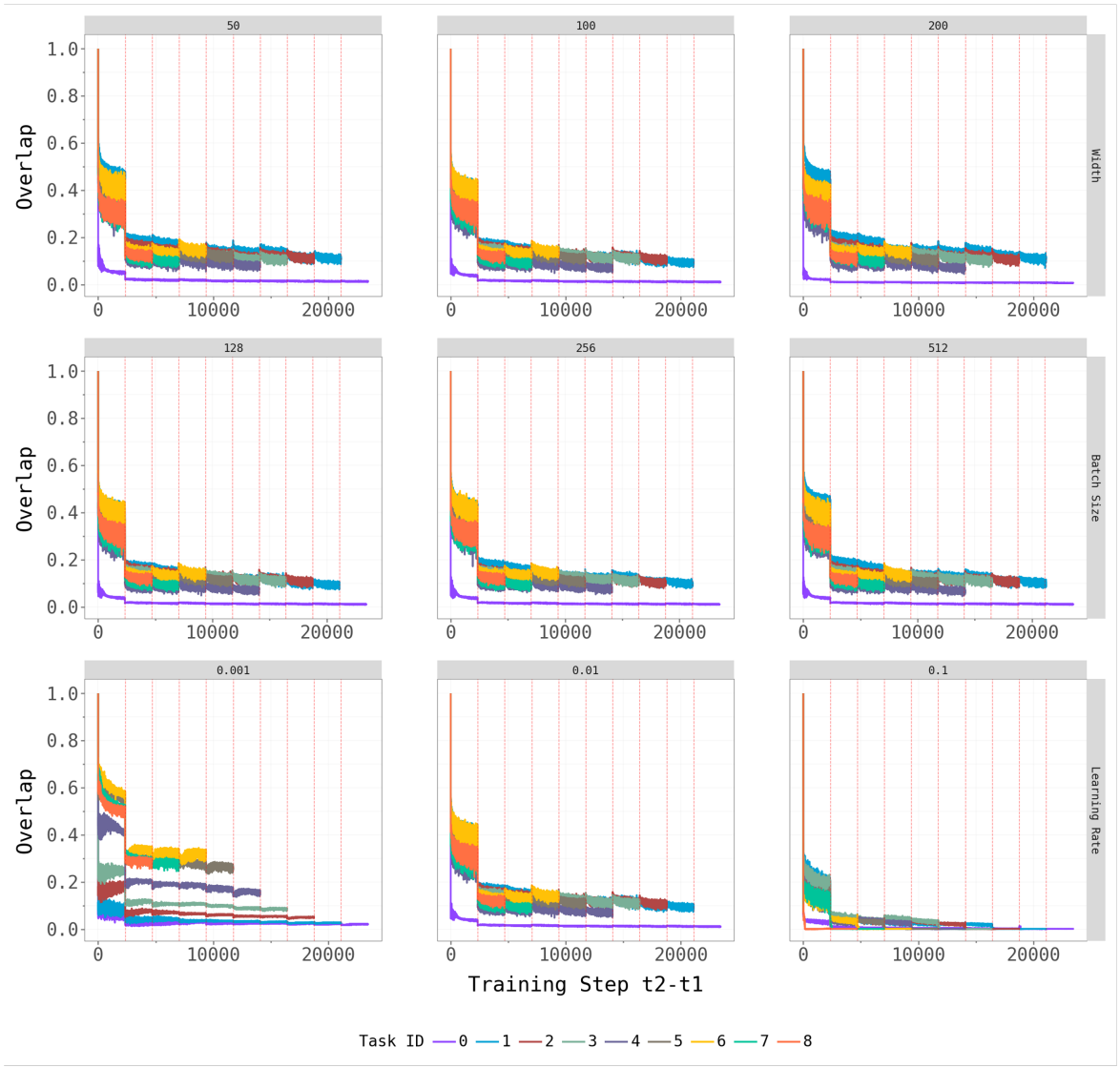


Figure 6. **Ablation Study: Top-k overlap for Permuted MNIST**. The plots illustrate how the top-k dominant subspaces evolve during training on the Permuted MNIST dataset using CL-Bulk-SGD optimizer for different hyperparameter settings, where each trajectory tracks the overlap between initial task dominant subspace and that of subsequent training steps. The top row varies the hidden dimension $\{50, 100, 200\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.001, 0.01, 0.1\}$. The red-dashed lines indicate the distinct tasks during training, and different colors represent different task IDs.

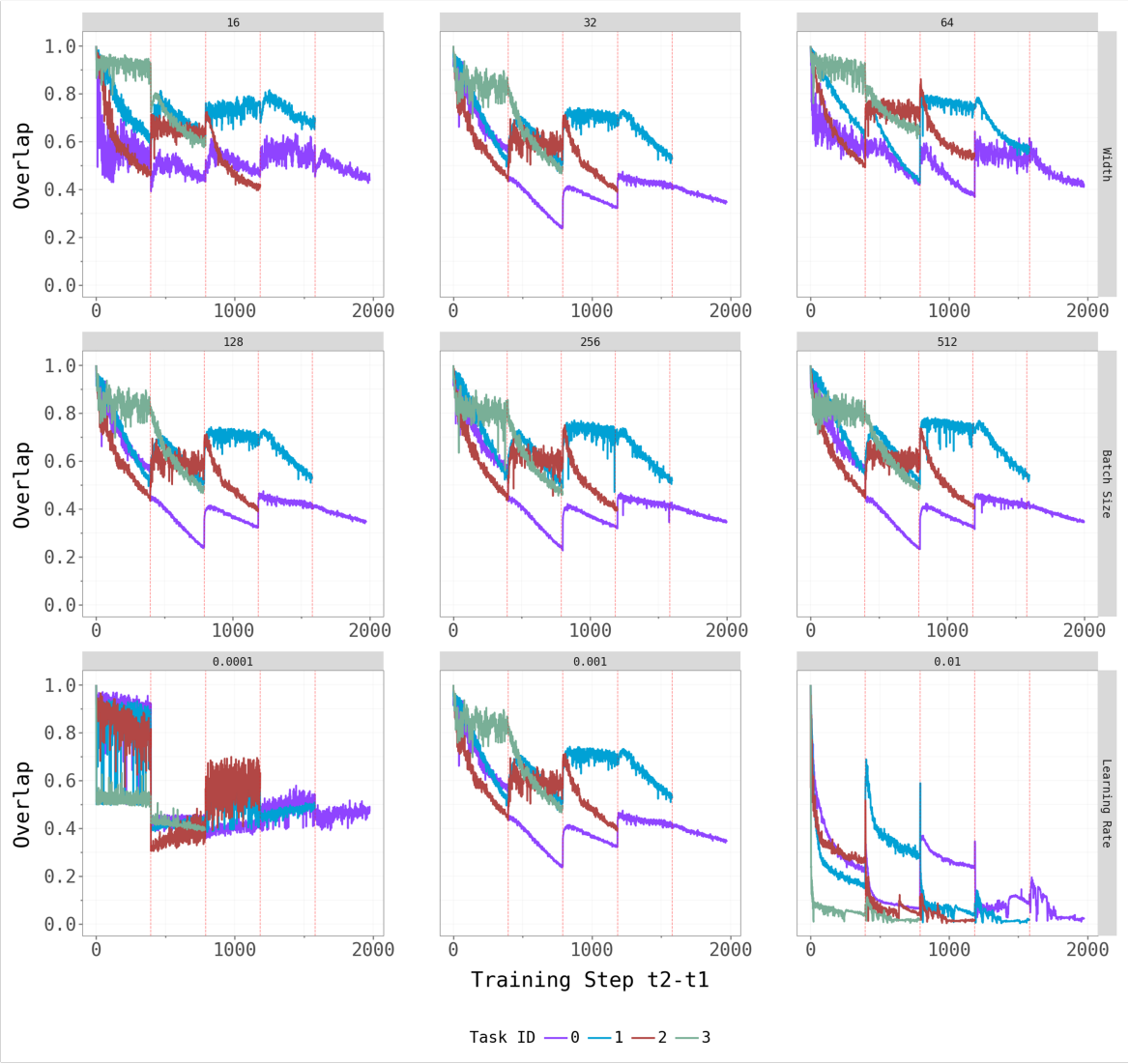


Figure 7. **Ablation Study: Top-k overlap for Split-CIFAR10**. The plots illustrate how the top-k dominant subspaces evolve during training on the Split-CIFAR10 dataset using CL-Bulk-SGD optimizer for different hyperparameter settings, where each trajectory tracks the overlap between initial task dominant subspace and that of subsequent training steps. The top row varies the number of output channels $\{16, 32, 64\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.0001, 0.001, 0.01\}$. The red-dashed lines indicate the distinct tasks during training, and different colors represent different task IDs.

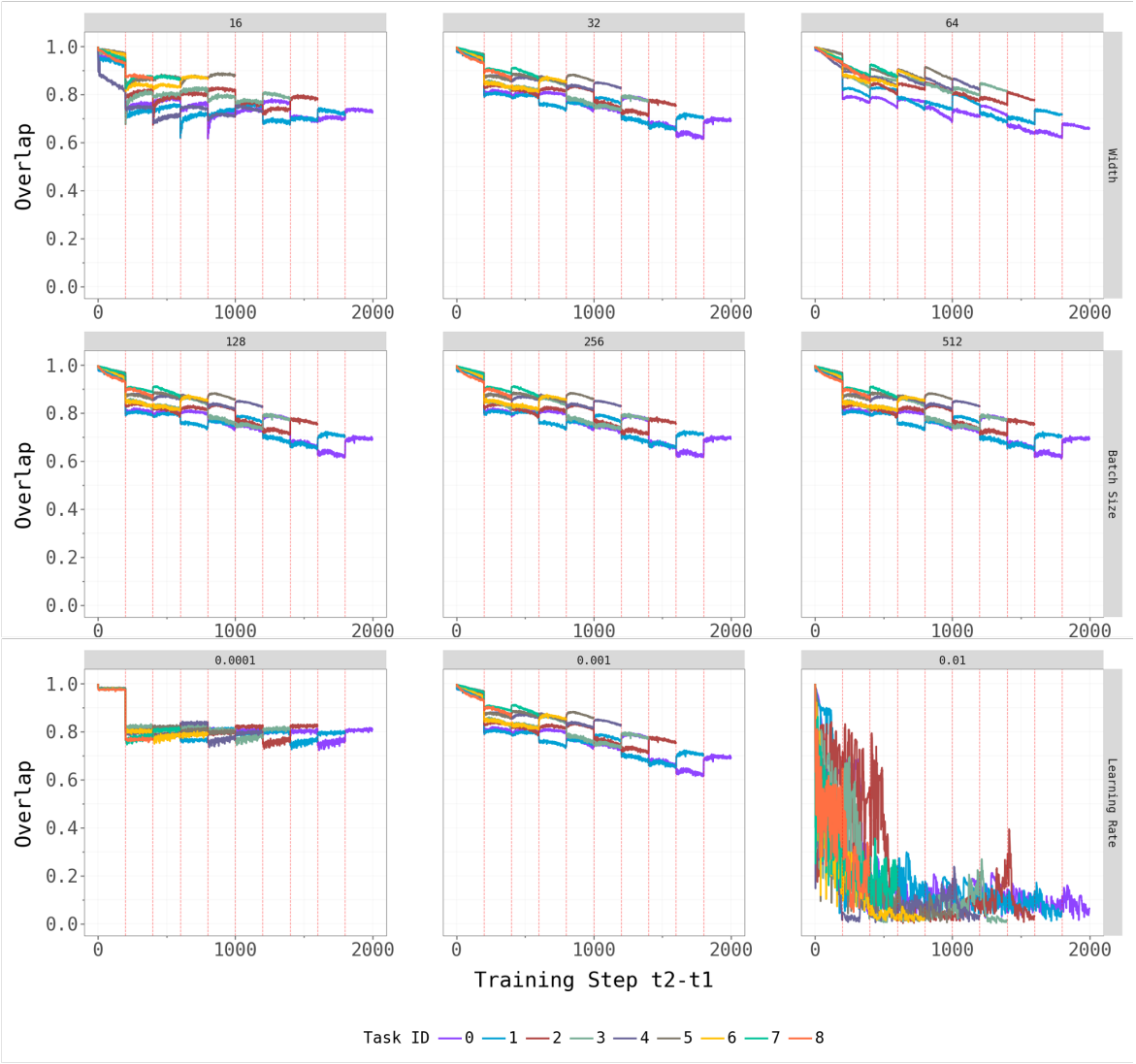


Figure 8. **Ablation Study: Top-k overlap for Split-CIFAR100**. The plots illustrate how the top-k dominant subspaces evolve during training on the Split-CIFAR100 dataset using CL-Bulk-SGD optimizer for different hyperparameter settings, where each trajectory tracks the overlap between initial task dominant subspace and that of subsequent training steps. The top row varies the number of output channels $\{16, 32, 64\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.0001, 0.001, 0.01\}$. The red-dashed lines indicate the distinct tasks during training, and different colors represent different task IDs.

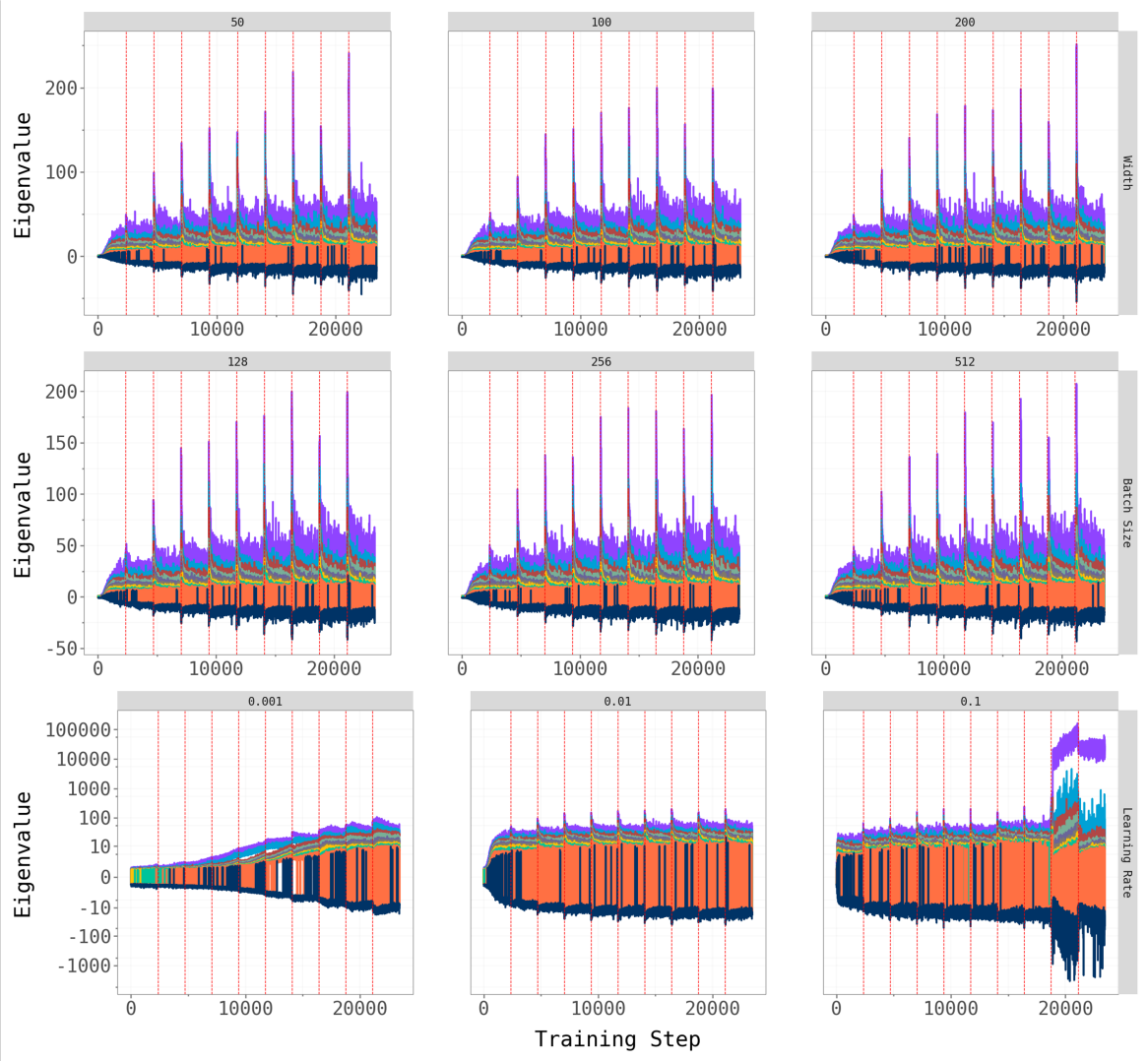


Figure 9. **Ablation Study: Top-k eigenvalues for Permuted MNIST**.The plots illustrate the evolution of top-k eigenvalues during training on the Permuted MNIST dataset using CL-Bulk-SGD optimizer for different hyperparameter settings. The top row varies the hidden dimension $\{50, 100, 200\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.001, 0.01, 0.1\}$ (symmetric logarithmic scale). The red-dashed lines indicate distinct tasks during training.

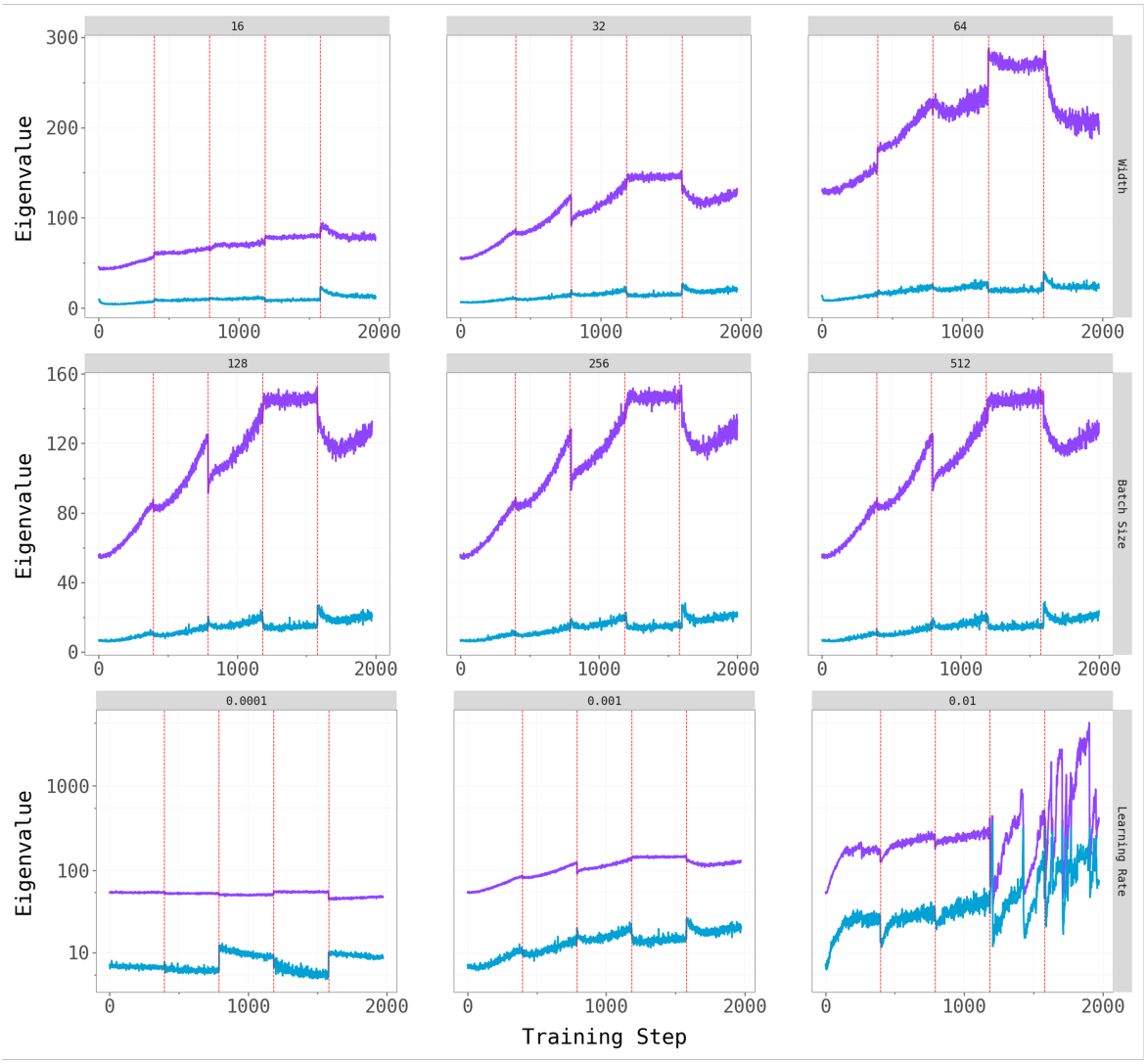


Figure 10. **Ablation Study: Top-k eigenvalues for Split-CIFAR10**. The plots illustrate the evolution of top-k eigenvalues during training on the Split-CIFAR10 dataset using CL-Bulk-SGD optimizer for different hyperparameter settings. The top row varies the number of output channels $\{16, 32, 64\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.0001, 0.001, 0.01\}$ (symmetric logarithmic scale). The red-dashed lines indicate distinct tasks during training.

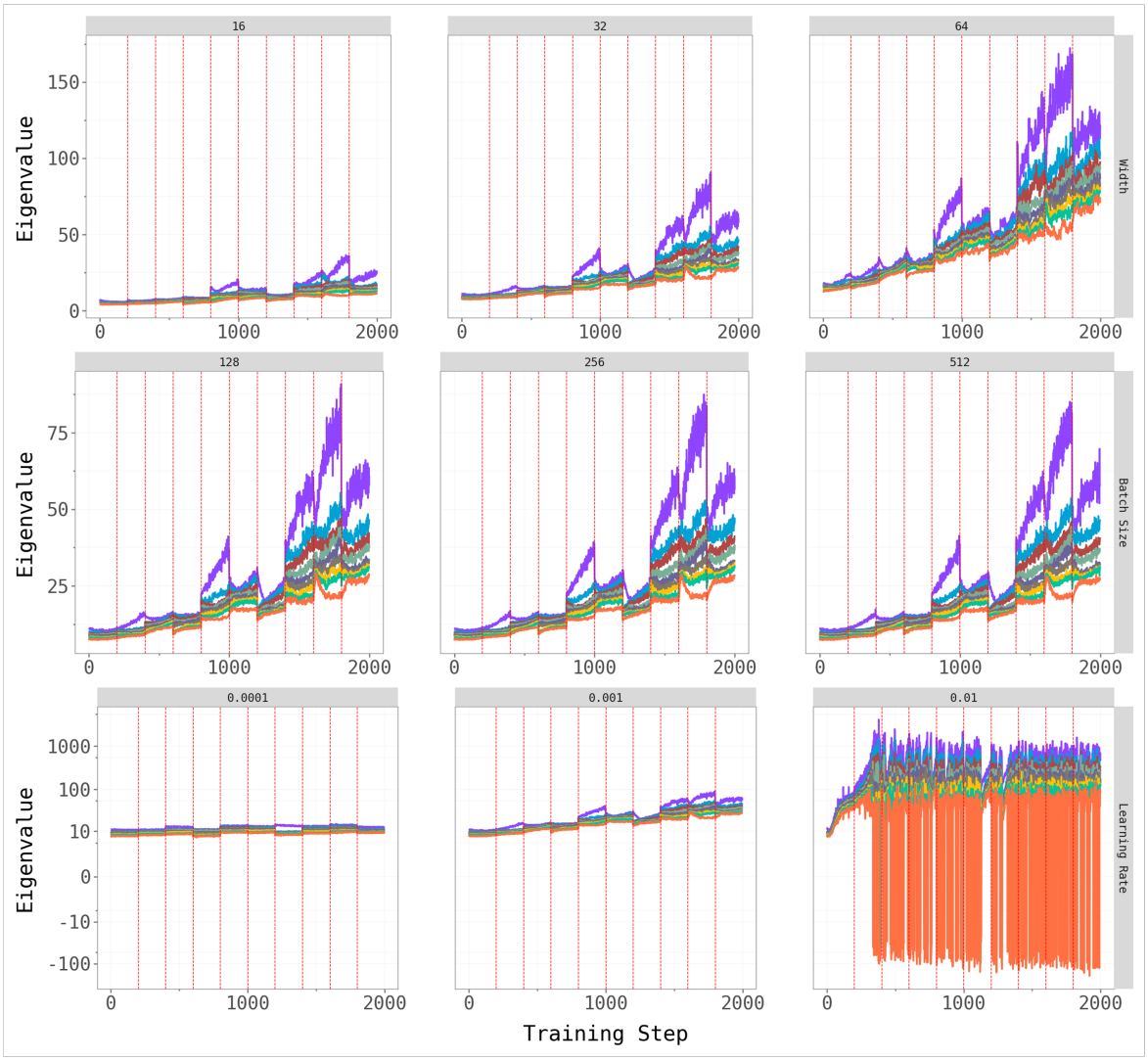


Figure 11. **Ablation Study: Top-k eigenvalues for Split-CIFAR100**. The plots illustrate the evolution of top-k eigenvalues during training on the Split-CIFAR100 dataset using CL-Bulk-SGD optimizer for different hyperparameter settings. The top row varies the number of output channels $\{16, 32, 64\}$, the middle row shows different batch sizes $\{128, 256, 512\}$, and the bottom row displays different learning rates $\{0.0001, 0.001, 0.01\}$ (symmetric logarithmic scale). The red-dashed lines indicate distinct tasks during training.