

# Nostalgia: Preventing forgetting in Supervised Fine-Tuning for Vision

Anonymous CVPR submission

Paper ID \*\*\*\*\*

## Abstract

Vision, Language, and multimodal models have demonstrated remarkable generalization through extensive pre-training on large, diverse datasets. However, in practice, these models require domain-specific supervised fine-tuning to maximize utility in specialized applications. This process, while essential, often induces catastrophic forgetting—models lose knowledge acquired during previous training tasks and perform poorly on these learned tasks while fine-tuning on the current data. Traditional solutions either rely on replaying prior data, introduce heavy computational overhead, or lack comprehensive theoretical justification and are heuristic driven, making them unsuitable for many real-world deployment settings. We introduce **Nostalgia**, a novel optimization-based method for fine-tuning vision models in domain-specific settings, designed to robustly mitigate catastrophic forgetting in multiple downstream tasks. Unlike conventional approaches, Nostalgia does not require access to any prior task data or labelled replay buffers. Instead, it incorporates principled constraints inspired by recent theoretical advances in continual learning to maintain essential properties of the pre-trained model during adaptation. Our approach enables efficient and scalable fine-tuning while offering strong theoretical retention guarantees. We also provide empirical results across several popular vision benchmarks, demonstrating that Nostalgia consistently improves retention of prior task performance without compromising adaptation to new tasks.

## 1. Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) such as GPT-4o[1], Gemini[3], and LLaMA-4[4] have redefined AI by achieving human-level or superior performance on complex multimodal benchmarks. Their large-scale pre-training across diverse languages, modalities, and domains enables advanced reasoning, dialogue, and adaptability, making them powerful backbones for scientific, industrial, and other applications.

Fine-tuning such pre-trained models has become the

dominant paradigm for task adaptation, offering efficiency and strong performance across vision, language, and reinforcement learning. However, repeated adaptation introduces well-known challenges such as catastrophic forgetting, where there is a significant loss in performance on the previously learnt tasks, specifically related to pretraining tasks and loss of the generalization gained during pre-training—motivating the search for learning strategies that can integrate new information without erasing prior knowledge. This phenomenon is often referred to as *over adaptation*.

While many methods in the past have been developed over the years to solve the problem of over-adaptation in downstream SFT, most of them don't scale well or don't have any theoretical backing. We propose a method inspired by local algorithms as defined by Lanzillotta et al 2024[2], from a continual learning perspective, while approximating second-order methods for effective computations. Local algorithms, although they have a stronger theoretical backing, they lack computational support due to its need for second-order gradient computations, and assumptions related to second-order polynomial approximations and constraint optimisation-based methods. We propose a method in which we can combine this with modern implementations of gradient computation, making this more efficient and scalable to larger LLMs and VLMs.

## 2. Related Works

**Regularization and Parameter-Constrained Approaches:** A classical line of work introduces explicit regularization terms to keep fine-tuned parameters close to their pre-trained initialization. Methods such as L2-SP and Elastic Weight Consolidation (EWC) constrain updates using L2 or Fisher-weighted penalties, discouraging deviation from important pre-trained weights. Other extensions—such as Selective Projection Decay or Laplace-based priors—attempt to model parameter importance more precisely. While these methods are conceptually simple, their performance is highly sensitive to regularization strength, and they often underperform on large-scale multimodal models, where the correspondence

between parameter distance and functional similarity is weak. The term below represents the loss function  $\mathcal{L}_{l2sp}(\theta)$  for a parameter  $\theta$  as the loss function of the task,  $\mathcal{L}_{task}(\theta)$ , and a regulariser representing the  $l2$ -norm based distance from  $\theta_0$  the parameter weights after pre-training:

$$\mathcal{L}_{l2sp}(\theta) = \mathcal{L}_{task}(\theta) + \frac{\lambda}{2} \|\theta - \theta_0\|_2^2$$

A more selective formulation is *Elastic Weight Consolidation (EWC)*, which estimates the importance of each parameter based on the Fisher Information Matrix  $F$  computed during pretraining. The fine-tuning loss is augmented as:

$$\mathcal{L}_{EWC}(\theta) = \mathcal{L}_{task}(\theta) + \frac{\lambda}{2} \sum_i F_i (\theta_i - \theta_{0,i})^2,$$

**Adapters:** Adapters (Houlsby et al., 2019) and their variants (e.g., AdaptFormer, AdapterFusion) introduce small, trainable bottleneck modules within otherwise frozen transformer models, allowing the model to adapt to new tasks without modifying the original backbone parameters. The key idea is to insert these lightweight modules—typically consisting of a down-projection, non-linearity, and up-projection—between existing layers of a pre-trained model. This design enables parameter-efficient fine-tuning, as only the adapters are updated while the bulk of the model remains frozen. Variants such as AdaptFormer improve the integration of these modules by strategically placing them in attention and feed-forward sublayers, while AdapterFusion allows multiple pre-trained adapters to be combined, facilitating multi-task learning and transfer learning by leveraging knowledge from previously learned tasks. Overall, adapters provide a scalable and modular approach to extending large pre-trained transformers to diverse downstream tasks without incurring the computational and memory costs of full fine-tuning.

**Ensembling:** Weight interpolation (“model soups”) averages parameters across fine-tuned and pre-trained models, often improving robustness without retraining. Output ensembling aggregates predictions from different checkpoints to strike a balance between specialization and generality. Although effective in mid-scale settings (e.g., ViT, CLIP, LLaMA-7B), these techniques scale poorly to frontier models due to memory and alignment constraints. These methods have shown resilience even in other modalities such as Language.

**Rehearsal Methods:** Rehearsal-based strategies mitigate catastrophic forgetting by replaying samples from previous tasks during new task training. In language model SFT, this involves mixing pretraining data with downstream batches

to preserve general knowledge and prevent domain overfitting. Approaches vary in buffer maintenance, sampling balance, and selective replay, offering a scalable, practical alternative to costly second-order or parameter-constrained methods for continual learning and SFT retention.

**Low Rank Adaptation:** LoRA fine-tuning is a parameter-efficient technique that introduces low-rank adaptation matrices into existing weight tensors of a pretrained model while keeping the original parameters frozen. Instead of updating the full set of model weights, LoRA learns compact task-specific updates that are injected through a rank-decomposed representation. This approach significantly reduces the number of trainable parameters and memory usage, while maintaining competitive performance compared to full fine-tuning. By separating the pretrained backbone from the learned adaptations, LoRA also enables efficient task switching, sharing, and composability across multiple downstream objectives.

### 3. Problem setup

#### 3.1. Supervised Fine Tuning as Continual Learning

Our problem considers the setting of continual learning as the sequential acquisition of multiple supervised tasks  $\tau_0, \tau_1, \tau_2, \dots, \tau_T$ , where the data for task  $\tau_t$  is denoted  $D_t = \{(x, y) \mid x \in \mathcal{X}_t, y \in \mathcal{Y}_t\}$ .  $\tau_0$  is considered as the pre-training task. We define forgetting for task  $\tau_t$  as:

$$E_t(\theta) = L_t(\theta) - L_t(\theta_t) \quad (1)$$

where  $L_t(\theta)$  is the loss function for  $\tau_t$  evaluated for model parameters  $\theta$ , and  $\theta_t$  are the parameters at the completion of learning task  $\tau_t$ .

The goal of continual learning is to incrementally update the model on each new task  $\tau_t$  using only  $D_t$  and possibly an external memory  $M_t$  (e.g., a buffer), while retaining performance on previously learned tasks. Performance is measured by minimizing both forgetting  $E_t(\theta)$  and the following multi-task loss:

$$L_t^{MT}(\theta) = \frac{1}{t} \sum_{i=1}^t L_i(\theta) \quad (2)$$

The continual learning objective is to find update rules so that, after each step, the updated parameters approximately minimize the multi-task loss using only current task data,  $D_t$  and memory,  $M_t$ :

$$\min_{\theta_t} obj_t(D_t, M_t) \approx L_t^{MT}(\theta_t) \quad (3)$$

The central challenge is that direct access to past task data  $D_o$ , for  $o < t$ , is unavailable. Thus, the algorithm must approximate the true multi-task loss with whatever information is stored or accessible.

Previous work addresses this by using polynomial approximations for the loss function near each task optimum. Empirically and theoretically, loss surfaces tend to be well-behaved and locally convex near minima, so quadratic approximations are widely adopted:

$$\hat{L}_t(\theta) \approx L_t(\theta_t) + (\theta - \theta_t)^T \nabla L_t(\theta_t) \quad (4)$$

$$+ \frac{1}{2}(\theta - \theta_t)^T H_t(\theta_t)(\theta - \theta_t) \quad (5)$$

where  $H_t(\theta_t)$  is the Hessian of  $L_t$  evaluated at  $\theta_t$ . This lets us redefine the forgetting term as

$$E_t(\theta) = (\theta - \theta_t)^T \nabla L_t(\theta_t) + \frac{1}{2}(\theta - \theta_t)^T H_t(\theta_t)(\theta - \theta_t) \quad (6)$$

Lanzillotta et al. [2] primarily classify continual learning algorithms into two categories — local and global — based on their approach to multi-task loss approximation:

**Definition 3.1** (Local and global task loss approximations.). Let  $I(X; Y)$  denote the mutual information of the pair of random variables  $(X, Y)$ . We say that the task loss’s second order polynomial approximation  $\hat{L}_t(\theta)$  is *local* when  $I(\hat{L}_t(\theta); \theta_t) > 0 \forall \theta \in \Theta$ , and that it is *global* when  $I(\hat{L}(\theta); \theta_t) = 0 \forall \theta \in \Theta - \{\theta_t\}$ .

Local algorithms, however, do provide a strong theoretical justification for searching the space from which we can get zero forgetting, as stated in the following lemma, from [2]:

**Lemma:** (Optimal quadratic local continual learning) For any continual learning algorithm producing a sequence of parameters  $\theta_1, \theta_2, \dots, \theta_t$  such that  $\theta_i$  is the local minima of  $L_i$  and  $\sup_{\theta_i, \theta_k} \|\theta_i - \theta_k\|^3 < \epsilon$ , the following relationship holds:

$$E(1), \dots, E(t-1) = 0 \implies$$

$$E(t) = \frac{1}{2} \Delta_t^T \left( \frac{1}{t} \sum_{j=1}^{t-1} H_j^* \right) \Delta_t \geq 0$$

Moreover, if  $E(1), \dots, E(t-1) = 0$  the optimal learning objective for task  $t$  is:

$$\min_{\Delta_t \in \Theta} L_t(\theta_{t-1} + \Delta_t) \quad s.t. \quad \Delta_t^T \left( \frac{1}{t} \sum_{j=1}^{t-1} H_j^* \right) \Delta_t = 0$$

Note that the constraints described through the lemma force the update steps to be lie in the *null-space* of the average Hessian matrix. Intuitively when a quadratic approximation to the loss is accurate enough and each task solution is a local minimum, the parameter updates must be taken along directions where the multi-task loss landscape is (absolutely) flat to prevent forgetting.

### 3.2. Efficient Computation: Nostalgia

The primary challenge with enforcing optimal updates as defined by Lemma is that computing and storing the second-order Hessians is not feasible for over-parametrized models such as LLMs, and that this is a constrained optimisation problem. In our work, we exploit the fact that the spectral analysis of the Hessian is a low-rank decomposition based on only a few eigen-vectors, to efficiently compute the model. Our primary idea is that each Hessian  $H_j$  can be approximated as:

$$H_j \approx Q_j \Lambda_j Q_j^T$$

where  $Q_j \in \mathbb{R}^{d \times k}$  representing the top-k eigen-vectors and  $\Lambda_j \in \mathbb{R}^{k \times k}$  representing diagonal matrix with the top-k eigen-values. Each of these steps is essentially being low-rank update, and not a dense matrix. We employ **Lanczos iteration** with a small number of steps to get an orthonormal basis  $Q_j$  spanning the top curvature directions.

To combat the constraint matrix part of this, we employ the following orthogonal projection:

$$\Delta_t = (I - QQ^T)g_t$$

where  $g_t$  represents an update step from the nor. This gives an easy way in which we can compute the gradients  $g_t$  usually, and project them from the directions of previous curvatures to enforce the constraints, while optimising. We combine this method of updates with LoRA-based fine-tuning methods to get a more effective computation of  $\Delta_t$  and combining this with better modern-day computation ability.

We also propose a more computationally friendly version of nostalgia: layer-wise nostalgia. Where we only consider the hessian between the parameters belonging to the same layer (or set of parameters). The algorithm is described in Algorithm 3.

**Lanczos Iterations:** Lanczos method provides an efficient iterative procedure for estimating the leading eigenvalues and eigenvectors of large symmetric matrices, such as the Hessian of a neural network loss function. By constructing an orthonormal basis of the Krylov subspace  $\mathcal{K}_k(H, v) = \text{span}\{v, Hv, H^2v, \dots, H^{k-1}v\}$  for a symmetric matrix  $H$  and an initial vector  $v$ , the method projects  $H$  onto this low-dimensional subspace, yielding a tridiagonal matrix  $T_k$  whose eigenvalues approximate those of  $H$ . This approach avoids explicit storage or inversion of  $H$ , making it well-suited for large-scale models where the Hessian is prohibitively expensive to compute. In practice, the Hessian of smooth scalar-valued loss functions is symmetric by construction, and near local minima it is sufficiently well-conditioned for the Lanczos approximation to yield stable

spectral estimates. Consequently, the Lanczos method has become a standard tool for approximating curvature information in modern deep learning and optimization research.

We also propose a second algorithm for more effective computations, where we only rely on the idea of limited Hessians, where we limit the Hessians to local matrices only. This is much faster for computations.

---

**Algorithm 1** Lanczos Method for Computing the Projection Operator

---

**Require:** Symmetric matrix  $\bar{H} \in \mathbb{R}^{n \times n}$  (average Hessian), initial vector  $v_1$  with  $\|v_1\|_2 = 1$ , number of iterations  $k$

**Ensure:** Projection operator  $P = QQ^\top$  spanning the low-curvature subspace of  $\bar{H}$

```

1: Initialize  $\beta_0 = 0, v_0 = 0$ 
2: for  $j = 1$  to  $k$  do
3:    $w_j \leftarrow \bar{H}v_j - \beta_{j-1}v_{j-1}$ 
4:    $\alpha_j \leftarrow v_j^\top w_j$ 
5:    $w_j \leftarrow w_j - \alpha_j v_j$ 
6:   if  $j < k$  then
7:      $\beta_j \leftarrow \|w_j\|_2$ 
8:     if  $\beta_j = 0$  then
9:       break
10:    end if
11:     $v_{j+1} \leftarrow w_j / \beta_j$ 
12:  end if
13: end for
14: Form tridiagonal matrix

```

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & & & \\ & & \ddots & & \\ & & & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

```

15: Compute eigen-decomposition  $T_k = U\Lambda U^\top$ 
16: Construct  $V_k = [v_1, v_2, \dots, v_k]$ 
17: Obtain approximate eigenvectors of  $\bar{H}$  as  $Q = V_k U$ 
18: Form projection operator  $P = QQ^\top$ 
19: return  $P$ 

```

---

### 3.3. Complexity and Accuracy of Nostalgia

The computational efficiency of the Lanczos algorithm arises from its reliance on matrix-vector products rather than explicit matrix operations. For a symmetric matrix  $\bar{H} \in \mathbb{R}^{n \times n}$  and  $k$  Lanczos iterations, the dominant cost comes from computing the  $k$  matrix-vector products  $\bar{H}v_j$ , resulting in a time complexity of  $\mathcal{O}(kn)$  per task. The additional operations for orthogonalization, scalar updates, and tridiagonal matrix construction scale as  $\mathcal{O}(k^2)$ , which is negligible when  $k \ll n$ .

The storage cost is similarly efficient, requiring  $\mathcal{O}(kn)$

memory to maintain the basis vectors  $V_k = [v_1, v_2, \dots, v_k]$  and  $\mathcal{O}(k^2)$  for the tridiagonal matrix  $T_k$ . In practice,  $k$  is typically small (e.g.,  $k \leq 50$ ), making the algorithm suitable for large-scale models where direct computation or storage of the Hessian is infeasible. When used to construct projection operators in continual learning, the Lanczos procedure needs to be executed only once per task, making it an effective and scalable approximation to curvature-based constraints.

---

**Algorithm 2** Continual Learning with Lanczos Null-Space Projection and LoRA

---

**Require:** pretrained weights  $\theta_0$ , task stream  $\{\mathcal{T}_1, \dots, \mathcal{T}_N\}$ , LoRA rank  $r$ , Lanczos steps  $k$ , EMA rate  $\eta \in (0, 1]$

**Ensure:** final weights  $\theta_N$

```

1: Initialize  $\bar{H} \leftarrow H_0$   $\triangleright$  stored Hessian estimate (single matrix)
2: for  $t = 1$  to  $N$  do
3:    $\Pi_{\text{null}} \leftarrow \text{LANCZOSPROJECTION}(\bar{H}, k)$ 
4:   Initialize LoRA adapters  $p \leftarrow 0 \in \mathbb{R}^{d_p}$ 
5:   for each minibatch  $b \in \mathcal{T}_t$  do
6:     compute loss  $\ell_b(\theta_{t-1}, p)$ 
7:      $g_p \leftarrow \nabla_p \ell_b(\theta_{t-1}, p)$ 
8:      $g_p^\perp \leftarrow \Pi_{\text{null}}(g_p)$ 
9:     update adapters  $p \leftarrow \text{OPTIMSTEP}(p, g_p^\perp)$ 
10:  end for
11:   $\Delta\theta_t \leftarrow \text{INJECT}(p)$ 
12:   $\theta_t \leftarrow \theta_{t-1} + \Delta\theta_t$ 
13:  reset adapters  $p \leftarrow 0$ 
14:  estimate task Hessian  $H_t \approx \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_t}(\theta_t)$ 
15:   $\bar{H} \leftarrow (1 - \eta)\bar{H} + \eta H_t$ 
16: end for
17: return  $\theta_N$ 

```

---

**Algorithm 3** Layer-wise Continual Learning with Lanczos Projection and LoRA

---

**Require:** pretrained  $\theta_0$ , tasks  $\{\mathcal{T}_1, \dots, \mathcal{T}_N\}$ , layers  $\mathcal{L}$ , LoRA rank  $r$ , Lanczos steps  $\{k_\ell\}$ , EMA rate  $\eta$

**Ensure:** final  $\theta_N$

- 1: Initialize  $\{\bar{H}_\ell \leftarrow H_{0,\ell}\}_{\ell \in \mathcal{L}}$
- 2: **for**  $t = 1$  **to**  $N$  **do**
- 3:   **for**  $\ell \in \mathcal{L}$  **do**
- 4:      $\Pi_{\text{null},\ell} \leftarrow \text{LANCZOSPROJECTION}(\bar{H}_\ell, k_\ell)$
- 5:   **end for**
- 6:    $p \leftarrow 0$
- 7:   **for**  $b \in \mathcal{T}_t$  **do**
- 8:      $\ell_b \leftarrow \mathcal{L}_b(\theta_{t-1}, p)$
- 9:      $\{g_{p,\ell}\} \leftarrow \nabla_p \ell_b$
- 10:    **for**  $\ell \in \mathcal{L}$  **do**
- 11:      $g_{p,\ell}^\perp \leftarrow \Pi_{\text{null},\ell}(g_{p,\ell})$
- 12:    **end for**
- 13:     $p \leftarrow \text{OPTIMSTEP}(p, \{g_{p,\ell}^\perp\})$
- 14:   **end for**
- 15:    $\Delta\theta_t \leftarrow \text{INJECT}(p); \quad \theta_t \leftarrow \theta_{t-1} + \Delta\theta_t; \quad p \leftarrow 0$
- 16:   **for**  $\ell \in \mathcal{L}$  **do**
- 17:      $H_{t,\ell} \approx \nabla_{\theta_\ell}^2 \mathcal{L}_{\mathcal{T}_t}(\theta_t)$
- 18:      $\bar{H}_\ell \leftarrow (1 - \eta)\bar{H}_\ell + \eta H_{t,\ell}$
- 19:   **end for**
- 20: **end for**
- 21: **return**  $\theta_N$

---

**3.4. Datasets and Evaluation Protocol**

To evaluate robustness and retention, we follow the five natural distribution shifts introduced by Wortsman *et al.* [? ]. These test sets—ImageNet-V2, ImageNet-R, ImageNet-Sketch, ObjectNet, and ImageNet-A—provide a comprehensive measure of performance under realistic domain shifts while remaining semantically aligned with ImageNet-1K.

**Datasets.** **ImageNet-1K** serves as the in-distribution (ID) reference for all evaluations. **ImageNet-V2** [? ] is a re-collection of the original validation set following identical annotation guidelines, providing a near-distribution test. **ImageNet-R** [? ] contains artistic and synthetic renditions of 200 ImageNet classes (30 k images). **ImageNet-Sketch** [? ] consists of sketch-style drawings of ImageNet objects. **ObjectNet** [? ] introduces viewpoint, pose, and background variability while controlling for object identity. **ImageNet-A** [? ] comprises naturally occurring adversarial images that elicit misclassifications in standard classifiers. Together, these datasets form a widely adopted robustness benchmark encompassing both semantic and distributional shifts.

**Evaluation protocol.** All models are fine-tuned from ImageNet-1K pretrained checkpoints using the proposed method and baselines. After training, we report Top-1 accuracy on the ImageNet-1K validation set (ID performance) and on each of the five shifted test sets (OOD performance). Following [? ], we compute the **average OOD accuracy**—the mean of accuracies across ImageNet-V2, ImageNet-R, ImageNet-Sketch, ObjectNet, and ImageNet-A—as our primary robustness metric. We additionally report the ID→OOD generalization gap (ID accuracy minus average OOD accuracy) and the standard deviation over three random seeds.

**Reporting.** Results are summarized in a single table listing Top-1 accuracy on ImageNet-1K (ID), each of the five OOD datasets, the average OOD score, ID–OOD gap, number of trainable parameters, and per-epoch runtime. For continual-stream experiments, we also include accuracy-versus-task plots and ablations over the Lanczos rank  $k$ , LoRA rank  $r$ , and EMA rate  $\eta$  to assess stability and compute trade-offs.

**References**

- [1] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1
- [2] Giulia Lanzillotta, Sidak Pal Singh, Benjamin F Grewe, and Thomas Hofmann. Local vs global continual learning. *arXiv preprint arXiv:2407.16611*, 2024. 1, 3
- [3] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1
- [4] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1