RETHINKING CROSS-TASK GENERALIZATION IN CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) aims to sequentially learn tasks with reduced forgetting and improved cross-task generalization. Although the two goals are usually treated separately in previous works, our theoretical analysis shows that improving cross-task generalization helps reduce forgetting. Previous works address the cross-task generalization problem by introducing large-scale reference data or freezing the pre-trained weights. However, the former is computationally expensive, while the latter does not assess the model's generalization ability in full fine-tuning. To improve the cross-task generalization with full parameter tuning, we find that parameters with largest (weight) updates in fine-tuning can be vulnerable in cross-task generalization and suffer more from forgetting, while parameters that have consistent updates in pre-training and fine-tuning are more robust. Inspired by this, we propose a double filtering mask to filter weight updates in fine-tuning that are inconsistent with pre-trained weight magnitudes. Although simple to implement, our method significantly improves model's cross-task generalization and achieves comparable performance to CL models that require large-scale reference data.

1 INTRODUCTION

Continual learning (CL) aims to learn models sequentially from a stream of tasks. The main challenges of CL include: (1) preventing catastrophic forgetting, where models forget previous knowledge when learning new tasks (McCloskey & Cohen, 1989; French, 1999); (2) improving current models' zero-shot performance on future tasks (Lopez-Paz & Ranzato, 2017), which requires models' generalization on unseen, out-of-distribution (OOD) data.

Most CL methods tackle the two challenges separately, possibly because the optimal setting for reducing forgetting and increasing generalization is not always the same (Lin et al., 2023a). However, our theoretical analysis shows that improving models' cross-task generalization after each task is an effective way to reduce forgetting: with less generalization error, models will distort less of representations when learning new tasks and have less forgetting. Although pre-trained models like CLIP (Radford et al., 2021) greatly improve models' generalization ability and alleviate forgetting Mehta et al. (2023), such ability may degrade in sequential fine-tuning (Zheng et al., 2023).

Previous CL methods address the cross-task generalization issue by preserving models' pre-trained knowledge. Zheng et al. (2023) distills knowledge on large-scale reference data to reduce pre-trained feature deviation; Tang et al. (2024); Xu et al. (2024) freeze pre-trained weights and introduce additional task-specific blocks to learn new tasks. However, the former method requires intensive computation on the additional data; while the latter freezes pre-trained weights and thus does not assess the pre-trained model's cross-task generalization capacity in full fine-tuning. In this paper, we study the following research question:

Without freezing the pre-trained model or using additional data/task-specific blocks, can we improve model's cross-task generalization to reduce forgetting in CL?

To address this problem, we first study weight updates in fine-tuning that are generalizable across tasks. Recent CL models like MagMax (Marczak et al., 2024) merge the largest weight updates across tasks, because they are the most important in preserving task knowledge. However, are the large updates of weights in one task generalizable to other tasks in CL? When fine-tuning a pre-trained model in a task, parameters whose weights are updated most usually have small-magnitude weights

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

in pre-training (zhou et al., 2025). In pre-training, parameters with small-magnitude weights may be unimportant or not well-trained (Appendix B). Therefore, they may not be able to provide a generalizable reference in fine-tuning. Our experiments show that large weight updates of parameters, whose pre-trained weights have small magnitudes, may not be generalizable across tasks, especially when there are strong distribution shift.

Inspired by the above observations, we filter weight updates that are less generalizable across tasks during each task's learning. Specifically, we use pre-trained weights as a reference to find a mask that filters out (1) large updates corresponding to small-magnitude weights in pre-training; (2) small updates corresponding to large-magnitude weights in pre-training, which is shown to benefit generalization in zhou et al. (2025). Our method is simple to implement: the algorithm to find the mask is learning free; the filtering process only needs to be operated limited times in each task's learning. However, it significantly improves model's generalization ability and reduces forgetting, achieving comparable performance to the model that is fine-tuned with large-scale reference data.

In summary, our main contributions are:

1. We theoretically analyze the effect of cross-task generalization in mitigating forgetting.
2. We design experiments to show that parameters with large weight updates in tasks are vulnerable in cross-task generalization.
3. Based on our observation, we filter less generalizable weight updates in each task's learning by a double filtering mask, which reduces forgetting without using additional data or freezing the pre-trained model.

2 Related Work

Continual learning. Continual learning (CL) studies how models learn from non-stationary data streams without catastrophically forgetting previously acquired knowledge (McCloskey & Cohen, 1989; French, 1999; Parisi et al., 2019; van de Ven et al., 2022). Traditional methods are commonly grouped into regularization-based (Kirkpatrick et al., 2017; Zenke et al., 2017; Li & Hoiem, 2016; Aljundi et al., 2018), replay-based (Lopez-Paz & Ranzato, 2017; Rebuffi et al., 2017; Chaudhry et al., 2019b;a; Buzzega et al., 2020; Aljundi et al., 2019b;a; Prabhu et al., 2020), and architecture-based approaches (Rusu et al., 2016; Mallya & Lazebnik, 2018; Serra et al., 2018). Within them, the most relevant works to ours find subnetworks in one model to reduce task interference (Wortsman et al., 2020; Konishi et al., 2023; Kang et al., 2022). However, those works do not learn subnetworks/masks for pre-trained models. Recently, pre-trained models like CLIP (Radford et al., 2021) have improved capacity to learn tasks, which also alleviates forgetting (Mehta et al., 2023). In this paper, we explore the way to mask parameters based on the relationship between their pre-trained weights and task weight updates, to extract generalizable weight updates and mitigate forgetting.

Continual learning generalization. Previous works study generalization in CL along with forgetting (Lin et al., 2023b). However, they focus on in-task generalization on test data of all seen tasks (Belkin et al., 2018; Bartlett et al., 2020; Hastie et al., 2022; Mitra, 2019; Ju et al., 2020; Mei & Montanari, 2022; Arora et al., 2019; Ju et al., 2022). Nowadays, pre-trained models greatly improve the model's overall generalization ability and makes zero-shot learning on unseen tasks feasible (Devlin et al., 2019; Radford et al., 2021). Our work studies cross-task generalization on unseen, out-of-domain tasks, which is closely related to models' zero-shot transfer capacity. Existing works treat model's zero-shot transfer in CL as an additional goal besides forgetting (Lopez-Paz & Ranzato, 2017; Zheng et al., 2023; Tang et al., 2024). However, our theoretical analysis shows that better cross-task generalization helps reduce forgetting. Moreover, previous works achieve this goal by utilizing additional reference data for distillation (Zheng et al., 2023) or freezing the pre-trained encoder and injecting task-specific blocks (Tang et al., 2024; Xu et al., 2024). The former requires intensive extra computation; the latter may not fully utilize the model's generalization capacity. Compared to them, our work does not require extra data. It explores generalizable weight updates in fine-tuning, which enables all parameter learning and filters less generalizable updates after that.

Continual learning via model merging. Nowadays, model merging (Wortsman et al., 2022a; Matena & Raffel, 2022; Wortsman et al., 2022b; Ilharco et al., 2023; Yadav et al., 2023; Wang et al., 2024) becomes an effective tool to combine multiple weight updates into a single model. In CL,

MagMax (Marczak et al., 2024) merges largest weight updates of each task to reduce forgetting. However, parameters that have large weight updates in a task usually have small magnitude weights in pre-training (zhou et al., 2025). Since the small magnitude weights may be less important or not well-trained during pre-training, they may not be generalizable across tasks. In this paper, we analyze the generalization issue of merging large updates in CL and propose a method to relieve this issue.

3 Analysis and Motivation

In CL, a model sequentially learns tasks $1, ..., N$ where each task comes once at a time. We do not use previous data for replay. The goal is to learn one model that solves all tasks at the end, which requires to minimize catastrophic forgetting during sequential learning. Following settings of previous works (Evron et al., 2022; Lin et al., 2023b; Li et al., 2025), we first conduct a theoretical analysis on a linear model to show that cross-task (OOD) generalization contributes to forgetting mitigation.

3.1 Linear Model Setting

For each task i, we have the training data $\mathcal{D}_i = \{\mathbf{X}_i, \mathbf{y}_i\}$ where the input $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ is a matrix with n_i samples of d dimensional feature vectors; $\mathbf{y}_i \in \mathbb{R}^{n_i}$ contains n_i scalar values. We use a linear model $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}, \mathbf{w} \in \mathbb{R}^d$ to solve all tasks, and denote the weight learned in task i as $\mathbf{w}_i$.

Assumption. Assume the model has enough capacity and there is a ground truth weight $\mathbf{w}^* \in \mathbb{R}^d$ that fits all N tasks' data. Therefore, for any task i we have the target $\mathbf{y}_i = \mathbf{X}_i\mathbf{w}^*$. The loss in task i is $\left\|\hat{\mathbf{y}}_i - \mathbf{y}_i\right\|_2^2$. To simplify notations, here we do not use the average loss.

3.2 Cross-Task Generalization Helps Reduce Forgetting

For two sequential tasks i and $(i+1)$ in CL, the model starts from an initial weight and sequentially updates the weight at each task. The two CL metrics we discuss are: (1) the cross-task generalization error $gen(\mathbf{w}_{i-1})$ of the initial weight $\mathbf{w}_{i-1}$, which is $\mathbf{w}_{i-1}$'s average loss on two tasks' data; (2) the forgetting $fgt(\mathbf{w}_{i+1})$ of the last weight $\mathbf{w}_{i+1}$, which is $\mathbf{w}_{i+1}$'s loss on the first task's data. Below, we show that the model's cross-task generalization ability plays a role in reducing forgetting.

Analysis 1 *Given the initial weight $\mathbf{w}_{i-1}$, we sequentially learn tasks i and $(i+1)$ with inputs $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ using Gradient Descent. Each input $\mathbf{X}_i$ satisfies $\mathbf{X}_i^T\mathbf{X}_i = \mathbf{Q}_i\mathbf{\Omega}_i{\mathbf{Q}_i}^T$, where $\mathbf{Q}_i \in \mathbb{R}^{d\times d}$ is an orthonormal basis and $\mathbf{\Omega}_i = diag(o_i^1, o_i^2, ..., o_i^d)$ is a diagonal matrix. The initial weight $\mathbf{w}_{i-1}$'s cross-task (OOD) generalization error on tasks i and $(i+1)$ is bounded by:*

$$gen(\mathbf{w}_{i-1}; \mathbf{X}_i, \mathbf{X}_{i+1}) \leq (1/2)\big(\max(\mathbf{\Omega}_i) + \max(\mathbf{\Omega}_{i+1})\big)\left\|\mathbf{w}^* - \mathbf{w}_{i-1}\right\|_2^2, \tag{1}$$

The forgetting of task i after learning task $(i+1)$ is bounded by:

$$fgt(\mathbf{w}_{i+1}; \mathbf{X}_i) \leq \underbrace{\left\|\mathbf{w}^* - \mathbf{w}_{i-1}\right\|_2^2}_{\textit{Generalization related}} \underbrace{\left\|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{P}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\right\|_2^2}_{\textit{Feature alignment, } \neq \mathbf{0}}, \tag{2}$$

where α is the learning rate, $\mathbf{P}_{i+1} = \mathbf{Q}_i^T\mathbf{Q}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_{i+1})^t\mathbf{Q}_{i+1}^T\mathbf{Q}_i$ is the transformation between two basis $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$, t is the GD steps of learning both tasks, $\left\|\cdot\right\|_2$ represents the L2-norm.

The proof is in Appendix A. In Eq. (2), the forgetting is bounded by two factors: the alignment between input features and the distance between the optimal and initial weight, which also decides the initial weight's cross-task generalization error in Eq. (1).

Feature alignment mitigates forgetting. One way to reduce forgetting is to better align features across tasks. When $\mathbf{Q}_i = \mathbf{Q}_{i+1}$ (features have the same basis) or $\mathbf{Q}_i \perp \mathbf{Q}_{i+1}$ (features are orthogonal) which make $\mathbf{Q}_i^T\mathbf{Q}_{i+1} = \mathbf{I}$, we have Eq. (2) equal to 0. In other cases, the feature space of task i will be distorted after task $(i+1)$ and result in forgetting. Previous works (Bai et al., 2024; Hu et al., 2026) address feature alignment by forcing features to be orthogonal or finding a global basis for all tasks' features. However, the former reduces knowledge transfer and the latter is highly dependent on the pre-trained model's capacity.

Cross-task generalization is a key. Without explicitly aligning the feature space, the forgetting in Eq. (2) can still be small when the distance $\left\|\mathbf{w}^* - \mathbf{w}_{i-1}\right\|_2^2$ between the optimal and initial weights are small. In this case, the model does not have to deviate far to learn new tasks, reducing the effect of distortion. According to Eq. (1), this requires small cross-task generalization error on the weight after each task's learning, since it is the initial weight of the following tasks.

We validate the effect of cross-task generalization by mixing the multi-task learning (MTL) weights in initialization, which is regarded as generalizable weights for the corresponding CL task (Mirzadeh et al., 2020). In Table 1, pre-trained models with weights $\boldsymbol{\theta}_{pre}$ have less forgetting on split CIFAR100 when mixed with MTL weights $\boldsymbol{\theta}_{MTL}$ in initialization. This supports that improved generalization can help reduce forgetting, even for large-scale pre-trained models.

Table 1: MTL mix-up rate and forgetting. The model is initialized as $(1-\alpha)\boldsymbol{\theta}_{pre} + \alpha\boldsymbol{\theta}_{MTL}$ and sequentially fine-tuned.

Model	$\alpha = 0$	0.1	0.2	0.3	0.4
ViT B/32	1.60	1.39	0.71	0.67	0.55
ViT B/16	2.86	1.52	1.23	1.04	0.34

To improve model's cross-task generalization ability, we first identify weight updates in sequential learning that are generalizable across tasks. Based on our observations, we introduce a method that filtering out less generalizable updates during training, which improves model's cross-task generalization and reduces forgetting.

4 Generalizable Weight Updates in CL

In this section, we identify weight updates that are generalizable across tasks in sequential fine-tuning.

4.1 Splitting Weight Updates with Reference to Pre-trained Weights

Recent CL methods merge the largest weight updates in each task, because they are the most important in preserving task knowledge (Marczak et al., 2024). However, given a pre-trained model, parameters whose weights are updated most in tasks usually have small-magnitude weights in pre-training (zhou et al., 2025). Since parameters with small magnitude weights may be unimportant or not well-trained in pre-training (Appendix B), they may not be able to provide a generalizable reference in fine-tuning. This raises the following question:

Are the large updates of weights in one task generalizable to other tasks in CL?

To answer this question, in each task of CL we split model parameters into two subsets: (1) **Subset 1**: all parameters excluding Subset 2; (2) **Subset 2**: parameters that have large weight updates but their pre-trained weights' magnitudes are small. A comparison between two subsets of parameters is shown in Fig. 3. Then we only update weights for the selected parameter subset at each task and compare their CL performance.

Formally, for a task i we find the mask for Subset 1 by solving the objective below:

$$\min_{\lambda_i, \mathbf{m}_i} \left\|\lambda_i \mathbf{m}_i \odot \left|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1}\right| - \left|\boldsymbol{\theta}_{pre}\right|\right\|_2, \tag{3}$$

where $\odot$ is the element-wise multiplication, $\boldsymbol{\theta}_{pre}$ is the pre-trained weights, $\boldsymbol{\theta}_i$ is the parameter weights after fine-tuning task i, $\mathbf{m}_i$ is a 0, 1 mask on the weight updates.

The objective in Eq. (3) finds scaling and masks for weight updates $|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1}|$ to best match the pre-trained weight magnitudes $|\boldsymbol{\theta}_{pre}|$. After λ_i scaling, the mask will be 1 for weight updates which are not too large compared to the pre-trained weights (Subset 1); otherwise the mask will be 0.

The algorithm to compute the masks is in Section 5.1. After finding the mask at each task, we update the parameter weights by $\mathbf{m}_i \odot (\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1})$ for Subset 1 and $(1 - \mathbf{m}_i) \odot (\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i-1})$ for Subset 2.

Experimental Setting. Experiments in Section 4, 5 are conducted on a CLIP ViT-B/32 model on 10-split CIFAR100 dataset. We sequentially train the model in the **task-aware** setting. Results are averaged from 3 random orders. We measure the model's forgetting by the BWT score (Lopez-Paz & Ranzato, 2017) on previous tasks; and cross-task generalization by the model's **task-agnostic** accuracy on unseen tasks (OOT) in CIFAR100 and the accuracy on an unseen, out-of distribution dataset Aircraft. Details of the metrics and datasets are shown in Section 6.1.

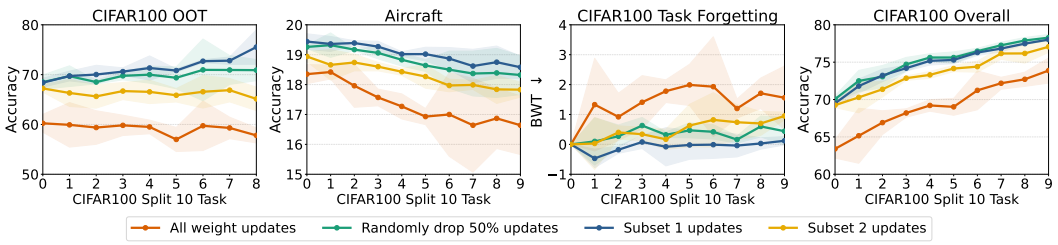

Figure 1: The left two figures show the model's cross-task generalization performence; and the right two figures show the model's forgetting and overall task-agnostic accuracy. Parameters with large updates in tasks but small magnitude weights in pre-training (Subset 2) achieves worse generalization performance and higher forgetting compared to the rest (Subset 1).

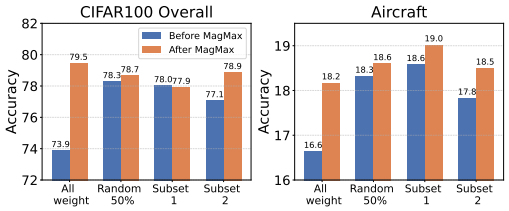

Figure 2: ID and OOD accuracy of parameter subsets before and after applying Magmax.

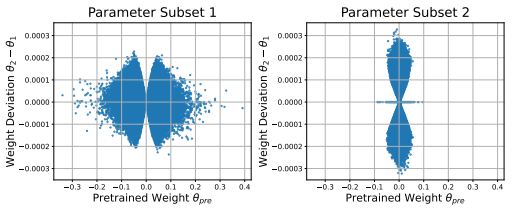

Figure 3: Pre-trained weights and task weight updates of parameter subsets.

4.2 CL Performance of Parameter Subsets

Parameters that change the most during fine-tuning may be vulnerable in cross-task generalization. In Fig. 1, parameters in Subset 2 have consistently worse generalization performance on unseen tasks than those in Subset 1, and also suffer more from forgetting. Taking into account the influence of the tunable parameter numbers, we also compare with randomly masking 50% weight updates. And training with Subset 2 still underperforms that with random masking. We hypothesize that this is because pre-trained weights with small magnitudes may be unimportant or not well-trained in pre-training. An analysis on linear models is shown in Appendix B. Since downstream tasks are usually less general than pre-trained ones, allowing abrupt large updates to small magnitude weights may easily overfit the current task and distort the pre-trained knowledge.

Parameters with consistent changes in fine-tuning and pre-training are more robust in CL. Parameters in Subset 1 achieve better cross-task generalization and less forgetting compared to all other subsets, which suggests that they are more robust in CL. We notice that Subset 1 includes parameters that have large magnitude weights in pre-training, which are preferred to be frozen for better generalization in zhou et al. (2025). Even with updates on these parameters, Subset 1 achieves better cross-task generalization than others. One may think of using only Subset 1 parameters after each task for better CL performance. However, only using Subset 1 updates has slightly worse overall accuracy than random masking, indicating a capacity issue. We address this issue in Section 5.

Merging large weight udpates does not fully resolve the generalization issue. Although large weight updates may be less generalizable, recent CL methods like MagMax (Marczak et al., 2024) merge the largest weight updates of tasks. In Fig. 2, we validate that the parameter subset benefited the most from MagMax is Subset 2, which achieves on par performance to full parameter MagMax on CIFAR100. However, MagMax does not fully resolve the generalization issue of Subset 2/full parameter tuning on unseen tasks. After MagMax, Subset 2 does not outperform Subset 1 (before MagMax)'s generalization performance on Aircraft. Meanwhile, MagMax does not address Subset 1's capacity issue on CIFAR100. This calls for new methods to address the above issues.

5 Method

Based on the above observations, parameters with consistent updates in fine-tuning and pre-training have less forgetting and better cross-task generalization. However, only extracting updates of those

parameters after fine-tuning may degrade the model's performance in that task. To address this issue, we propose a DoubleFilter method that filters inconsistent weight updates several times during task learning, which enables the model to learn after filtering and improve the task performance.

5.1 Computing the Mask to Match Magnitudes

General Objective. As in Section 4.1, we match magnitudes of weights and weight updates, and compute the mask to filter inconsistent weight updates. For future use, we write a general version of our objective that matches a target magnitude $\left|\boldsymbol{\theta}_{target}\right|$ and a reference magnitude $\left|\boldsymbol{\theta}_{ref}\right|$, as below:

$$\min_{\lambda,\mathbf{m}}\left\|\lambda\ \mathbf{m}\odot\left|\boldsymbol{\theta}_{target}\right|-\left|\boldsymbol{\theta}_{ref}\right|\right\|_2, \tag{4}$$

where λ is a scaling factor, $\odot$ is the element-wise multiplication, $\mathbf{m}$ is the mask for the target weight magnitude $\left|\boldsymbol{\theta}_{target}\right|$. θ_{target} and θ_{ref} are different for different kinds of masks.

The objective aims to find a scaling λ and mask $\mathbf{m}$ to best match magnitudes in $\left|\boldsymbol{\theta}_{target}\right|$ and $\left|\boldsymbol{\theta}_{ref}\right|$. Specifically, if the n-th magnitude has $\lambda\left|\boldsymbol{\theta}_{target}[n]\right| > 2|\boldsymbol{\theta}_{ref}[n]|$, the corresponding mask element $\mathbf{m}[n]$ will be 0. Since the magnitudes are non-negative, if $\lambda|\boldsymbol{\theta}_{target}[n]| \leq 2|\boldsymbol{\theta}_{ref}[n]|$, the mask element $\mathbf{m}[n]$ will be 1. Therefore, the objective in Eq. (4) is a one directional mask that filtering magnitudes in $\left|\boldsymbol{\theta}_{target}\right|$ that are greatly *larger* than those in $\left|\boldsymbol{\theta}_{ref}\right|$.

Computing the mask. To jointly learn the scaling λ and the mask $\mathbf{m}$, we apply an iterative algorithm that first initializes $\mathbf{m}^{(0)}$ with all-1 values, then iteratively updates λ and adjusts $\mathbf{m}$ accordingly. At step $s > 0$, the $\lambda^{(s)}$ is updated as:

$$\lambda^{(s)} = \left(\mathbf{m}^{(s-1)}\odot\left|\boldsymbol{\theta}_{target}\right|\right)^T\left(\left|\boldsymbol{\theta}_{ref}\right|\right)/\left\|\left(\mathbf{m}^{(s-1)}\odot\left|\boldsymbol{\theta}_{target}\right|\right)\right\|_2^2,$$

and the mask $\mathbf{m}^{(s)}$ is updated as:

$$\mathbf{m}^{(s)} = \mathbb{1}\{\lambda^{(s)}|\boldsymbol{\theta}_{target}| \leq 2|\boldsymbol{\theta}_{ref}|\}.$$

The algorithm will typically converge in 5 steps. In the following paper, the mask $\mathbf{m}$ used in the updating equation is computed after applying this algorithm.

5.2 Double Filtering Mask

We filter out inconsistent weight updates by a double filtering mask. Specifically, this mask filters out parameters that have (1) large weight updates in tasks but small magnitude weights in pretraining; (2) small weight updates in tasks but large magnitude weights in pre-training. The relationship between filtered and unfiltered parameters are shown in Fig. 4. Based on the magnitude matching objective (Eq. (4)), our double filtering mask is constructed below.

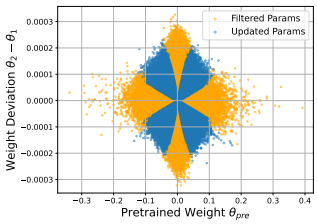

Figure 4: Pre-trained weights and task weight updates of parameters selected by the double filtering mask (blue dots).

Large update mask. As shown in Section 4, we first filter parameters that have large weight updates but small magnitude pre-trained weights. In the t-th iteration of learning a task i, the filtering mask $\mathbf{m}_{large,i}^{(t)}$ is computed by solving the objective below:

$$\min_{\lambda_{large,i}^{(t)},\mathbf{m}_{large,i}^{(t)}}\left\|\lambda_{large,i}^{(t)}\mathbf{m}_{large,i}^{(t)}\odot\left|\boldsymbol{\theta}_i^{(t)}-\boldsymbol{\theta}_{i-1}\right|-\left|\boldsymbol{\theta}_{pre}\right|\right\|_2, \tag{5}$$

where $\lambda_{large,i}^{(t)}$ is the corresponding scaling factor, $\boldsymbol{\theta}_i^{(t)}$ is the model weights at the t-th iteration.

Double filtering mask. Unlike large weight updates, the small weight updates in a task usually correspond to large magnitude weights in pre-training. Previous works suggest that restricting such small updates helps reduce noisy updates (Yadav et al., 2023; Marczak et al., 2024) and preserve the model's pre-trained generalization capacity (zhou et al., 2025). Therefore, after filtering the large weight updates by the mask $\mathbf{m}_{large,i}^{(t)}$, we filter small weight updates with the following objective:

$$\min_{\lambda_i^{(t)},\mathbf{m}_i^{(t)}}\left\|\lambda_i^{(t)}\mathbf{m}_i^{(t)}\odot\left|\boldsymbol{\theta}_{pre}\right|-\mathbf{m}_{large,i}^{(t)}\odot\left|\boldsymbol{\theta}_i^{(t)}-\boldsymbol{\theta}_{i-1}\right|\right\|_2. \tag{6}$$

In the above objective, if the scaled magnitude of a pre-trained weight is much larger than its weight update, the weight update will not be selected by the mask. This satisfies our goal because if a weight update is too small compared to the pre-trained weight magnitude, it does not introduce much task information and thus can be discarded to reduce noisy, less-generalizable updates.

5.3 Learning Procedure

When learning a task, in each iteration we fine-tune all parameters in the model. After certain iterations, we filter inconsistent weight updates by the double filtering mask and keep training after that. At the t-th iteration of task i with filtering, we compute the double filtering mask $\mathbf{m}_i^{(t)}$ by Eq. (6) and update the parameter weight by:

$$\boldsymbol{\theta}_i^{(t)} \leftarrow \boldsymbol{\theta}_{i-1} + \mathbf{m}_i^{(t)} \odot (\boldsymbol{\theta}_i^{(t)} - \boldsymbol{\theta}_{i-1}),$$

where $\boldsymbol{\theta}_i^{(t)}$ is the weight after t iterations' learning, $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{pre}$.

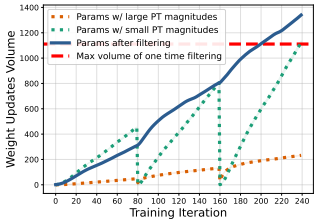

Figure 5: Volume of different parameters' weight updates with three-time filtering.

Besides the end of task learning, we filter weight updates several times during training, which enables the model to learn after filtering and improve task performance. In Fig. 5 we show the volume of weight updates during learning, with three times filtering at the iteration 80,160 and 240 (end of task learning). We define the weight update volume as the sum of each parameter's absolute weight deviation, which reflects the total weight updates for specific parameter groups. After filtering, the remaining parameters tend to update more to compensate discarded updates. We find it unnecessary to filter frequently because within certain iterations, parameters with small PT magnitudes do not have significantly large updates and can contribute to task learning.

6 Experiments

6.1 Experimental Settings

Datasets. We evaluate our models on both class-incremental and task-incremental settings. For both two settings, we train the model in the task-aware learning following Marczak et al. (2024) while evaluating in different settings. For class-incremental evaluation, we test on split image classification datasets CIFAR100, ImageNet-R and CUB200 Marczak et al. (2024); for task-incremental evaluation, we test on Multi-domain Task-Incremental Learning (MTIL) benchmark (Zheng et al., 2023).

Metrics. We follow previous works' CL evaluation metrics: for class-incremental evaluation, we evaluate the model's accuracy on all class data (Marczak et al., 2024); for task-incremental evaluation in MTIL, we compute the *Transfer* and *Last* accuracies for corresponding tasks (Zheng et al., 2023). The *Transfer* score reflects the model's cross-task generalization ability, and the *Last* score is the tasks' average accuracy at the end. We measure the forgetting by the BWT score (Lopez-Paz & Ranzato, 2017), which is the average drop of task accuracies after the last task.

Models. The models we compare are from three categories: (1). the models that do not use extra data or frozen encoder, including EWC (Kirkpatrick et al., 2017), WiSE-FT (Wortsman et al., 2022b), MagMax (Marczak et al., 2024), and our DoubleFilter; (2). the models utilize extra data but not freezing the encoder, including LwF (Li & Hoiem, 2016), ZSCL (Zheng et al., 2023); (3). the model that do not use extra data but use frozen encoder and task-specific blocks, including L2P (Wang et al., 2022b) and S-Prompts (Wang et al., 2022a). Since CL models usually contains multiple techniques for CL, we compare with some variants of more recent models for fair comparison with our model.

Implementation. We build our model based on CLIP ViT-B/16 encoder, use the final classification layer output by CLIP's text encoder and keep it frozen during fine-tuning. Our model does not have additional blocks for different tasks. For class-incremental experiments, we fine-tune the image encoder with a batch size of 128. We train CIFAR100, ImageNet-R for 10 epochs each task and learning rate 1e-5, and CUB200 for 30 epochs (Marczak et al., 2024) and learning rate 5e-5. For MTIL experiments, we use a batch size 64 and train each task for 1000 iterations. We use a learning rate 5e-5 for Aircraft and 1e-5 for all other tasks (Zheng et al., 2023). For our DoubleFilter model, we filter inconsistent updates per 100 iterations and at the end of task learning. In optimization, we apply a cosine annealing learning rate schedule and AdamW optimizer with weight decay 0.1.

Table 2: Class-incremental evaluation on splitted datasets. The **bold** scores are the best and the underline scores are the second best. Our method DoubleFilter is shadowed in grey.

	CIFAR100				ImageNet-R				CUB200		
	/5	/10	/20	/50	/5	/10	/20	/50	/5	/10	/20
Zero-shot	66.91				77.73				56.08		
LwF	83.25	73.45	72.05	68.84	81.15	82.97	81.82	80.32	**65.12**	60.67	58.90
EWC	**84.41**	76.24	75.39	72.97	82.15	82.42	81.48	81.47	59.10	54.49	53.31
WiSE-FT	80.94	77.31	74.40	74.18	81.22	81.50	81.02	81.55	59.30	56.18	55.13
TIES Seq	83.03	80.04	77.77	76.28	83.28	82.50	82.27	81.23	63.32	**60.77**	**59.36**
MAGMAX	84.16	80.41	78.49	**76.75**	83.60	83.33	82.27	**81.75**	63.89	60.74	58.90
DoubleFilter	83.24	**80.76**	**78.66**	76.32	**83.93**	**83.43**	**82.35**	81.43	63.27	59.92	57.37

Table 3: MTIL evaluation. The **bold** scores are the best and the underline scores are the second best.

	Extra data	Frozen	Aircraft	Caltech101	CIFAR100	DTD	EuroSAT	Flowers	Food	MNIST	OxfordPet	Cars	SUN397	Average
Zero-shot			24.8	92.9	68.4	43.8	47.7	71.4	85.8	59.5	89.1	65.8	62.6	64.7
Transfer														
EWC	×	×		74.9	59.1	43.2	45.0	55.7	73.9	52.7	84.7	47.6	53.8	59.1
WiSE-FT	×	×		73.5	55.6	35.6	41.5	47.0	68.3	53.9	69.3	26.8	51.9	52.3
MAGMAX	×	×		74.7	56.0	41.0	36.7	55.2	72.0	54.7	81.5	49.6	55.8	57.7
DoubleFilter	×	×		76.7	61.7	42.0	47.1	63.5	81.7	62.9	85.5	51.8	60.5	63.3
L2P	×	✓		65.6	50.9	30.4	41.4	49.3	71.8	36.3	77.5	55.3	53.4	53.2
S-Prompts	×	✓		67.3	49.4	26.4	39.7	47.1	70.2	34.3	78.9	56.7	52.2	52.2
LwF	✓	×		74.5	56.9	39.1	51.1	52.6	72.8	60.6	75.1	30.3	55.9	56.9
ZSCL	✓	×		86.0	67.4	45.4	50.4	69.1	87.6	61.8	86.8	60.1	66.8	**68.1**
Last														
EWC	×	×	36.5	88.3	74.8	65.2	88.0	64.6	87.3	99.6	91.6	71.8	76.6	76.8
WiSE-FT	×	×	27.2	90.8	68.0	68.9	86.9	74.0	87.6	99.6	92.6	77.8	81.3	77.7
MAGMAX	×	×	40.2	96.1	81.1	72.0	97.8	76.3	88.4	99.2	93.0	70.5	68.9	80.3
DoubleFilter	×	×	48.9	95.0	79.0	72.3	96.6	85.4	90.0	99.2	95.1	79.0	77.9	**83.5**
L2P	×	✓	38.0	87.1	84.2	72.9	86.0	96.1	89.2	99.0	94.1	79.6	76.0	82.0
S-Prompts	×	✓	37.5	95.1	83.7	70.2	97.5	96.5	89.0	99.1	94.0	79.5	75.8	83.4
LwF	✓	×	26.3	87.5	71.9	66.6	79.9	66.9	83.8	99.6	92.1	66.1	80.4	74.6
ZSCL	✓	×	40.6	92.2	81.3	70.5	94.8	90.5	91.9	98.7	93.9	85.3	80.2	**83.6**

6.2 MAIN RESULTS

Class-Incremental Results. In Table 2, in generic classification dataset CIFAR100 and ImageNet-R, our DoubleFilter outperforms other models in 5/8 cases. However, in fine-grained classification dataset CUB200, DoubleFilter does not outperform traditional method like LwF. This may be because the generic classification tasks rely more on the pre-trained knowledge to distinguish image classes (e.g., dog or cat), while CUB200 classifies specific bird species that has less pre-trained information. We hypothesize that when using an effective reference model, DoubleFilter may perform better.

MTIL Results. In Table 3, DoubleFilter significantly outperforms other methods that do not use extra data or freeze the pre-trained encoder. Compared to MagMax, DoubleFilter achieves $\sim$6% improvement in *Transfer* scores and $\sim$3% improvement in *Last* scores. Compare to models with frozen encoder and task-specific block like L2P and S-Prompts, our model also has better scores in both *Transfer* and *Last*. This indicates that DoubleFilter better utilizes pre-trained model's capacity in learning downstream tasks. Compared to ZSCL which uses 100k reference images for distillation, our method achieves comparable CL *Last* performance within a 0.1% difference. However, our method takes significantly less time and no extra data to train.

6.3 ABLATION STUDY

Effect of different masking strategies. In Table 4, we compare different masking strategies to filter weight updates on MTIL. For all strategies, we filter updates every 100 iterations. DoubleFilter achieves better *Transfer* and *Last* scores than just filtering out large weight updates as Eq (.5). This validates the effect of filtering small updates corresponding to large magnitude pre-trained weights.

Table 4: Effect of masking strategies. 'Msk.' is the average masked ratio (%), 'Vol.' is the avg masked volume ratio (%).

Method	Msk.	Vol.	Transfer↑	Last ↑	BWT ↓
DoubleFilter	43	57	63.3	83.5	3.7
Large Update Filter	75	66	62.3	82.9	4.4
Smallest PT	20	53	59.3	81.3	6.0
Rand Drop	50	50	69.2	76.6	1.1
Smallest Update	80	56	69.6	75.9	0.7

Table 5: Effect of DoubleFilter iteration intervals.

Iteration	Transfer ↑	Last ↑	BWT↓
20	65.7	82.7	2.7
100	63.3	83.5	3.7
200	63.8	83.4	3.8
500	65.6	80.9	3.5
1000	67.9	78.0	2.5

We also compare DoubleFilter with three masking strategies which only take the task weight updates or the pre-trained weight magnitudes as a reference. We select different mask ratios with similar update volume for fair comparison. By selecting parameters with smallest 20% pre-trained magnitudes to update, the model achieves lowest scores in Transfer and Last, consistent to our hypothesis that largest weight updates in each task may be less generalizable across tasks. On the other hand, by randomly dropping 50% updates or selecting the smallest 80% updates, models have the capacity issue which forget less but also learn less. These results validate the effect of our DoubleFilter that filter updates based on the relationship between pre-trained weights and task weight updates.

Filtering intervals. Table 5 shows the effect of different filtering intervals on MTIL. When filtering weight updates more frequently (i.e. in 20 iterations), the model has better transfer ability but lower last accuracy. That may confirm our hypothesis in Section 5.3 that although large weight updates may not be generalizable across tasks, they contribute to learning and are important for model capacity. On the other hand, when filtering updates less frequently (in 500, 1000 iterations), the model also suffers from the capacity issue because it has less chance to learn and compensate the discarded updates. We find 100 iterations as the interval achieves stable performance in most cases, including the class-incremental evaluation. Therefore, we use this interval for all tasks without additional hyperparameter searching.

Computation cost. Table 6 compares the computation cost of representative models, d is the number of parameters. Our method is simple and efficient to implement: compared to MagMax, DoubleFilter needs multiple (T) times' operation. However, for each task it only operates limited times (10 for MTIL and $\sim$ 3 for class-incremental) which makes the complexity still controllable. DoubleFilter does not need sorting, so it saves time compared to methods which requires finding smallest weight updates. Compare to ZSCL, DoubleFilter does not need to train with large-scale reference data.

Table 6: Computation cost of models.

	DoubleFilter	Smallest	MagMax	ZSCL
Time	$O(T \cdot d)$	$O(d \log(d))$	$O(d)$	-
Data	0	0	0	100k

7 Conclusion

In this paper, we first conduct a theoretical analysis to show that cross-task generalization contributes to reducing forgetting. Then we study which weight updates in fine-tuning are generalizable across tasks. We find that weight updates consistent (after scaling) with pre-trained weight magnitudes are robust across tasks, achieving better generalization and less forgetting. Based on this, we propose a double filtering method that filters out weight updates that are too large or too small compared to pretrained magnitudes. Experiments show that although simple to implement, our method achieves significant improvement compared to MagMax in MTIL.

The limitations of our model include: (1) it relies on the reference of the pre-trained model. If a task does not require much pre-trained information, our model may not perform well; and (2) our model discards large weight updates, which may limit the model's learning capacity. We expect future works to better utilize such large updates and further improve the performance.

AI USE STATEMENT

In this work, we used generative AI tools for no tasks with required disclosure. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, and generating synthetic data set is not applicable to this work. Additionally, we used generative AI tools for editing a research paper to improve readability, identifying relevant literature. We have reviewed all AI-assisted work. We checked all AI improved writing and related works. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision*, pp. 144–161. Springer, 2018. doi: 10.1007/978-3-030-01219-9_9.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11849–11860, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019b.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning (ICML)*, pp. 322–332. PMLR, 2019.

Xueying Bai, Jinghuan Shang, Yifan Sun, and Niranjan Balasubramanian. Continual learning with global alignment. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences (PNAS)*, 117(48):30063–30074, 2020.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we must understand kernel learning. In *International Conference on Machine Learning (ICML)*, pp. 541–549. PMLR, 2018.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In Po-Ling Loh and Maxim Raginsky (eds.), *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pp. 4028–4079. PMLR, 02–05 Jul 2022. URL https://proceedings.mlr.press/v178/evron22a.html.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3 (4):128–135, 1999. doi: 10.1016/S1364-6613(99)01294-2.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022. arXiv preprint arXiv:1903.08560 (2019).

Ningge Hu, Xiao Li, Zihao Zhao, Bo Xu, Jiaqing Liu, and Xuezheng Fan. Opcr: Continual generalized category discovery via orthogonal prototypes and confidence-aware label refinement. *Neurocomputing*, pp. 133011, 2026.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Peizhong Ju, Xiaojun Lin, and Jia Liu. Overfitting can be harmless for ℓ_1-regularized linear regression. *arXiv preprint arXiv:2003.01180*, 2020.

Peizhong Ju, Xiaojun Lin, and Jia Liu. On the generalization error of three-layer neural tangent kernel models. *arXiv preprint arXiv:2202.01234*, 2022.

Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International conference on machine learning*, pp. 10734–10750. PMLR, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. Parameter-level soft-masking for continual learning. In *International conference on machine learning*, pp. 17492–17505. PMLR, 2023.

Hongbo Li, Sen Lin, Lingjie Duan, Yingbin Liang, and Ness Shroff. Theory on mixture-of-experts in continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7XgKAabsPp.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016.

Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In *International Conference on Machine Learning*, pp. 21078–21100. PMLR, 2023a.

Sen Lin, Peizhong Ju, Yingbin Liang, and Ness Shroff. Theory on forgetting and generalization of continual learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21078–21100. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/lin23f.html.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *European Conference on Computer Vision*, pp. 379–395. Springer, 2024.

Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214): 1–50, 2023.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75 (4):667–766, 2022. arXiv preprint arXiv:1908.05355 (2019).

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020.

Partha P Mitra. Understanding overfitting in ℓ_1-minimum norm interpolation. *arXiv preprint arXiv:1911.01910*, 2019.

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. doi: 10.1016/j.neunet.2019.01.012.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pp. 524–540. Springer, 2020. doi: 10.1007/978-3-030-58536-5_31.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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*, 2017.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4548–4557. PMLR, 2018.

Longxiang Tang, Zhuotao Tian, Kai Li, Chunming He, Hantao Zhou, Hengshuang Zhao, Xiu Li, and Jiaya Jia. Mind the interference: Retaining pre-trained knowledge in parameter efficient continual learning of vision-language models. In *European conference on computer vision*, pp. 346–365. Springer, 2024.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jiménez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *International Conference on Machine Learning (ICML)*, 2024. URL `https://proceedings.mlr.press/v235/wang24k.html`.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35:5682–5695, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 139–149. IEEE, 2022b.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in neural information processing systems*, 33:15173–15184, 2020.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022a.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022b.

Yicheng Xu, Yuxin Chen, Jiahao Nie, Yusong Wang, Huiping Zhuang, and Manabu Okumura. Advancing cross-domain discriminability in continual learning of vision-language models. *Advances in Neural Information Processing Systems*, 37:51552–51576, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017.

Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. Preventing zero-shot transfer degradation in continual learning of vision-language models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19068–19079. IEEE, 2023.

chao zhou, Tom Jacobs, Advait Gadhikar, and Rebekka Burkholz. Pay attention to small weights. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 146132–146167. Curran Associates, Inc., 2025. doi: 10.52202/085713-4890. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d71a4a6c796cacd9b8a298589943cdf3-Paper-Conference.pdf.

A PROOF OF ANALYSIS 1

For each task i, we have the training data $\mathcal{D}_i = \{\mathbf{X}_i, \mathbf{y}_i\}$ where the input $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ is a matrix with n_i samples of d dimensional feature vectors; $\mathbf{y}_i \in \mathbb{R}^{n_i}$ contains n_i scalar values. We use a linear model $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}, \mathbf{w} \in \mathbb{R}^d$ to solve all tasks, and denote the weight learned in task i as $\mathbf{w}_i$. Assume the model has enough capacity and there is a ground truth weight $\mathbf{w}^* \in \mathbb{R}^d$ that fits all N tasks' data. Therefore, for any task i we have the target $\mathbf{y}_i = \mathbf{X}_i\mathbf{w}^*$. The loss on task i is $\|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2$, $\|\cdot\|_2$ represents the L2-norm. For simple notations, here we do not compute the average loss.

To prove the analysis 1, we first derive the expression of model weights updated after each task, by gradient descent (GD).

A.1 WEIGHT AFTER GRADIENT DESCENT

For an over-parametrized linear model, there can be multiple model weights that give the same outputs (Lin et al., 2023a). Therefore, we first prove the theorem below to show that deviations between those weights have to be orthogonal to the input data.

Theorem 1 *For each regression task i, to get the same output, the deviation $\Delta\mathbf{w}_i$ added to the weight $\mathbf{w}_i$ can only be vectors orthogonal to the input data $\mathbf{X}_i$.*

Proof. Assume there is a non-orthogonal deviation $\Delta\mathbf{w}_i$ that can output the same value when added to $\mathbf{w}_i$. Then $\mathbf{w}_i$ and $\Delta\mathbf{w}_i$ satisfy:

$$\mathbf{X}_i(\mathbf{w}_i + \Delta\mathbf{w}_i) = \mathbf{y}_i.$$

Since $\mathbf{X}_i\mathbf{w}_i = \mathbf{y}_i$, $\Delta\mathbf{w}_i$ satisfies the following after rearrangement:

$$\begin{aligned}\mathbf{X}_i\Delta\mathbf{w}_i &= \mathbf{y}_i - \mathbf{X}_i\mathbf{w}_i \\ &= \mathbf{0}\end{aligned}$$

which means $\Delta\mathbf{w}_i$ is orthogonal to the data $\mathbf{X}_i$. This contradicts the assumption and proves the theorem.

Then we derive the weight updated in gradient descent after each task, in sequential training.

Theorem 2 *For each task i, we decompose the input matrix as $\mathbf{X}_i^T\mathbf{X}_i = \mathbf{Q}_i\mathbf{\Omega}_i{\mathbf{Q}_i}^T$, where α is the learning rate, $\mathbf{Q}_i \in \mathbb{R}^{d\times d}$ is an orthonormal basis and $\mathbf{\Omega}_i = diag(o_i^1, o_i^2, ..., o_i^d)$ is a diagonal matrix. Based on the low-rank assumption, only $o_i^1, ...o_i^{r_i}$ $(r_i < d)$ are non-zero. We learn each task by t steps' gradient descent (GD) on that task's data. The weight $\mathbf{w}_i$ after learning task i is:*

$$\mathbf{w}_i = \mathbf{w}^* - (\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1}), \quad \mathbf{B}_i = \mathbf{Q}_i(\mathbf{I} - \alpha\mathbf{\Omega}_i)\mathbf{Q}_i^T \tag{7}$$

$$\mathbf{w}_{i-1} = \mathbf{w}^* - \prod\nolimits_{j<i}(\mathbf{B}_j)^t(\mathbf{w}^* - \mathbf{w}_0), \quad \mathbf{B}_j = \mathbf{Q}_j(\mathbf{I} - \alpha\mathbf{\Omega}_j)\mathbf{Q}_j^T \tag{8}$$

where $\mathbf{w}_0$ is the initialized weight, α is the learning rate. $\mathbf{w}_{i-1}$ is the start weight for task i, which is the weight learned after t steps GD of task $(i-1)$.

Proof. We prove the theorem by induction. We define any solution $\mathbf{w}_i^*$ for task i as $\mathbf{w}_i^* = \mathbf{w}^* + \Delta\mathbf{w}_i$ since there can be multiple weights that output the same value. For each task i starting with the weight $\mathbf{w}_{i-1}$, the weight after the first GD step is:

$$\begin{aligned}\mathbf{w}_i^1 &= \mathbf{w}_{i-1} - \alpha\mathbf{X}_i^T\mathbf{X}_i(\mathbf{w}_{i-1} - (\mathbf{w}^* + \Delta\mathbf{w}_i)) \\ &= \mathbf{w}_{i-1} - \alpha\mathbf{X}_i^T\mathbf{X}_i(\mathbf{w}_{i-1} - \mathbf{w}^*) && \text{(By Theorem 1)} \\ &= (\mathbf{Q}_i\mathbf{I}\mathbf{Q}_i^T - \alpha\mathbf{Q}_i\mathbf{\Omega}_i\mathbf{Q}_i^T)\mathbf{w}_{i-1} - (\mathbf{Q}_i\mathbf{I}\mathbf{Q}_i^T - \alpha\mathbf{Q}_i\mathbf{\Omega}_i\mathbf{Q}_i^T)\mathbf{w}^* + \mathbf{I}\mathbf{w}^* && (\mathbf{Q}_i \text{ is an orthonormal matrix}) \\ &= \mathbf{w}^* - \mathbf{Q}_i(\mathbf{I} - \alpha\mathbf{\Omega}_i)\mathbf{Q}_i^T(\mathbf{w}^* - \mathbf{w}_{i-1}) \\ &= \mathbf{w}^* - \mathbf{B}_i(\mathbf{w}^* - \mathbf{w}_{i-1})\end{aligned}$$

Assume at the t-th step, the weight $\mathbf{w}_i^t$ satisfies the theorem: $\mathbf{w}_i^t = \mathbf{w}^* - (\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1})$, then at the $(t+1)$-th step, the weight is:

$$\begin{aligned}
\mathbf{w}_i^{t+1} &= \mathbf{w}_i^t - \alpha \mathbf{X}_i^T \mathbf{X}_i (\mathbf{w}_i^t - \mathbf{w}^*) \\
&= \mathbf{w}^* - \mathbf{B}_i(\mathbf{w}^* - \mathbf{w}_i^t) \\
&= \mathbf{w}^* - \mathbf{B}_i(\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1}) \qquad (\mathbf{w}_i^t = \mathbf{w}^* - (\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1})) \\
&= \mathbf{w}^* - (\mathbf{B}_i)^{t+1}(\mathbf{w}^* - \mathbf{w}_{i-1})
\end{aligned}$$

This satisfies the expression $\mathbf{w}_i^t = \mathbf{w}^* - (\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1})$ for any step t. Eq. (7) is proved.

Since the start weight $\mathbf{w}_{i-1}$ of the task i is the weight learned at task $(i-1)$, we further expand the start weight as:

$$\begin{aligned}
\mathbf{w}_{i-1} &= \mathbf{w}_{i-1}^t \\
&= \mathbf{w}^* - (\mathbf{B}_{i-1})^t(\mathbf{w}^* - \mathbf{w}_{i-2}) \qquad \text{(By Eq. (7))} \\
&= \mathbf{w}^* - (\mathbf{B}_{i-1})^t(\mathbf{B}_{i-2})^t(\mathbf{w}^* - \mathbf{w}_{i-3}) \qquad (\mathbf{w}^* - \mathbf{w}_{i-2} = (\mathbf{B}_{i-2})^t(\mathbf{w}^* - \mathbf{w}_{i-3})) \\
&\ldots \\
&= \mathbf{w}^* - \prod\nolimits_{j<i} (\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_0)
\end{aligned}$$

Eq. (8) is proved. Then Theorem 2 is proved.

A.2 Cross-Task Generalization Error and Forgetting

The bounds for CL metrics in Analysis 1 is derived below.

Theorem 3 *Given the initial weight $\mathbf{w}_{i-1}$, we sequentially learn tasks i and $(i+1)$ with inputs $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$. Each input $\mathbf{X}_i$ satisfies $\mathbf{X}_i^T\mathbf{X}_i = \mathbf{Q}_i\mathbf{\Omega}_i{\mathbf{Q}_i}^T$, where $\mathbf{Q}_i \in \mathbb{R}^{d\times d}$ is an orthonormal basis and $\mathbf{\Omega}_i = diag(o_i^1, o_i^2, ..., o_i^d)$ is a diagonal matrix. The initial weight $\mathbf{w}_{i-1}$'s generalization error on tasks i and $(i+1)$ is bounded by:*

$$gen(\mathbf{w}_{i-1}; \mathbf{X}_i, \mathbf{X}_{i+1}) \leq \frac{1}{2}\big(\max(\mathbf{\Omega}_i) + \max(\mathbf{\Omega}_{i+1})\big)\|\mathbf{w}^* - \mathbf{w}_{i-1}\|_2^2, \tag{9}$$

The forgetting of task i after learning task $(i+1)$ is bounded by:

$$fgt(\mathbf{w}_{i-1}; \mathbf{X}_i) \leq \|\mathbf{w}^* - \mathbf{w}_{i-1}\|_2^2 \underbrace{\big\|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{P}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\big\|_2^2}_{\neq \mathbf{0}}, \tag{10}$$

where α is the learning rate, $\mathbf{P}_{i+1} = \mathbf{Q}_i^T\mathbf{Q}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_{i+1})^t\mathbf{Q}_{i+1}^T\mathbf{Q}_i$ is the transformation between two basis $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$, t is the GD steps of learning both tasks, $\|\cdot\|_2$ represents the L2-norm.

Proof. We show the derivation of each metric below.

In-Task Loss For the weight $\mathbf{w}_i$ after learning task i, its in-task loss $\mathcal{L}(\mathbf{w}_i; \mathbf{X}_i, \mathbf{y}_i)$ is:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_i; \mathbf{X}_i, \mathbf{y}_i) &= ||\mathbf{X}_i\mathbf{w}_i - \mathbf{y}_i||_2^2 \\
&= ||\mathbf{X}_i(\mathbf{w}^* - \mathbf{w}_i)||_2^2 \\
&= (\mathbf{w}^* - \mathbf{w}_i)^T\mathbf{X}_i^T\mathbf{X}_i(\mathbf{w}^* - \mathbf{w}_i) \\
&= (\mathbf{w}^* - \mathbf{w}_{i-1})^T(\mathbf{B}_i)^t\mathbf{Q}_i\mathbf{\Omega}_i\mathbf{Q}_i^T(\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1}) \\
&= (\mathbf{w}^* - \mathbf{w}_{i-1})^T\mathbf{Q}_i(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\mathbf{\Omega}_i(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\mathbf{Q}_i^T(\mathbf{w}^* - \mathbf{w}_{i-1}) \\
&= \|\underbrace{\mathbf{\Omega}_i^{\frac{1}{2}}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t}_{\text{converges to 0}}\mathbf{Q}_i^T(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2
\end{aligned}$$

We assume the in-task loss is 0 and GD converges, which requires $0 < \alpha o_i < 2$ for each o_i in $\mathbf{\Omega}_i$. Notice that $\mathbf{w}_i$ still deviates to the global optimal $\mathbf{w}^*$ at the k-th dimension where $o_i^k = 0$. However, the loss is calculated in the space of $\mathbf{X}_i^T\mathbf{X}_i$ which does not span the k-th dimension. So the loss can still be zero at the convergence time.

Generalization Error (Eq. (9)) In CL, when learning a new task, its initial weight is the weight learned after the previous task. So given the initial weight $\mathbf{w}_{i-1}$, we compute its cross-task generalization error on task i and $i+1$ by computing its average loss on data of task i and $i+1$, shown below:

$$\begin{aligned} gen(\mathbf{w}_{i-1}; \mathbf{X}_i, \mathbf{X}_{i+1}) &= \frac{1}{2}\big(\|\mathbf{X}_i(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2 + \|\mathbf{X}_{i+1}(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2\big) \\ &\leq \frac{1}{2}\big(\|\mathbf{X}_i\|_2^2\|\mathbf{w}^* - \mathbf{w}_{i-1}\|_2^2 + \|\mathbf{X}_{i+1}\|_2^2\|\mathbf{w}^* - \mathbf{w}_{i-1}\|_2^2\big) \\ &= \frac{1}{2}\big(\max(\mathbf{\Omega}_i) + \max(\mathbf{\Omega}_{i+1})\big)\|\mathbf{w}^* - \mathbf{w}_{i-1}\|_2^2 \end{aligned}$$

Forgetting (Eq. (10)) After learning task $i+1$, we show the forgetting of the weight $\mathbf{w}_{i+1}$ on the previous task i's data (assume the in-task loss of task i is 0):

$$\begin{aligned} fgt(\mathbf{w}_{i+1}; \mathbf{X}_i, \mathbf{y}_i) &= \mathcal{L}(\mathbf{w}_{i+1}; \mathbf{X}_i, \mathbf{y}_i) - \underbrace{\mathcal{L}(\mathbf{w}_i; \mathbf{X}_i, \mathbf{y}_i)}_{=0} \\ &= \|\mathbf{X}_i(\mathbf{w}^* - \mathbf{w}_{i+1})\|_2^2 \\ &= (\mathbf{w}^* - \mathbf{w}_{i+1})^T\mathbf{X}_i^T\mathbf{X}_i(\mathbf{w}^* - \mathbf{w}_{i+1}) \\ &= (\mathbf{w}^* - \mathbf{w}_i)^T(\mathbf{B}_{i+1})^t\mathbf{Q}_i\mathbf{\Omega}_i\mathbf{Q}_i^T(\mathbf{B}_{i+1})^t(\mathbf{w}^* - \mathbf{w}_{i-1}) \\ &= (\mathbf{w}^* - \mathbf{w}_{i-1})^T(\mathbf{B}_i)^t(\mathbf{B}_{i+1})^t\mathbf{Q}_i\mathbf{\Omega}_i\mathbf{Q}_i^T(\mathbf{B}_{i+1})^t(\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1}) \\ &= \|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{Q}_i^T(\mathbf{B}_{i+1})^t(\mathbf{B}_i)^t(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2 \\ &= \|\mathbf{\Omega}_i^{\frac{1}{2}}\underbrace{\mathbf{Q}_i^T\mathbf{Q}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_{i+1})^t\mathbf{Q}_{i+1}^T\mathbf{Q}_i}_{\mathbf{P}_{i+1}}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\mathbf{Q}_i^T(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2 \\ &\leq \|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{P}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\|_2^2\|\mathbf{Q}_i^T(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2 \\ &= \|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{P}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\|_2^2\|(\mathbf{w}^* - \mathbf{w}_{i-1})\|_2^2 \end{aligned}$$

The forgetting $\mathcal{L}(\mathbf{w}_{i+1}; \mathbf{X}_i, \mathbf{y}_i) = 0$ when $\mathbf{Q}_i = \mathbf{Q}_{i+1}$ (features have the same basis) or $\mathbf{Q}_i \perp \mathbf{Q}_{i+1}$ (features are orthogonal) which make $\mathbf{Q}_i^T\mathbf{Q}_{i+1} = \mathbf{I}$. Also, when the data $\mathbf{X}_i$ is full-ranked and the training converges, $(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t \to \mathbf{0}$ and the term $\|\mathbf{\Omega}_i^{\frac{1}{2}}\mathbf{P}_{i+1}(\mathbf{I} - \alpha\mathbf{\Omega}_i)^t\|_2^2$ is also 0 (See the In-Task Loss). However, the task data are usually low-ranked so the feature alignment is necessary. In other cases, the deviation to the global optimal in task i may entangle with the space of task $(i+1)$ and the forgetting is not zero .

B Small Magnitude Weights After Pre-Training

We derive the linear model's weights after pre-training. Based on Theorem 2, given a random initialized weight $\mathbf{w}_0$, the model weight $\mathbf{w}_{pre}$ after pre-training is:

$$\begin{aligned} \mathbf{w}_{pre} &= \mathbf{w}^* - (\mathbf{B}_{pre})^t(\mathbf{w}^* - \mathbf{w}_0), \;\; \mathbf{B}_{pre} = \mathbf{Q}_{pre}(\mathbf{I} - \alpha\mathbf{\Omega}_{pre})\mathbf{Q}_{pre}^T \\ &= \mathbf{Q}_{pre}\mathbf{I}\mathbf{Q}_{pre}^T\mathbf{w}^* - \mathbf{Q}_{pre}(\mathbf{I} - \alpha\mathbf{\Omega}_{pre})^t\mathbf{Q}_{pre}^T(\mathbf{w}^* - \mathbf{w}_0) \\ &= \mathbf{Q}_{pre}\big(\mathbf{I} - (\mathbf{I} - \alpha\mathbf{\Omega}_{pre})^t\big)\mathbf{Q}_{pre}^T\mathbf{w}^* + \mathbf{Q}_{pre}(\mathbf{I} - \alpha\mathbf{\Omega}_{pre})^t\mathbf{Q}_{pre}^T\mathbf{w}_0 \end{aligned}$$

wehre $\mathbf{Q}_{pre}$ are the basis of the pre-trained data and $\mathbf{\Omega}_{pre}$ is the corresponding diagonal matrix.

Assume the model weights will converge on the pre-trained data and achieve the zero loss, we have $(1 - \alpha\mathbf{\Omega}_{pre}[n])^t \to \mathbf{0}$ on the n-th basis that the pre-trained data spans. Therefore, $\mathbf{w}_{pre}$ can be disentangled into two parts on different basis in $\mathbf{Q}_{pre}$:

$$\big(\mathbf{Q}_{pre}[n]\big)^T\mathbf{w}_{pre} = \begin{cases} \big(\mathbf{Q}_{pre}[n]\big)^T\mathbf{w}^*, & (1 - \alpha\mathbf{\Omega}_{pre}[n])^t \to \mathbf{0} \\ \big(\mathbf{Q}_{pre}[n]\big)^T\mathbf{w}_0, & (1 - \alpha\mathbf{\Omega}_{pre}[n])^t \to \mathbf{1} \end{cases}$$

Since the initialization $\mathbf{w}_0$ is usually small and $\mathbf{Q}_{pre}$ is orthonormal, we have the corresponding weight value to be small when projected on the basis $\mathbf{Q}_{pre}[n]$. Based on this, the i-th element in

$\mathbf{w}_{pre}$ can be written as:

$$\mathbf{w}_{pre}[i] = \sum\nolimits_n \left(\mathbf{Q}_{pre}[n]\right)^T \mathbf{w}_{pre} \cdot \mathbf{Q}_{pre}[n, i],$$

where $\mathbf{Q}_{pre}[n, i] \in \mathbb{R}$ is the i-th element in the n-th basis $\mathbf{Q}_{pre}[n]$. Since the projected value for $\mathbf{w}_0$ is small (close to 0), $\mathbf{w}_{pre}[i]$ is mainly decided by the space that pre-trained data spans, and thus we have

$$\mathbf{w}_{pre}[i] \approx \sum\nolimits_{(1-\alpha\mathbf{\Omega}_{pre}[n])^t \to \mathbf{0}} \left(\mathbf{Q}_{pre}[n]\right)^T \mathbf{w}^* \cdot \mathbf{Q}_{pre}[n, i],$$

Since $\mathbf{Q}_{pre}$ is an orthonormal basis, the magnitude of each element satisfies $\|\mathbf{Q}_{pre}[n, i]\| \leq 1$. Therefore, the magnitude of $\mathbf{w}_{pre}[i]$ satisfies:

$$\|\mathbf{w}_{pre}[i]\| \leq \sum\nolimits_{(1-\alpha\mathbf{\Omega}_{pre}[n])^t \to \mathbf{0}} \|\left(\mathbf{Q}_{pre}[n]\right)^T \mathbf{w}^*\| + \epsilon,$$

where ϵ is for the dimensions that are not well-spanned.

For a close to zero weight after pre-training, it either comes from the space that is not spanned by the pre-trained data or is considered as unimportant in pre-training (i.e., $\|(\mathbf{Q}_{pre}[n])^T \mathbf{w}^*\|$ is amll). Considering the over-parametrized nature of large models, the first case may usually happen. For the first case, those parameters may not be well-trained and can not provide a generalizable reference in fine-tuning; for the second case, changing the weights close to optimal in pre-training may degrade model's original generalization capacity. In either case, it is undesirable to rely on abrupt changes on small pre-trained weights in fine-tuning.