Does Catastrophic Forgetting Happen in Tiny Subspaces?

Rufat Asadli Armin Begic Jan Schlegel Philemon Thalmann

1. Background

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.[1] 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* (as opposed to the *dominant subspace*, cf. Section 2). This partition into bulk and dominant subspace has not yet been explored from a continual learning perspective. Therefore, this project aims to fill this gap by investigating the effect of restricting parameter updates to either the *bulk subspace* or the *dominant subspace* on learning and forgetting.

2. Approach

More formally, we henceforth adapt the notation of Song et al. (2024) to the continual learning setting and consider for some task $t \in \{1, \dots, \mathcal{T}\}$ a training loss function $\mathcal{L}_t : \mathbb{R}^p \to \mathbb{R}, \theta_t \mapsto \mathcal{L}_t(\theta_t)$ and its Hessian $\nabla^2 \mathcal{L}_t(\theta_t) \in \mathbb{R}^{p \times p}$. Moreover, denote with $\lambda_1(\theta_t), \lambda_2(\theta_t), \dots, \lambda_p(\theta_t)$ the eigenvalues of $\nabla^2 \mathcal{L}_t(\theta_t)$ in decreasing order and with $u_1(\theta_t), u_2(\theta_t), \dots, u_p(\theta_t)$ the corresponding eigenvectors.

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

We then define the top-k *dominant subspace* for task t at θ_t as $S_k(\theta_t) := \text{span}\{u_i(\theta_t) : i \in [k]\}$ and the *bulk subspace* for task t as its orthogonal complement $S_k^\perp(\theta_t)$. In previous work, k is generally chosen to correspond to the number of classes in the dataset. We will investigate how this translates if the data stems from multiple sequential tasks.

Concretely, inspired by Song et al. (2024), we expect that for *every task*, learning happens primarily in its respective *bulk subspace*. Hence, 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 $\mathcal{D}_2, \dots, \mathcal{D}_\tau$, we perform parameter updates according to Dom-SGD or Bulk-SGD algorithms proposed in Song et al. (2024). Specifically, at every SGD step, we calculate the Hessian for the current batch and obtain its top-k eigenvectors in a scalable manner by leveraging Lanczos method and Hessian-vector products as is implemented in Noah Golmant (2018). Based on these, we compute the dominant subspace $S_k(\theta_t)$ and its orthogonal complement $S_k^\perp(\theta_t)$, followed by projecting the SGD gradient updates onto $S_k(\theta_t)$ in CL-Dom-SGD and onto $S_k^\perp(\theta_t)$ in CL-Bulk-SGD to update the parameters. We then compare CL-Dom-SGD and CL-Bulk-SGD with a baseline model trained sequentially using standard SGD in terms of the metrics mentioned in Section 3. This way we hope to better understand the connection between subspace-restricted learning and forgetting. Using these insights, we plan on including information about the subspaces of previous tasks into our parameter updates. One possibility we have in mind is to enforce orthogonality of gradient updates to the relevant subspaces of previous tasks.

3. Evaluation

In line with existing continual learning literature, we will assess the effectiveness of learning within the bulk and the dominant subspaces separately on the permuted `MNIST`, `Split-CIFAR10` and `Split-CIFAR100`. For the permuted `MNIST`, a MLP will be used, and for the `Split-CIFAR10` and `Split-CIFAR100` experiments, a small CNN will be employed. Finally, catastrophic forgetting will be quantified by means of average accuracy and average maximum forgetting.

References

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.

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 planned 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 $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\tau\}$, initial parameters θ_0, learning rate η

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