Model Tailor: Mitigating Catastrophic Forgetting in Multi-modal Large Language Models

Didi Zhu [*1 2] **Zhongyi Sun** [2] **Zexi Li** [1] **Tao Shen** [1] **Ke Yan** [2] **Shouhong Ding** [2] **Chao Wu** [1] **Kun Kuang** [1]

Abstract

Catastrophic forgetting emerges as a critical challenge when fine-tuning multi-modal large language models (MLLMs), where improving performance on target tasks often leads to a significant performance drop on the original tasks. This paper presents a comprehensive analysis of catastrophic forgetting in MLLMs and introduces a post-training adjustment method called Model Tailor. Our method primarily preserves the pre-trained parameters while replacing a small number ($\leq 10\%$) of fine-tuned parameters, maintaining $\sim 99\%$ effectiveness on original tasks versus pre-training, and achieving $\sim 97\%$ on new tasks compared to standard fine-tuning. Specifically, we derive a sparse mask to identify the *"model patch"*, based on a fusion strategy that integrates salience and sensitivity analysis. Subsequently, a compensation mechanism is introduced to *"decorate the patch"*, enhancing the model's performance on both target and original tasks. Additionally, our method is adaptable to multi-task scenarios. Through extensive experiments on InstructBLIP and LLaVA-1.5 in both image captioning and visual question answering tasks, our approach demonstrates significant task adaptability while preserving inherent pre-trained capabilities.

1. Introduction

Recent advancements in large language models (LLMs) (Devlin et al., 2018; Brown et al., 2020; Touvron et al., 2023; Wu et al., 2023; Zhu et al., 2024; 2023e) have marked a significant milestone in the pursuit of human-like artificial intelligence. This evolution has been accelerated by integrating additional modalities, particularly vision, leading to the emergence of multi-modal LLMs (MLLMs) (Li et al., 2023b; Dai et al., 2023; Liu et al., 2023b; Zhu et al., 2023a; Achiam et al., 2023). Prominent examples such as GPT-4V (Achiam et al., 2023) have played a pivotal role in enriching multi-modal dialog systems, demonstrating the versatility and depth of these advanced models. MLLMs typically comprise modality-specific sub-modules (such as a vision encoder and an LLM) to capture knowledge from different modalities, necessitating the incorporation of a lightweight projector for achieving cross-modal alignment (Li et al., 2023b; Liu et al., 2023b; Zhu et al., 2023a).

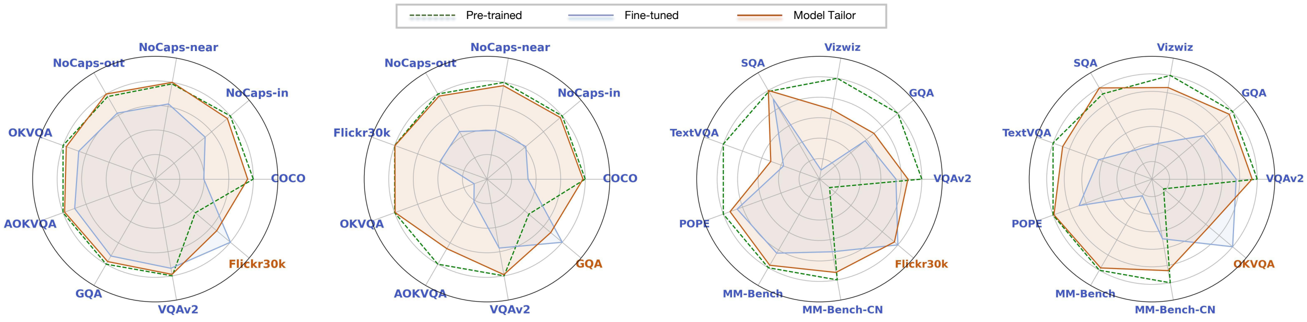
Due to the architectural complexity of MLLMs and the inherent disparities in data characteristics across modalities, these models face challenges in task generalization compared to uni-modal LLMs (Sung et al., 2023). As a result, MLLMs often demonstrate notably poor performance when faced with unseen tasks (Lin et al., 2023; Zhai et al., 2023; He et al., 2023). While fine-tuning MLLMs on these unseen data appears to be a straightforward solution, our empirical findings indicate that such approaches significantly impair the models' performance on their original tasks, as evidenced in Figure 1. This observed decline, which is defined as catastrophic forgetting in MLLMs, highlights a critical challenge that has been seldom explored. This naturally raises a pivotal question:

> *How to enhance the performance of MLLMs on target tasks without diminishing their effectiveness in original tasks?*

Current approaches to mitigate catastrophic forgetting in machine learning (Goodfellow et al., 2013; Masana et al., 2022; Yang et al., 2023) are primarily devised for small models and rely heavily on full-model fine-tuning. In MLLMs, full-model fine-tuning on new tasks leads to heightened computational demands, escalating data storage and training costs, thus becoming impractical with more downstream tasks and highlighting its inefficiency for large-scale use. Furthermore, while parameter-efficient methods like Low-Rank Adaptation (LoRA) (Hu et al., 2021) are designed to reduce computational and memory burdens, they fall short in addressing catastrophic forgetting in MLLMs. During fine-tuning with LoRA, a substantial number of redundant

[*] Work done during internship at Tencent YouTu Lab. [1] Department of Computer and Science, Zhejiang University [2] Tencent Youtu Lab. Correspondence to: Chao Wu <chao.wu@zju.edu.cn>, Kun Kuang <kunkuang@zju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).



(a) InstructBLIP on Flickr30k (b) InstructBLIP on GQA (c) LLaVA on Flickr30k (d) LLaVA on OKVQA

Figure 1: **Catastrophic Forgetting in Multi-modal Large Language Models.** After fine-tuning on two distinct tasks (in orange), InstructBLIP and LLaVa1.5 exhibit a significant performance decline on their original tasks (in blue). Our method offers a remedy to this issue, mitigating the adverse effects of catastrophic forgetting.

parameter modifications may still be retained, which ineffectively serve the target task and simultaneously erode the pre-trained knowledge (empirically verified in Figure 4).

In response to these challenges, we introduce Model Tailor—a parameter-efficient post-training method that meticulously integrates fine-tuned parameters into the pre-trained model fabric. Just as a tailor selects patches to enhance a garment, Model Tailor discerns and modifies a minimal set of parameters, known as the *"model patch"*, from the fine-tuned model to fortify the model's capabilities on new tasks without forfeiting its pre-trained proficiencies. Guided by the principles of the Lottery Ticket Hypothesis (Frankle & Carbin, 2019; Li et al., 2023a), the model patch is identified via a fusion strategy that analyzes both parameter changes and loss variations. Moreover, drawing inspiration from the Optimal Brain Surgeon (OBS) theory (LeCun et al., 1989; Hassibi et al., 1993), we reinforce the model's grasp on target task knowledge through *"patch decoration"* – a process of precise weight compensation based on the inverse of the Hessian matrix. As illustrated in Figure 1, Model Tailor maximizes the restoration of the model's performance on original tasks while also enhancing its efficacy on target tasks. Additionally, our approach demonstrates remarkable flexibility in multi-task scenarios. By stitching these distinct patches, the model can absorb task-specific knowledge while also enhancing its performance on the original task. In summary, our work makes several significant contributions:

- **Revealing Catastrophic Forgetting in MLLMs.** We present the comprehensive analysis of catastrophic forgetting in MLLMs, specifically in tasks like image captioning and visual question answering (VQA), highlighting challenges in multimodal generation and comprehension.
- **Pioneering Parameter-Efficient Solution in MLLMs.** To the best of our knowledge, Model Tailor is the first parameter-efficient method to address catastrophic forgetting in MLLMs. By implementing the "model patch" and "patch decorator", our approach effectively tackles this issue and demonstrates adaptability in multi-task scenarios.
- **Theoretical Analysis and Empirical Validation.** Through rigorous validation, we verify the effectiveness of our proposed method. Our results show significant improvements in retaining performance on original tasks while efficiently adapting to new tasks, thereby confirming the practicality and robustness of our approach.

2. Reltaed Work

Multi-modal Large Language Models. MLLMs mark a significant leap in vision-language modeling, substantially enhancing reasoning and understanding capabilities. Designed to process and interpret information across modalities, MLLMs adeptly handle complex tasks requiring deep contextual understanding. Recent advancements (Li et al., 2023b; Dai et al., 2023; Liu et al., 2023b;a; Zhu et al., 2023a; Alayrac et al., 2022; Li et al., 2023a; Achiam et al., 2023) have leveraged the formidable reasoning power of LLMs like LLaMA (Zhang et al., 2023e) and ChatGPT (Wu et al., 2023). Inspired by the notable success of instruction tuning in LLMs (Ouyang et al., 2022; Wang et al., 2022b;a), the field of MLLMs is increasingly focusing on incorporating instruction-following data to further enhance models' understanding in downstream tasks. Pioneering work in this field encompasses LLaVA (Liu et al., 2023b) and InstructBLIP (Dai et al., 2023). Recently, LLaVA-1.5 (Liu et al., 2023a) has built upon LLaVA's framework, refining its instructions to improve performance across a wider range of VQA tasks. Additionally, other studies have expanded on this concept, applying instruction tuning to different modalities such as video (Zhang et al., 2023a) and point cloud (Hong et al., 2023) to explore diverse applications. Recently, He et al. (2023) explored instruction tuning in the continual learning of MLLMs. Our research diverges from this study in terms of task setting, data assessment, and methodology. For more detailed discussions, please refer to Appendix A.

Catastrophic Forgetting. (1) **Catastrophic Forgetting in LLMs:** Recently, there has been a growing focus on addressing the problem of overfitting in LLMs when fine-tuning on small datasets (Dong et al., 2021; Korbak et al., 2022; Howard & Ruder, 2018; Lee et al., 2019; Zhang et al., 2020) or after in-context learning (Alayrac et al., 2022; Wang et al., 2023). The most advanced methods have endeavored to combat this challenge by employing techniques such as learning a sparse mask through loss optimization or randomly selecting masks (Panigrahi et al., 2023) and rescaling parameters to achieve model fusion (Yu et al., 2023). Nevertheless, these methodologies have shown limited effectiveness when applied to MLLMs, owing to the intricate architecture and non-uniform weight distributions arising from their cross-modal nature (Sung et al., 2023). (2) **Catastrophic Forgetting in MLLMs:** Few studies (He et al., 2023; Lin et al., 2023; Zhai et al., 2023) explore this phenomenon in MLLMs. He et al. (2023) delves into traditional continual learning, which is distinct from our focus. Lin et al. (2023) examines CLIP's performance in image classification tasks, not extending to text generation models like LLaVA. Meanwhile, Zhai et al. (2023) investigates the performance of LLaVA and InstructBLIP in image classification tasks, without exploring deeper into multimodal tasks. Crucially, Lin et al. (2023) and Zhai et al. (2023) mainly apply existing anti-forgetting methods to MLLMs, concentrating on analytical approaches without proposing novel solutions. In contrast, our work focuses on the fundamental problem of catastrophic forgetting in multimodal generation and reasoning tasks in MLLMs, and we are the first parameter-efficient work to tackle this challenge. For an overview of catastrophic forgetting in smaller models, please refer to Appendix A.

3. Problem Formulation

Given a Multi-modal Large Language Model $\mathcal{M}$ either pre-trained or demonstrating strong generalization capabilities on a series of tasks $\mathcal{P} = \{D_1, D_2, \cdots, D_n\}$, where the model parameters are denoted by $\mathbf{\Theta}_{\text{pre}}$. Catastrophic forgetting in MLLMs is characterized by a significant degradation in performance on previously learned tasks after undergoing supervised fine-tuning (SFT) for a new and unseen task $\mathcal{T} = D^{\text{new}}$, which updates the model's parameters to $\mathbf{\Theta}_{\text{sft}}$. To counteract this, we aim to design a fusion function $\mathcal{F}(\cdot)$, formulated as follows:

$$\mathbf{\Theta}_{\text{fusion}} = \mathcal{F}(\mathbf{\Theta}_{\text{sft}}, \mathbf{\Theta}_{\text{pre}}), \tag{1}$$

where $\mathbf{\Theta}_{\text{fusion}}$ denotes the optimally fused parameters. The primary goal of $\mathcal{F}(\cdot)$ is to enhance the performance of $\mathcal{M}$ on a downstream task $\mathcal{T}$, as well as preserve the pre-trained knowledge on original tasks $\mathcal{P}$.

Formally, the optimization objective is to minimize the loss on $\mathcal{T}$, while ensuring minimal deviation from the original task performances, which can be expressed as:

$$\begin{aligned} &\mathcal{L}_{\mathcal{T}}(\mathbf{\Theta}_{\text{fusion}}) - \mathcal{L}_{\mathcal{T}}(\mathbf{\Theta}_{\text{sft}}) \leq \epsilon_{\text{t}}, \\ &\mathcal{L}_{\mathcal{P}}\left(\mathbf{\Theta}_{\text{fusion}}\right) - \mathcal{L}_{\mathcal{P}}\left(\mathbf{\Theta}_{\text{pre}}\right) \leq \epsilon_{p}, \end{aligned} \tag{2}$$

where $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{L}_{\mathcal{P}}$ denote the loss functions corresponding to the downstream task $\mathcal{T}$ and the pre-training tasks $\mathcal{P}$. ϵ_t and ϵ_p denote the acceptable performance degradation margins for $\mathcal{T}$ and $\mathcal{P}$ post-fusion.

4. Method

4.1. Overview of Model Tailor

Inspired by the Lottery Ticket Hypothesis (Frankle & Carbin, 2019; Li et al., 2023a) and the Optimal Brain Surgeon (LeCun et al., 1989; Hassibi et al., 1993) (refer to Appendix B for related preliminaries), Model Tailor akin to a skilled tailor carefully selects and decorates patches of new fabric (fine-tuned parameters) and seamlessly integrating them onto the existing garment (pre-trained model). Model Tailor consists of two primary steps as illustrated in Figure 2:

Step 1: Identify model patch. The initial step is akin to selecting the ideal piece of fabric, identifying a crucial subset of the model's fine-tuned parameters critical for enhancing target task performance. This crucial subset is termed as the *"model patch"*.

Step 2: Decorate the patch. After identifying the patch, we apply targeted compensation to this selected subset. The second step is designed to alleviate the loss rise on the target task due to the removal of other non-selected fine-tuned parameters. This compensatory step known as *"patch decorator"*, is akin to adding embellishments to a garment, enhancing the model's capabilities on the target task.

Through the aforementioned two steps, Model Tailor can adeptly fine-tune the model's performance on the target task while meticulously preserving the core integrity of its pre-trained knowledge base. Formally, the fusion process in Model Tailor can be expressed as:

$$\begin{aligned} \mathbf{\Theta}_{\text{fusion}} &= \mathcal{F}(\mathbf{\Theta}_{\text{sft}}, \mathbf{\Theta}_{\text{pre}}) \\ &= \boldsymbol{M} \odot (\mathbf{\Theta}_{\text{sft}} + \boldsymbol{C}) + (\boldsymbol{I} - \boldsymbol{M}) \odot \mathbf{\Theta}_{\text{pre}}, \end{aligned} \tag{3}$$

where $\boldsymbol{I}$ represents the identity matrix and $\boldsymbol{M}$ is a sparse binary mask used to identify the model patch, while $\boldsymbol{C}$ represents the patch decorator, referring to the compensation values for the parameters selected by $\boldsymbol{M}$.

4.2. Layer-wise Objective of Model Tailor

Due to the large scale of $\mathcal{M}$, calculating masks and compensation values poses a considerable computational challenge. To mitigate this, we leverage a pruning-inspired strategy (Frantar & Alistarh, 2022; 2023), breaking down Model

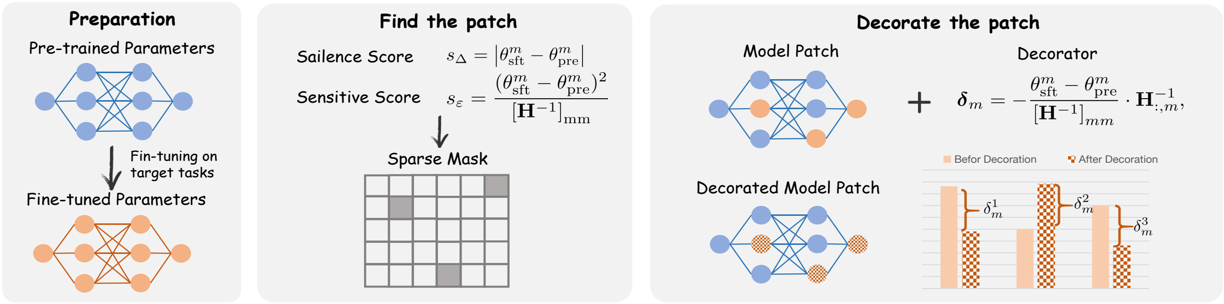

Preparation
Pre-trained Parameters
Fin-tuning on target tasks
Fine-tuned Parameters
Find the patch
Sailence Score $s_{\Delta} = |\theta^m_{\text{sft}} - \theta^m_{\text{pre}}|$
Sensitive Score $s_{\varepsilon} = \frac{(\theta^m_{\text{sft}} - \theta^m_{\text{pre}})^2}{[\mathbf{H}^{-1}]_{\text{mm}}}$
Sparse Mask
Decorate the patch
Model Patch
Decorator
$+ \quad \boldsymbol{\delta}_m = -\frac{\theta^m_{\text{sft}} - \theta^m_{\text{pre}}}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}^{-1}_{:,m},$
Decorated Model Patch
Befor Decoration
After Decoration
δ^1_m
δ^2_m
δ^3_m

Figure 2: **Overall Framework of Model Tailor.** Model Tailor consists of two primary steps. The first step in this process focuses on the identification of a "model patch", which is defined as a critical subset of fine-tuned parameters deemed essential for improving the model's effectiveness on a given target task. The second step is dedicated to applying compensatory adjustments, a methodological intervention designed to counterbalance any potential performance deficits that may arise from the exclusion of certain parameters during the fine-tuning phase.

Tailor into more manageable layer-wise sub-tasks. For each layer ℓ in $\mathcal{M}$, we define the layer function as $f_\ell\left(X^\ell_{\mathcal{T}}, \mathbf{\Theta}^\ell_{\text{sft}}\right)$, where $X^\ell_{\mathcal{T}}$ signifies the layer inputs for the target task and $\mathbf{\Theta}^\ell_{\text{sft}}$ represents the corresponding layer weights in layer ℓ. The expectation over the layer inputs $X^\ell_{\mathcal{T}}$ is usually approximated by taking the mean over a small set of N input samples (Frantar & Alistarh, 2022). Our goal is to refine $\mathbf{\Theta}^\ell_{\text{fusion}}$ to achieve performance as close as possible to the fully fine-tuned weights for the target task and fully pre-trained weights for the original tasks. Formally, we seek to optimize the weights $\mathbf{\Theta}^\ell_{\text{fusion}}$ such that they minimize any discrepancies in expected layer output, which can be expressed as:

$$\begin{aligned} &\arg\min_{\mathbf{\Theta}^\ell_{\text{fusion}}} \mathbb{E}_{\boldsymbol{X}^\ell_{\mathcal{T}}} \mathcal{L}\left(f_\ell\left(\boldsymbol{X}^\ell_{\mathcal{T}}, \mathbf{\Theta}^\ell_{\text{sft}}\right), f_\ell\left(\boldsymbol{X}^\ell_{\mathcal{T}}, \mathbf{\Theta}^\ell_{\text{fusion}}\right)\right), \\ &\quad \text{s.t.}\ \mathbb{E}_{\boldsymbol{X}^\ell_{\mathcal{P}}} \mathcal{L}\left(f_\ell\left(\boldsymbol{X}^\ell_{\mathcal{P}}, \mathbf{\Theta}^\ell_{\text{pre}}\right), f_\ell\left(\boldsymbol{X}^\ell_{\mathcal{P}}, \mathbf{\Theta}^\ell_{\text{fusion}}\right)\right) < \epsilon, \end{aligned} \tag{4}$$

where $\mathbf{\Theta}^\ell_{\text{fusion}} = \mathcal{F}(\mathbf{\Theta}^\ell_{\text{sft}}, \mathbf{\Theta}^\ell_{\text{pre}})$. $\boldsymbol{X}^\ell_{\mathcal{P}}$ is the layer input on pre-trained tasks. ϵ is a threshold. $\mathcal{L}(\cdot)$ represents a specific loss function, commonly employed as the squared loss in many works (Hubara et al., 2021; Nagel et al., 2020; Wang et al., 2020). The constraint imposed on this optimization ensures that the performance on the pre-training tasks $\mathcal{P}$ does not degrade beyond ϵ. This optimization target aligns with the global objective introduced in Equation 2, adapting it to a layer-wise perspective with a focus on output consistency.

It is worth noting that the practical implementation of Equation 4 faces obstacles due to the unavailability of pre-training data $\boldsymbol{X}^\ell_{\mathcal{P}}$, often restricted by proprietary or privacy considerations. Therefore, we approximate this constraint by limiting the distance between fusion parameters $\mathbf{\Theta}^\ell_{\text{fusion}}$ and pre-trained parameters $\mathbf{\Theta}^\ell_{\text{pre}}$. In other words, our goal is to retain as many pre-trained parameters as possible to maintain the model's performance on the original task. Hence, the layer-wise optimization objective can be explicitly formulated as follows:

$$\begin{aligned} &\arg\min_{\mathbf{\Theta}^\ell_{\text{fusion}}} \left\|\mathbf{\Theta}^\ell_{\text{sft}}\mathbf{X}^\ell_{\mathcal{T}} - \mathbf{\Theta}^\ell_{\text{fusion}}\mathbf{X}^\ell_{\mathcal{T}}\right\|^2_2, \\ &\quad \text{s.t.}\ \|\mathbf{\Theta}^\ell_{\text{fusion}} - \mathbf{\Theta}^\ell_{\text{pre}}\|_1 < \eta, \end{aligned} \tag{5}$$

where η represents a constraint parameter that denotes the allowable distance or deviation between the fusion parameters $\mathbf{\Theta}^\ell_{\text{fusion}}$ and the pre-trained parameters $\mathbf{\Theta}^\ell_{\text{pre}}$. This constraint ensures that the fusion parameters stay within a certain range of the pre-trained parameters.

4.3. Identify Model Patch

Building upon the introduction in subsection 4.1, we provide the formal definitions of model patch and layer patch:

Definition 4.1 (Model Patch)**.** Given a MLLM $\mathcal{M}$ with its pre-trained parameters $\mathbf{\Theta}_{\text{pre}}$ and fine-tuned parameters $\mathbf{\Theta}_{\text{sft}}$, A *model patch* $\mathbf{\Theta}_{\text{patch}} \subseteq \mathbf{\Theta}_{\text{sft}}$, is identified through a binary mask $\boldsymbol{M} \in \{0,1\}^{|\mathbf{\Theta}_{\text{sft}}|}$, defined as:

$$\mathbf{\Theta}_{\text{patch}} = \{\theta_i | M_i = 1, \theta_i \in \mathbf{\Theta}_{\text{sft}}\}, \tag{6}$$

where M_i denotes the i-th element of $\boldsymbol{M}$. $\mathbf{\Theta}_{\text{patch}}$ captures the most crucial subset of fine-tuned parameters for the target task, thereby facilitating the achievement of the global objective expressed in Equation 2.

Then, we extend this concept to individual layers within the model, introducing the notion of a Layer Patch.

Definition 4.2 (Layer Patch)**.** For each layer ℓ in an MLLM, characterized by layer-specific pre-trained parameters $\mathbf{\Theta}^\ell_{\text{pre}}$ and fine-tuned parameters $\mathbf{\Theta}^\ell_{\text{sft}}$, a layer patch $\mathbf{\Theta}^\ell_{\text{patch}} \subseteq \mathbf{\Theta}^\ell_{\text{sft}}$ is derived by $\boldsymbol{M}^\ell \in \{0,1\}^{|\mathbf{\Theta}^\ell_{\text{sft}}|}$, expressed as:

$$\mathbf{\Theta}^\ell_{\text{patch}} = \{\theta^i_{\text{sft}} | M^\ell_i = 1, \theta^i_{\text{sft}} \in \mathbf{\Theta}^\ell_{\text{sft}}\}, \tag{7}$$

with M^ℓ_i being the i-th element of $\boldsymbol{M}^\ell$. Similarly, $\mathbf{\Theta}^\ell_{\text{patch}}$ enables the achievement of the layer-wise optimization objective expressed in Equation 5.

We are naturally led to a crucial question: how do we identify this layer patch, or more specifically, how do we determine the mask $\boldsymbol{M}^{\ell}$? This inquiry propels us to investigate two principal mask selection strategies.

Sensitivity-based Score. Drawing from the principles of OBS, we introduce a sensitivity based score, s_{ε}, as a measure to evaluate the impact of reverting a fine-tuned parameter θ_{sft} to its pre-tuning state θ_{pre} on the target task performance. We present Theorem 4.3 proved in Appendix C, which offers second-order analytical lens to evaluate the significance of each parameter.

Theorem 4.3. *Consider a layer ℓ within an MLLM $\mathcal{M}$, and let θ_m represent a parameter at index m. Altering θ_m from its fine-tuned state $\theta^m_{\textit{sft}}$ to its pre-trained state $\theta^m_{\textit{pre}}$, induces a increase $\Delta\mathcal{L}_{\mathcal{T}}$ in the model's loss for $\mathcal{T}$, quantified as:*

$$\Delta\mathcal{L}_{\mathcal{T}} = \frac{(\theta^m_{\textit{sft}} - \theta^m_{\textit{pre}})^2}{[\mathbf{H}^{-1}]_{\textit{mm}}}, \tag{8}$$

where $\mathbf{H}^{-1}$ denotes the inverse of the Hessian matrix and $\left[\mathbf{H}^{-1}\right]_{mm}$ is its m-th diagonal element. The Hessian matrix $\mathbf{H} = \nabla^2_{\theta^m_{\textit{sft}}}\mathcal{L}_{\mathcal{T}}$ takes into account the local geometry of the loss at a given point $\theta^m_{\textit{sft}}$ assuming $\mathcal{L}_{\mathcal{T}}$ is twice differentiable.

The optimal selection of the parameter index m that results in the smallest increase in loss. This increase in loss is quantified as the sensitivity-based statistic s_{ε}:

$$s_{\varepsilon} = \frac{(\theta^m_{\text{sft}} - \theta^m_{\text{pre}})^2}{2\left[\mathbf{H}^{-1}\right]_{mm}}. \tag{9}$$

The value of s_{ε} is indicative of how the adjustment of the m-th parameter influences the model's task-specific performance, with higher scores highlighting parameters of paramount importance to the model's efficacy.

Salience-based Score. Our second strategy employs a salience based score that measures the absolute deviation of fine-tuned parameters from their pre-trained counterparts, focusing on those that have been significantly altered to adapt to the specifics of the target task. This deviation is quantitatively captured for each parameter $\theta^m_{\text{sft}} \in \boldsymbol{\Theta}^{\ell}_{\text{sft}}$ within layer ℓ, by computing:

$$s_{\Delta} = \left|\theta^m_{\text{sft}} - \theta^m_{\text{pre}}\right|, \tag{10}$$

where s_{Δ} serves as an indicator of a parameter's adaptability. This metric underscores the parameters that have shown substantial responsiveness to the task adaptation process, thereby shedding light on their pivotal role in the model's learning evolution for the new task.

Score Fusion. To discern the most impactful parameters for a layer ℓ, we devise a hybrid importance score, s_{h}, synthesized from two above perspectives:

$$s = \omega \cdot \tilde{s}_{\Delta} + (1 - \omega) \cdot \tilde{s}_{\varepsilon}, \tag{11}$$

where $\tilde{s}_{\Delta}$ and $\tilde{s}_{\varepsilon}$ represent the min-max normalized salience and sensitivity scores, respectively. ω is a weighting factor that adjusts the balance between the $\tilde{s}_{\Delta}$ and $\tilde{s}_{\varepsilon}$.

Let ρ represent the predefined sparsity level, indicating the percentage of parameters to be removed. The process of determining the final mask M^{ℓ} for layer ℓ can be expressed as: $\boldsymbol{M}^{\ell}_i = \mathbb{1}\left\{s^i \geq s'\right\}$. $\mathbb{1}\{\cdot\}$ is the indicator function, assigning a value of 1 when the condition is met and 0 otherwise. s' denotes the score threshold, determined by the ρ-th highest score. By prioritizing parameters above the hybrid score threshold, we approximate the layer patch outlined in Definition 4.2, focusing solely on parameters vital for the target task's performance.

4.4. Decorate the Patch

Once identifying the layer patch $\Theta^{\ell}_{\text{patch}}$, the subsequent step involves optimizing these selected parameters to ensure the model's performance on the target task remains robust. Then, we have the definition of the patch decorator, a crucial mechanism for enhancing the layer's effectiveness.

Definition 4.4 (Patch Decorator)**.** Given a layer ℓ in an MLLM and the corresponding layer patch $\boldsymbol{\Theta}^{\ell}_{\text{patch}}$ identified by the binary mask $\boldsymbol{M}^{\ell}$, the patch decorator δ^i is the adjustment for each selected parameter $\theta^i_{\text{sft}} \in \boldsymbol{\Theta}^{\ell}_{\text{patch}}$, yielding the ajusted layer patch:

$$\widehat{\boldsymbol{\Theta}}^{\ell}_{\text{patch}} = \{\theta^i_{\text{sft}} + \delta^i | \boldsymbol{M}^{\ell}_i = 1, \theta^i_{\text{sft}} \in \boldsymbol{\Theta}^{\ell}_{\text{sft}}\}. \tag{12}$$

The patch decorator is designed to compensate the increased loss in target performance resulting from excluding parameters not covered by the binary mask $\boldsymbol{M}^{\ell}$, i.e., $\boldsymbol{\Theta}^{\ell}_{\text{sft}} \setminus \boldsymbol{\Theta}^{\ell}_{\text{patch}}$.

Theorem 4.3 delineates the change in loss resulting from the alteration of a parameter from its fine-tuned state to its pretrained state. Building upon this, we present Theorem 4.5 proved in Appendix C, which specifies the compensatory values for the remaining parameters.

Theorem 4.5. *Given the alteration of a parameter from its fine-tuned state, $\theta^{\textit{ft}}_m$, to its pre-trained state, $\theta^{\textit{pre}}_m$, the optimal perturbation $\Delta\boldsymbol{\Theta}$ for each retained fine-tuned parameters is calculated as follows:*

$$\Delta\boldsymbol{\Theta}^*_m = -\frac{\theta^m_{\textit{sft}} - \theta^m_{\textit{pre}}}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}^{-1}_{:,m}, \tag{13}$$

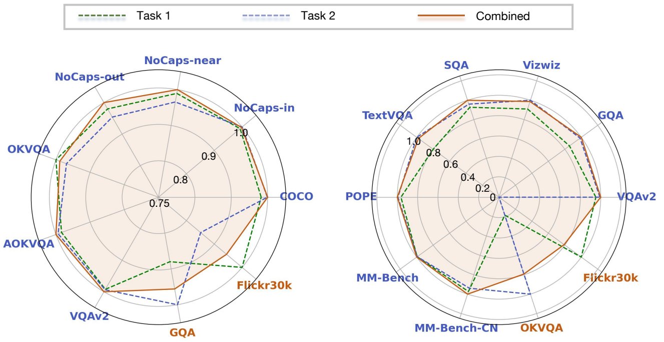
where $\mathbf{H}^{-1}$ denotes the inverse of the Hessian matrix and $\mathbf{H}^{-1}_{:,m}$ signifies the m-th column of $\mathbf{H}^{-1}$.

Theorem 4.5 serves as the foundational pillar for determining the patch decorator in Model Tailor. In fact, the

Table 1: **H-score and average performance on InstructBLIP (Vicuna-7B) with a sparsity ratio $\rho = 10\%$.** "#Params" refers to the number of parameters modified. The optimal and sub-optimal results are denoted by boldface and underlining.

Method	#Params	**Pre-trained tasks**								**Target task**		
		COCO	NoCaps-in	NoCaps-near	NoCaps-out	OKVQA	AOKVQA	VQAv2	GQA	Flickr30k	Avg	Hscore
Zero-shot	-	143.1	117.7	123.5	121.9	58.45	58.24	76.66	49.18	82.8	92.39	87.87
Fine-tune	288M	114.2	101.9	113.2	112.3	54.47	55.30	74.26	47.29	101.8	86.08	92.12
DARE	28.8M	114.1	103.3	113.5	109.2	58.07	55.89	76.65	49.19	101.3	86.8	92.43
Grafting	28.8M	127.9	112.6	121.5	116.3	53.58	54.58	71.90	47.01	97.9	<u>89.25</u>	<u>92.78</u>
Ours	28.8M	139.8	115.8	124.3	123.4	57.63	57.84	76.19	48.58	94.4	**93.10**	**93.67**

Method	#Params	**Pre-trained tasks**								**Target task**		
		COCO	NoCaps-in	NoCaps-near	NoCaps-out	OKVQA	AOKVQA	VQAv2	Flickr30k	GQA	Avg	Hscore
Zero-shot	-	143.1	117.7	123.5	121.9	58.45	58.24	76.66	82.8	49.18	92.39	65.44
Fine-tune	288M	109.8	94.9	99.0	100.4	38.32	41.13	67.85	66.6	59.81	75.31	67.42
DARE	28.8M	140.2	115.8	119.3	118.5	58.01	57.25	74.74	81.8	50.91	<u>90.72</u>	<u>66.46</u>
Grafting	28.8M	138.5	114.3	119.2	119.5	55.09	57.06	74.36	80.3	50.82	89.90	66.16
Ours	28.8M	141.8	116.5	121.8	120.7	58.33	53.98	76.28	82.7	56.15	**92.02**	**71.00**



(a) InstructBLIP

(b) LLaVA

Figure 3: **Model Tailor on Multi-Task Scenario.** "*Performance oscillations*" where models exhibit dips in efficacy on one task after fine-tuning on another, which are effectively bridged by Model Tailor's multi-task fusion.

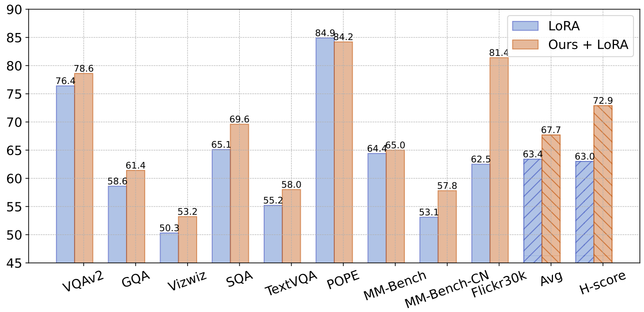


Figure 4: **Combination with LoRA on LLaVA.** The application of Model Tailor to LLaVA-1.5 fine-tuned using LoRA yields significant performance improvements across various datasets, indicating that Model Tailor's post-training refinement complements LoRA.

patch decorator δ^i represents the i-th value within the vector $\Delta\Theta^*_m$. By applying $\Delta\Theta^*_m$ to the selected sparse parameters, our method ensures a minimally disruptive integration of the fine-tuned parameters with the pre-trained model.

Additionally, Model Tailor can be extended to multi-task scenarios to absorb and integrate knowledge from multiple tasks, details described in Appendix B.

5. Experiments

5.1. Experimental Setup

Architectures and Datasets. Given the representativeness of InstructBLIP (Vicuna-7B) and LLaVA-1.5 (Vicuna-7B) in terms of structure and the scope of fine-tuned parameters—where the former fine-tunes the projector and the latter adjusts both the projector and LLM, we evaluate Model Tailor on these two models. For InstructBLIP, following (Dai et al., 2023), we engage with datasets including COCO Caption (Lin et al., 2014), NoCaps (in, near, out) (Agrawal et al., 2019), OKVQA (Marino et al., 2019), AOKVQA (Schwenk et al., 2022), GQA (Hudson & Manning, 2019), VQAv2 (Goyal et al., 2017), and Flickr30k (Young et al., 2014). Specifically, we fine-tune the models on Flickr30k and GQA - datasets associated with image captioning and VQA tasks and not encountered during the pre-training phase of InstructBLIP. Similarly, for LLaVA-1.5, in line with (Liu et al., 2023a), the datasets involved are VQAv2, GQA, Vizwiz (Gurari et al., 2018), SQA (Lu et al., 2022), TextVQA (Singh et al., 2019), POPE (Li et al., 2023c), MM-Bench (Liu et al., 2023c), and MM-Bench-CN (Zhang et al., 2023d). We fine-tune LLaVA on Flickr30k and OKVQA tasks, both distinct from its pre-training exposure, and assess its performance on the remaining datasets.

Compared Baselines. We compare Model Tailor against three methods including: (a) **Standard Fine-Tuning**: For InstructBLIP, the fine-tuning process targets the Q-Former, a

Table 2: **H-score and average performance on LLaVA-1.5 (Vicuna-7B) with a sparsity ratio $\rho = 10\%$.** "#Params" refers to the number of parameters modified. The optimal and sub-optimal results are denoted by boldface and underlining.

Method	#Params	Pre-trained tasks								Target task		
		VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	Flickr30k	Avg	Hscore
Zero-shot	-	78.54	61.94	50.0	66.8	58.27	85.9	64.3	58.3	3.5	58.62	6.64
Fine-tune	2.7B	68.61	49.01	27.24	63.86	40.03	79.73	59.02	50.17	77.1	56.42	63.40
DARE	273M	78.12	59.25	48.9	64.92	57.17	84.86	64.77	57.47	25.6	60.12	36.64
Grafting	273M	74.48	58.28	43.16	66.82	52.56	80.35	64.52	55.49	58.2	61.56	60.03
Ours	273M	73.21	52.49	42.28	67.15	43.89	82.88	63.40	56.15	75.4	**61.87**	**66.94**

Method	#Params	Pre-trained tasks								Target task		
		VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	OKVQA	Avg	Hscore
Zero-shot	-	78.54	61.94	50.0	66.8	58.27	85.9	64.3	58.3	0.14	58.24	0.28
Fine-tune	2.7B	69.1	48.61	30.35	41.03	42.13	72.33	32.79	43.47	46.27	47.34	46.87
DARE	273M	78.04	61.65	49.19	67.58	57.91	86.44	65.03	58.16	0.83	58.31	1.64
Grafting	273M	75.23	58.42	43.27	67.26	53.51	85.29	62.16	54.42	30.8	58.93	41.25
Ours	273M	76.25	60.39	46.49	69.51	54.88	85.44	63.32	54.21	38.1	**60.95**	**47.71**

Table 3: **Computation Cost Analysis.** Model Tailor doesn't impose a significant computational burden.

	Training time (#GPU × s)	Post-training time (#GPU × s)
Fine-tune	4 × 18720	-
LoRA	4 × 4020	-
DARE	4 × 18720	1 × 9240
Grafting	4 × 18720	1 × 780
Ours	4 × 18720	1 × 1260

Table 4: **Ablation Study of Model Tailor.** The effectiveness of decorator is critical on InstructBLIP and LLaVA.

	InstructBLIP on Flickr30k			
	Pre-trained Tasks	Tagrte Task	Avg	H-score
Before Decoration	90.75	93.21	91.02	91.96
After Decoration	92.94	94.40	**93.10**	**93.67**
	LLaVA on Flickr30k			
	Pre-trained Tasks	Target Task	Avg	H-score
Before Decoration	58.26	71.58	59.74	64.24
After Decoration	60.18	75.40	**61.87**	**66.94**

connector bridging the vision encoder and the LLM, involving an extensive parameter set of approximately 288M. For LLaVA-1.5, fine-tuning is applied to both the MLP mapper and the final 12 layers of the LLM, constrained by GPU memory limits, involving a significant total of 2.7 billion parameters. (b) **Model Grafting** (Panigrahi et al., 2023): This strategy optimizes the target task loss to derive a mask, which identifies the most pivotal parameters for the task at hand. (c) **Drop & Rescale (DARE)** (Yu et al., 2023): In this approach, parameters from the fine-tuned model are randomly selected and rescaled to maintain the model's performance on both the target task and other tasks.

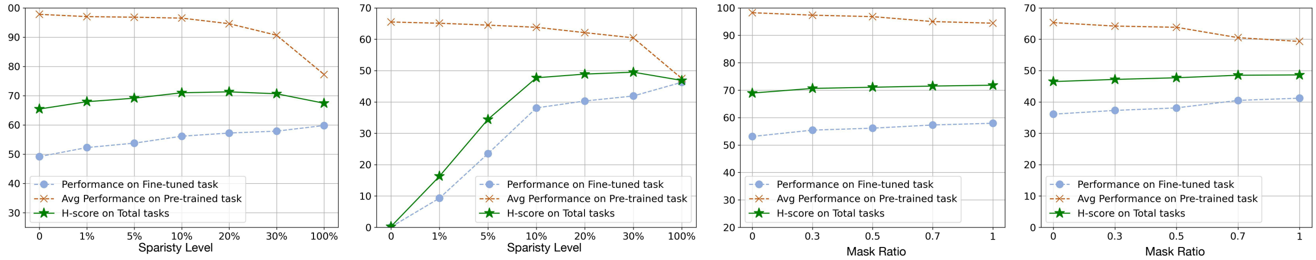
Evaluation Metrics. We utilize the arithmetic and harmonic means of performance across pre-trained and target tasks. For more details, please refer to Appendix D.

Implementation Details. Given the computational intensity of calculating the inverse Hessian matrix, our method incorporates an optimization technique derived from SparseGPT (Frantar & Alistarh, 2023), adept at handling the complexities of large-scale models, thereby ensuring the practical feasibility of Model Tailor. Please see Appendix D for detailed analysis and other implementation details.

5.2. Main Results on Single-Task setting

Table 1 and Table 2 illustrate that Model Tailor substantially mitigates catastrophic forgetting in MLLMs, outperforming current fine-tuning and forgetting mitigation methods in both InstructBLIP and LLaVA frameworks under a sparsity level of 10%. We provide further important observations:

- **Generalization capabilities of MLLMs raise significant concerns, particularly in adapting to unfamiliar tasks.** This is highlighted by the zero-shot performance of LLaVA on Flickr30k, and OKVQA is virtually null as shown in Table 2. Furthermore, subsequent fine-tuning to improve performance on such tasks often leads to a detrimental impact on the proficiency of pre-trained tasks. For example, LLaVA's performance on Vizwiz, as shown in Table 2, falls sharply from a baseline score of 50.0 to 27.24 and 30.35 after fine-tuning for Flickr30k and OKVQA, respectively. Similarly, the performance of InstructBLIP on COCO, after undergoing fine-tuning, also experiences a significant drop, descending from a high of 143.1 to 114.2 and 109.8, as reported in Table 1.
- **Existing approaches to prevent forgetting demonstrate limitations when applied to MLLMs.** DARE, except in the case of InstructBLIP on Flickr30k, fails to sustain performance on target tasks, with LLaVA's performance on OKVQA dropping back to baseline after DARE's selection, as presented in Table 2. Grafting, while offering some preservation of performance on target tasks, still



(a) InstructBLIP on GQA (b) LLaVA on OKVQA (c) InstructBLIP on GQA (d) LLaVA on OKVQA

Figure 5: **Comparison the performance of Model Tailor in various settings.** (a-b) Results of InstructBLIP and LLaVA-1.5 at varying sparsity levels. (c-d) Results of InstructBLIP and LLaVA-1.5 with different mask proportions.

leads to considerable forgetting on tasks for which the model was originally trained.

- **Model Tailor adeptly optimizes for specific tasks while preserving performance on pre-trained tasks.** It achieves superior H-scores and average metrics, highlighting its effectiveness in balancing focused enhancements with foundational robustness. This dual optimization is consistently reflected across InstructBLIP and LLaVA, confirming the method's comprehensive adaptability and excellence in the field.

5.3. Main Results on Multi-Task Setting

Figure 3 presents Model Tailor's adeptness in multi-task environments, utilizing relative performance indicators for clarity.The comparisons are conducted under a consistent sparsity level: each single task employs a sparsity of 10% and the multi-task fusion combines from two 5% sparsity tasks. The radial plots disclose an apparent oscillation in performance—models fine-tuned for one task show dips in efficacy when assessed on another, creating what can be termed as "*performance oscillations*". However, when Model Tailor's multi-task fusion is applied, these *performance oscillations* are seamlessly bridged. This not only rectifies the disparities but also leads to superior performance on the pre-trained tasks, suggesting that our method's integration of diverse knowledge domains significantly bolsters the model's robustness. Detailed quantitative assessments are provided in Table 5 and Table 6 in Appendix D, which further substantiate the visual observations from Figure 3.

5.4. Synergy with Parameter-Efficient Method

Model Tailor enhances the efficacy of existing parameter-efficient fine-tuning strategies such as LoRA (Hu et al., 2021). LoRA freezes pre-trained model weights and introduces trainable rank decomposition matrices, reducing parameters for downstream tasks. Specifically, we select and modify parameters on the model fine-tuned with LoRA. This synergy, shown in Figure 4, particularly benefits the LLaVA fine-tuned on Flickr30k, where applying Model Tailor results in notable gains across a suite of datasets. Specifically, on Flickr30k, we achieved an 18.9-point increase, and the overall H-score improved by 9.9%. These findings imply that a subset of parameters modified during LoRA's fine-tuning could be extraneous, which Model Tailor adeptly identifies and excludes. Moreover, the Model Tailor's post-training refinement complements the fine-tuning conducted during LoRA, leading to improved model synergy. For a detailed comparison, Table 7 in Appendix D provides a performance breakdown, highlighting the benefits of our approach in two fine-tuning scenarios for LLaVA.

5.5. Ablation Study

Effectiveness of Decoration. Table 4 elucidates the function of the decoration phase in Model Tailor, distinguishing between the pre-decorated state, "Before Decoration", where only mask selection has occurred, and "After Decoration", which involves weight compensation of the mask-selected parameters. The table clearly illustrates the efficacy of the decoration process, which significantly elevates model performance. This phase is crucial for ensuring that it adapts effectively to target tasks. More visualization results are provided in Figure 6 in Appendix D.

Influence of Sparsity Level. Figure 5 (a-b) examine the effects of sparsity on Model Tailor's performance. An intriguing pattern is discernible from the data: as sparsity levels increase, there is a corresponding enhancement in performance on tasks for which the model was initially trained. However, there is a noticeable decline in the model's effectiveness on new, target tasks. This trend is particularly pronounced when sparsity levels exceed 10%, at which point a performance plateau is observed across all tasks. A performance plateau is observed across all tasks once sparsity exceeds 10%. This plateauing effect is accentuated in LLaVA, particularly due to its subpar generalization on target tasks. The significant gains in performance from 0 to 10% sparsity highlight the presence of superfluous parameters, suggesting that fine-tuning incorporates a considerable amount of redundancy.

Trade-off on Mask Selection. The impact of the two distinct masking mechanisms employed in the process of identifying model patches is further explored in this study. As depicted in Figure 5 (c-d), we study the influence of changes in the ω parameter in Equation 11. An increased ω value, giving more weight to the salience-based score, correlates with improved performance on target tasks. Conversely, a reduced ω value, which gives more weight to the sensitivity-based score, enhances performance on pre-trained tasks. These findings suggest that salience-based scores are instrumental in honing the model's capabilities for new tasks, while sensitivity-based scores fortify the foundational knowledge of pre-trained tasks. Achieving a balanced combination of these scores results in a harmonious state in which both new and pre-existing task performances are optimized. This equilibrium ensures that the model simultaneously maintains its proficiency on tasks it was originally trained for, while adeptly adapting to new tasks.

5.6. Computation Cost

Computational Complexity. Computing the full Hessian matrix in Model Tailor is prohibitively expensive and impractical. In our implementation, we utilized a technique known as SparseGPT (Frantar & Alistarh, 2023), which is a layer-wise partial update method, significantly reducing the computational complexity of calculating the Hessian matrix. By adopting SparseGPT, we managed to reduce the computational complexity from $\mathcal{O}(nd^2)$ to $\mathcal{O}(d^3)$, where d represents the size of the model's hidden layer. This optimization makes the computation of the Hessian matrix feasible even with a vast number of parameters. The implementation details are provided in Appendix D.

Training Efficiency. To further demonstrate our method's efficiency, we present comparative actual training and post-training times of the LLaVA model in Table 3. It is important to note that our method is designed as a post-training adjustment, independent of the training methods such as standard fine-tuning or LoRA. In our primary experiments, we utilize the standard full fine-tuning approach for the initial training phase. Following this, the table demonstrates that the post-training time of our method is competitive when compared to other baselines, showcasing the effectiveness of our post-training optimization.

6. Conclusion

In conclusion, this study tackles the pressing issue of catastrophic forgetting during the fine-tuning process of MLLMs. We have developed a novel approach named Model Tailor, which thoughtfully preserves the core integrity of pre-trained parameters while allowing for selective fine-tuning of others that are crucial for task-specific optimization. This method employs a hybrid strategy that incorporates both preservation and adaptation techniques. Specifically, Model Tailor utilizes a carefully designed sparse mask that accurately identifies the parameters most critical for adaptation to new tasks. This targeted approach not only maintains essential pre-existing knowledge but also ensures that the adaptation is focused and efficient. The flexibility of the sparse mask allows it to be effectively applied across a variety of tasks without necessitating a complete retraining of the model. Furthermore, we implement a weight compensation mechanism on this selectively curated subset of parameters. The empirical results from our experiments corroborate the robustness and effectiveness of Model Tailor.

Acknowledgements

This work was supported by the National Key Research and Development Project of China (2021ZD0110505), the National Natural Science Foundation of China (62441605, 62376243), the Zhejiang Provincial Key Research and Development Project (2023C01043), and the Academy Of Social Governance Zhejiang University.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Agrawal, H., Desai, K., Wang, Y., Chen, X., Jain, R., Johnson, M., Batra, D., Parikh, D., Lee, S., and Anderson, P. Nocaps: Novel object captioning at scale. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8948–8957, 2019.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Dai, W., Li, J., Li, D., Tiong, A., Zhao, J., Wang, W., Li, B., Fung, P., and Hoi, S. Instructblip: Towards general-purpose vision-language models with instruction tuning. arxiv 2023. *arXiv preprint arXiv:2305.06500*, 2023.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, X., Luu, A. T., Lin, M., Yan, S., and Zhang, H. How should pre-trained language models be fine-tuned towards adversarial robustness? *Advances in Neural Information Processing Systems*, 34:4356–4369, 2021.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

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

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Gurari, D., Li, Q., Stangl, A. J., Guo, A., Lin, C., Grauman, K., Luo, J., and Bigham, J. P. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3608–3617, 2018.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

He, J., Guo, H., Tang, M., and Wang, J. Continual instruction tuning for large multimodal models. *arXiv preprint arXiv:2311.16206*, 2023.

Hong, Y., Zhen, H., Chen, P., Zheng, S., Du, Y., Chen, Z., and Gan, C. 3d-llm: Injecting the 3d world into large language models. *arXiv preprint arXiv:2307.12981*, 2023.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., and Soudry, D. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pp. 4466–4475. PMLR, 2021.

Hudson, D. A. and Manning, C. D. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6700–6709, 2019.

Korbak, T., Elsahar, H., Kruszewski, G., and Dymetman, M. Controlling conditional language models without catastrophic forgetting. In *International Conference on Machine Learning*, pp. 11499–11528. PMLR, 2022.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Lee, C., Cho, K., and Kang, W. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*, 2019.

Li, B., Zhang, Y., Chen, L., Wang, J., Yang, J., and Liu, Z. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023a.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023b.

Li, Y., Du, Y., Zhou, K., Wang, J., Zhao, W. X., and Wen, J.-R. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023c.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Lin, Y., Tan, L., Lin, H., Zheng, Z., Pi, R., Zhang, J., Diao, S., Wang, H., Zhao, H., Yao, Y., et al. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *arXiv preprint arXiv:2309.06256*, 2023.

Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023a.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023b.

Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023c.

Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Lyu, W., Dong, X., Wong, R., Zheng, S., Abell-Hart, K., Wang, F., and Chen, C. A multimodal transformer: Fusing clinical notes with structured ehr data for interpretable in-hospital mortality prediction. In *AMIA Annual Symposium Proceedings*, volume 2022, pp. 719. American Medical Informatics Association, 2022a.

Lyu, W., Zheng, S., Ma, T., and Chen, C. A study of the attention abnormality in trojaned berts. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4727–4741, 2022b.

Lyu, W., Zheng, S., Ling, H., and Chen, C. Backdoor attacks against transformers with attention enhancement. In *ICLR 2023 Workshop on Backdoor Attacks and Defenses in Machine Learning*, 2023a.

Lyu, W., Zheng, S., Pang, L., Ling, H., and Chen, C. Attention-enhancing backdoor attacks against bert-based models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10672–10690, 2023b.

Lyu, W., Lin, X., Zheng, S., Pang, L., Ling, H., Jha, S., and Chen, C. Task-agnostic detector for insertion-based backdoor attacks. *arXiv preprint arXiv:2403.17155*, 2024.

Marino, K., Rastegari, M., Farhadi, A., and Mottaghi, R. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pp. 3195–3204, 2019.

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., and Van De Weijer, J. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

Meng, Z., Zhang, J., Yang, C., Zhan, Z., Zhao, P., and WAng, Y. Diffclass: Diffusion-based class incremental learning. *arXiv preprint arXiv:2403.05016*, 2024.

Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., and Blankevoort, T. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Panigrahi, A., Saunshi, N., Zhao, H., and Arora, S. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

Schwenk, D., Khandelwal, A., Clark, C., Marino, K., and Mottaghi, R. A-okvqa: A benchmark for visual question answering using world knowledge. In *European Conference on Computer Vision*, pp. 146–162. Springer, 2022.

Singh, A., Natarajan, V., Shah, M., Jiang, Y., Chen, X., Batra, D., Parikh, D., and Rohrbach, M. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8317–8326, 2019.

Sung, Y.-L., Yoon, J., and Bansal, M. Ecoflap: Efficient coarse-to-fine layer-wise pruning for vision-language models. *arXiv preprint arXiv:2310.02998*, 2023.

Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., and Gong, Y. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12183–12192, 2020.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Wang, P., Chen, Q., He, X., and Cheng, J. Towards accurate post-training network quantization via bit-split and stitching. In *International Conference on Machine Learning*, pp. 9847–9856. PMLR, 2020.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022a.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Selvan Dhanasekaran, A., Naik, A., Stap, D., et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv e-prints*, pp. arXiv–2204, 2022b.

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.

Xuhong, L., Grandvalet, Y., and Davoine, F. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pp. 2825–2834. PMLR, 2018.

Yang, Y., Yuan, H., Li, X., Lin, Z., Torr, P., and Tao, D. Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. *arXiv preprint arXiv:2302.03004*, 2023.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2023.

Zhai, Y., Tong, S., Li, X., Cai, M., Qu, Q., Lee, Y. J., and Ma, Y. Investigating the catastrophic forgetting in multimodal large language models. *arXiv preprint arXiv:2309.10313*, 2023.

Zhang, H., Li, X., and Bing, L. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023a.

Zhang, J., Chen, C., Zhuang, W., and Lyu, L. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4782–4793, 2023b.

Zhang, M., Yuan, J., He, Y., Li, W., Chen, Z., and Kuang, K. MAP: Towards balanced generalization of iid and ood through model-agnostic adapters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV*, pp. 11921–11931, 2023c.

Zhang, M., Li, H., Wu, F., and Kuang, K. Metacoco: A new few-shot classification benchmark with spurious correlation. In *International Conference on Learning Representations, ICLR*, 2024.

Zhang, P., Wang, X. D. B., Cao, Y., Xu, C., Ouyang, L., Zhao, Z., Ding, S., Zhang, S., Duan, H., Yan, H., et al. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*, 2023d.

Zhang, R., Han, J., Zhou, A., Hu, X., Yan, S., Lu, P., Li, H., Gao, P., and Qiao, Y. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023e.

Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*, 2020.

Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023a.

Zhu, D., Li, Y., Shao, Y., Hao, J., Wu, F., Kuang, K., Xiao, J., and Wu, C. Generalized universal domain adaptation with generative flow networks. In *ACM International Conference on Multimedia (MM) 2023*, 2023b.

Zhu, D., Li, Y., Yuan, J., Li, Z., Kuang, K., and Wu, C. Universal domain adaptation via compressive attention matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6974–6985, 2023c.

Zhu, D., Li, Y., Zhang, M., Yuan, J., Liu, J., Kuang, K., and Wu, C. Bridging the gap: neural collapse inspired prompt tuning for generalization under class imbalance. *arXiv preprint arXiv:2306.15955*, 2023d.

Zhu, L., Chen, T., Ji, D., Ye, J., and Liu, J. Llafs: When large-language models meet few-shot segmentation. *arXiv preprint arXiv:2311.16926*, 2023e.

Zhu, L., Ji, D., Chen, T., Xu, P., Ye, J., and Liu, J. Ibd: Alleviating hallucinations in large vision-language models via image-biased decoding. *arXiv preprint arXiv:2402.18476*, 2024.

Appendix of Model Tailor: Mitigating Catastrophic Forgetting in MLLMs

In this appendix, we provide the details omitted in the main text, offering additional analyses, proofs, and discussions.

- Appendix A: Additional discussions on related works (cf. section 2 of the main paper).
- Appendix B: Preliminaries and its extension to multi-task scenarios to our method (cf.section 4 of the main paper).
- Appendix C: Detailed proofs of Theorem 4.3 and Theorem 4.5 (cf. section 4 of the main paper).
- Appendix D: Comprehensive experimental details and further results. (cf. section 5 the main paper).

A. More Details about Related Work

Difference with Continue Instruction Tuning. Recently, (He et al., 2023) has applied instruction tuning to the continual learning of MLLMs. Our work differs from this study in three ways: (1) While (He et al., 2023) focus on sequentially tasks and measuring the forgetting of older tasks when learning new tasks, our focus is on the forgetting of the pre-trained MLLM after fine-tuning specific tasks. (2) We do not require measuring task similarity or acquiring original task data, making our method more applicable to real-world scenarios. (3) Our post-training method is more flexible and parameter-efficient, as we only select a small number of parameters, avoiding full-model fine-tuning.

Catastrophic Forgetting in Small Models. This phenomenon is predominantly observed in continuous learning, which revolve around the sequential handling of tasks while quantifying the extent of forgetting of older tasks when new tasks are introduced (Tao et al., 2020; Masana et al., 2022; Zhang et al., 2023c; 2024; Zhu et al., 2023b;c;d; Zhang et al., 2023b; Meng et al., 2024). Current methods primarily focus on full-mdoel fine-tuning, such as the approach by (Xuhong et al., 2018; Ritter et al., 2018; Aljundi et al., 2018; Schwarz et al., 2018), which imposes a penalty on parameter changes for new tasks. Meanwhile, (Yang et al., 2023) draws inspiration from the neural collapse and introduce a distribution-aware loss. However, these strategies are not suitable for large-scale pre-trained models. As research into Attention and other methodologies advances (Lyu et al., 2022a;b; 2023b;a; 2024), the issue of catastrophic forgetting in Large Language Models (LLMs) and Multi-modal LLMs has increasingly caught the attention of researchers.

B. More Details about Method

B.1. Preliminary

Lottery Tickets Hypothesis. The Lottery Ticket Hypothesis (Frankle & Carbin, 2019) proposes an intriguing concept in neural networks, suggesting that within a dense, randomly-initialized network lies a smaller, potentially powerful subnetwork. This subnetwork, when trained in isolation, can achieve comparable or superior test accuracy to the original network, within a similar or reduced number of training iterations.

To elaborate, consider a dense feed-forward neural network denoted as $f(x;\theta)$, where θ represents the network's initial parameters, following a distribution $\theta_0 \sim \mathcal{D}_\theta$. Upon training this network using stochastic gradient descent (SGD) on a given dataset, it attains a minimum validation loss l at a specific iteration j, alongside a test accuracy of a. The hypothesis further explores the scenario of training the network $f(x; m \odot \theta)$, where $m \in \{0,1\}^{|\theta|}$ acts as a binary mask applied to the network's parameters. This setup maintains the initial parameter state as $m \odot \theta_0$. Through SGD optimization on the identical dataset, while keeping m constant, the network is expected to reach a minimum validation loss l' by iteration j', achieving a test accuracy a'. There exists at least one mask m that allows for training within the same or fewer iterations ($j' \leq j$), achieving similar or better accuracy ($a' \geq a$), while significantly reducing the number of parameters utilized ($\|m\|_0 \ll |\theta|$). This hypothesis highlights the existence of optimal subnetworks within larger networks that can perform equally well with a fraction of the parameters.

Optimal Brain Surgeon. The Optimal Brain Surgeon framework (LeCun et al., 1989; Hassibi et al., 1993) focuses on pruning weights from a densely connected neural network with the least increase in loss. It hinges on a meticulous calculation involving the Hessian matrix, $\mathbf{H}$, of the loss function. This matrix encapsulates the second-order partial derivatives of the loss w.r.t. the network's weights, offering deep insights into the loss landscape around the current model parameters.

The decision to prune a particular weight, w_p, is based on minimizing the ratio $\frac{w_p^2}{[\mathbf{H}^{-1}]_{pp}}$, where $\left[\mathbf{H}^{-1}\right]_{pp}$ refers to the p-th diagonal entry of the inverse Hessian matrix. This approach ensures the selected weight for pruning is the one whose removal incurs the minimal increase in the loss, effectively identifying the least critical connections in terms of the model's performance. Moreover, OBS proposes a method to adjust the remaining weights in the network to compensate for the pruned weight. The adjustment for the remaining weights, $\boldsymbol{\delta p}$, is determined by $-\frac{w_p}{[\mathbf{H}^{-1}]_{pp}} \cdot \mathbf{H}^{-1}_{:,p}$, where $\mathbf{H}^{-1}_{:,p}$ is the p-th column of the inverse Hessian. This compensatory update aims to mitigate the loss increase caused by pruning, ensuring the network maintains its performance as much as possible post-pruning.

B.2. Extension to Multi-Task Scenario

Expanding upon the above discussions centered around fine-tuning for a singular target task, our methodology exhibits versatility through its adaptability to multi-task scenarios. Formally, consider m target tasks, denoted as $\mathcal{T} = \{D_1, D_2, ..., D_m\}$, then we can obtain m models fine-tuned on m tasks with parameters $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, ..., \boldsymbol{\Theta}_m\}$. The fusion process in multi-task scenario, previously encapsulated by Equation 3, is formulated as follows :

$$\begin{aligned} \boldsymbol{\Theta}_{\text{fusion}} &= \mathcal{F}(\{\boldsymbol{\Theta}_i\}_{\text{i=1}}^m, \boldsymbol{\Theta}_{\text{pre}}) \\ &= \boldsymbol{M}_{\text{agg}} \odot (\text{Avg}\{\boldsymbol{\Theta}_i\}_{i=1}^m + \boldsymbol{C}_{\text{agg}}) + (I - \boldsymbol{M}_{\text{agg}}) \odot \boldsymbol{\Theta}_{\text{pre}}, \end{aligned} \tag{14}$$

where $\boldsymbol{M}_{\text{agg}}$ denotes the aggregated mask resulting from the union of individual task-specific masks $\boldsymbol{M}_i$, and $\boldsymbol{C}_{\text{agg}}$ signifies the average compensatory values determined for each task, calculated as follows:

$$\boldsymbol{M}_{\text{agg}} = \bigcup_{i=1}^{m} \boldsymbol{M}_i, \quad \boldsymbol{C}_{\text{agg}} = (\sum_{i=1}^{m} \boldsymbol{C}_i)/m. \tag{15}$$

The resultant model embodies a harmonized set of parameter adjustments and knowledge across multiple target tasks. By aggregating the adjustments and selections from various tasks, this method not only underscores the model's enhanced capacity for generalization but also optimizes its performance across a diverse array of objectives.

C. Proof

C.1. Proof of Theorem 4.5

Theorem C.1. *(Theorem 4.5) Given the alteration of a parameter from its fine-tuned state, θ^{ft}_m, to its pre-trained state, θ^{pre}_m, the optimal perturbation $\Delta\boldsymbol{\Theta}$ for each retained fine-tuned parameters is calculated as follows:*

$$\Delta\boldsymbol{\Theta}^* = -\frac{\theta^m_{sft} - \theta^m_{pre}}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}^{-1}_{:,m}, \tag{16}$$

where $\mathbf{H}^{-1}$ denotes the inverse of the Hessian matrix and $\mathbf{H}^{-1}_{:,m}$ signifies the m-th column of $\mathbf{H}^{-1}$.

Proof. Based on Equation 5, denote that $\mathcal{L}_{\mathcal{T}} = \left\|\boldsymbol{\Theta}^{\ell}_{\text{sft}}\mathbf{X}^{\ell}_{\mathcal{T}} - \boldsymbol{\Theta}^{\ell}_{\text{fusion}}\mathbf{X}^{\ell}_{\mathcal{T}}\right\|_2^2$. For clarity in the following proof, we will denote $\boldsymbol{\Theta}^{\ell}_{\text{sft}}$ simply as $\boldsymbol{\Theta}$. We begin with a Taylor expansion of the loss function $\mathcal{L}_{\mathcal{T}}$ around the corresponding optimal parameters $\boldsymbol{\Theta}^*$. This expansion allows us to estimate the change in loss as we adjust the parameters from their optimal values.

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\Theta}) = \mathcal{L}_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right) + \mathcal{L}'_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right) + \frac{\mathcal{L}''_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)}{2!}\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right)^2 + \frac{\mathcal{L}'''_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)}{3!}\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right)^3 + \ldots. \tag{17}$$

The goal is to minimize the impact on loss when a parameter is altered. The following equation represents the change in loss $\Delta\mathcal{L}_{\mathcal{T}}$ when the parameters $\boldsymbol{\Theta}$ deviate from their optimal setting $\boldsymbol{\Theta}^*$.

$$\Delta\mathcal{L}_{\mathcal{T}} = \mathcal{L}_{\mathcal{T}}(\boldsymbol{\Theta}) - \mathcal{L}_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right) = \mathcal{L}'_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right) + \frac{\mathcal{L}''_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)}{2!}\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right)^2 + \frac{\mathcal{L}'''_{\mathcal{T}}\left(\boldsymbol{\Theta}^*\right)}{3!}\left(\boldsymbol{\Theta} - \boldsymbol{\Theta}^*\right)^3 + \ldots. \tag{18}$$

To further refine our approach, we consider only the second-order term of the Taylor expansion, which involves the Hessian matrix $\mathbf{H}$. This term is the primary contributor to the change in loss when small perturbations are made to the parameters.

$$\Delta \mathcal{L}_{\mathcal{T}} = \left(\frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial \mathbf{\Theta}^*} \right)^{\top} \cdot \Delta \mathbf{\Theta} + \frac{1}{2} \Delta \mathbf{\Theta}^{\top} \cdot \mathbf{H} \cdot \Delta \mathbf{\Theta} + O\left(\|\Delta \mathbf{\Theta}\|^3 \right), \tag{19}$$

where $\Delta \mathbf{\Theta} = \mathbf{\Theta} - \mathbf{\Theta}^*$.

In the layer-wise objective as delineated in Equation 5, the constraint mandates that the cumulative absolute difference between the fusion parameters and their pre-trained counterparts, $\mathbf{\Theta}^{\ell}_{\text{pre}}$, must remain below a predefined threshold η. This restriction is crucial for retaining as much of the pre-trained knowledge as possible, thereby mitigating the risks of catastrophic forgetting. To effectively operationalize this constraint at an individual parameter level, we reformulate it into a more explicit condition, as shown in Equation 20:

$$\mathbf{e}_m^{\top} \cdot \Delta \mathbf{\Theta} + \theta^m_{\text{sft}} = \theta^m_{\text{pre}}. \tag{20}$$

This reformulation stipulates that for each fine-tuned parameter θ^m_{sft}, the corresponding adjustment $\Delta \mathbf{\Theta}$ should result in a fusion parameter that equals the pre-trained parameter value θ^m_{pre}. This constraint ensures that the knowledge embedded in each parameter of the pre-trained model is preserved to the maximum extent possible within the defined bounds of η.

To find the optimal perturbation $\Delta \mathbf{\Theta}$ that minimizes the quadratic term in the loss change while satisfying the constraint, we set up the following minimization problem.

$$\min_{\Delta \mathbf{\Theta}} \left(\frac{1}{2} \Delta \mathbf{\Theta}^{\top} \mathbf{H} \Delta \mathbf{\Theta} \right), \quad \text{s.t.} \quad \mathbf{e}_m^{\top} \Delta \mathbf{\Theta} + \theta^m_{\text{sft}} = \theta^m_{\text{pre}}. \tag{21}$$

In order to impose the best choice for the parameter to be removed, we can further consider the following constrained minimization problem.

$$\min_{m} \left\{ \min_{\Delta \mathbf{\Theta}} \left(\frac{1}{2} \Delta \mathbf{\Theta}^{\top} \cdot \mathbf{H} \cdot \Delta \mathbf{\Theta} \right) \middle| \, \mathbf{e}_m^{\top} \cdot \Delta \mathbf{\Theta} + \theta^m_{\text{sft}} = \theta^m_{\text{pre}} \right\}. \tag{22}$$

We use the method of Lagrange multipliers to solve this constrained optimization problem. The Lagrangian L integrates the objective function with the constraint, incorporating the Lagrange multiplier λ.

$$L(\Delta \mathbf{\Theta}) = \frac{1}{2} \Delta \mathbf{\Theta}^{\top} \cdot \mathbf{H} \cdot \Delta \mathbf{\Theta} + \lambda \left(\mathbf{e}_m^{\top} \cdot \Delta \mathbf{\Theta} + \theta^m_{\text{sft}} - \theta^m_{\text{pre}} \right). \tag{23}$$

Taking the derivative of the Lagrangian with respect to $\Delta \mathbf{\Theta}$ and setting it to zero gives us the direction in which the Lagrangian is stationary, which is necessary for finding the optimal point.

$$\frac{\partial L(\Delta \mathbf{\Theta})}{\partial \Delta \mathbf{\Theta}} = \mathbf{X}^{\top} \mathbf{X} \Delta \mathbf{\Theta} + \lambda \mathbf{e}_m^{\top} = 0. \tag{24}$$

Furthermore, the derivative of the Lagrangian with respect to the Lagrange multiplier λ must also equal zero to satisfy the constraint. This derivative yields:

$$\frac{\partial L(\Delta \mathbf{\Theta})}{\partial \lambda} = \mathbf{e}_m^{\top} \cdot \Delta \mathbf{\Theta} + \theta^m_{\text{sft}} - \theta^m_{\text{pre}} = 0. \tag{25}$$

By solving these equations, we obtain the expression for $\Delta \boldsymbol{\theta}$, which minimizes the objective function while satisfying the constraint that the final weights should align with the pre-update weights to avoid catastrophic forgetting:

$$\Delta \mathbf{\Theta} = -(\mathbf{X}^{\top} \mathbf{X})^{-1} \lambda \mathbf{e}_m^{\top}. \tag{26}$$

The Lagrange multiplier λ is then computed based on the imposed constraint and the inverse of the Hessian matrix, leading to the final update rule for the weight perturbation:

$$\lambda = (\theta^m_{\text{sft}} - \theta^m_{\text{pre}}) \cdot \frac{1}{(\mathbf{H}^{-1})_{mm}}. \tag{27}$$

The corresponding optimal perturbation so obtained is as follows when altering the m-th parameter θ_m:

$$\Delta\mathbf{\Theta}_m^* = -\frac{\theta_{\text{sft}}^m - \theta_{\text{pre}}^m}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}^{-1} \cdot \mathbf{e}^m = -\frac{\theta_{\text{sft}}^m - \theta_{\text{pre}}^m}{[\mathbf{H}^{-1}]_{mm}} \cdot \mathbf{H}_{:,m}^{-1}. \tag{28}$$

The proof of Theorem 4.5 is finished.

□

C.2. Proof of Theorem 4.3

Theorem C.2. *(Theorem 4.3) Consider a layer ℓ within an MLLM $\mathcal{M}$, and let θ_m represent a parameter at index m. Altering θ_m from its fine-tuned state θ_{sft}^m to its pre-trained state θ_{pre}^m, induces a increase $\Delta\mathcal{L}_{\mathcal{T}}$ in the model's loss for a task $\mathcal{T}$, quantified as:*

$$\Delta\mathcal{L}_{\mathcal{T}} = \frac{(\theta_{sft}^m - \theta_{pre}^m)^2}{[\mathbf{H}^{-1}]_{mm}}, \tag{29}$$

where $\mathbf{H}^{-1}$ denotes the inverse of the Hessian matrix and $\left[\mathbf{H}^{-1}\right]_{mm}$ is its m-th diagonal element. The Hessian matrix $\mathbf{H} = \nabla^2_{\theta_{sft}^m}\mathcal{L}_{\mathcal{T}}$ takes into account the local geometry of the loss at a given point θ_{sft}^m assuming $\mathcal{L}_{\mathcal{T}}$ is second-order differentiable.

Proof. The change in loss due to the optimal perturbation obtained in Equation 28 that aligns with the pre-trained model's parameters is quantified, giving us a measure of how the loss increases as we modify the network.

$$\begin{aligned} \Delta\mathcal{L}_{\mathcal{T}} &= \left(\frac{\partial\mathcal{L}_{\mathcal{T}}}{\partial\mathbf{\Theta}}\right)^{\top} \cdot \Delta\mathbf{\Theta} + \frac{1}{2}\Delta\mathbf{\Theta}^{\top} \cdot \mathbf{H} \cdot \Delta\mathbf{\Theta} + O\left(\|\Delta\mathbf{\Theta}\|^3\right) \\ &\approx \frac{1}{2}\Delta\mathbf{\Theta}^{\top} \cdot \mathbf{H} \cdot \Delta\mathbf{\Theta} \\ &\approx \frac{(\theta_{\text{sft}}^m - \theta_{\text{pre}}^m)^2}{2\,[\mathbf{H}^{-1}]_{mm}}. \end{aligned} \tag{30}$$

Returning to the challenge presented in Equation 22, the optimal selection of the parameter index m involves identifying and removing the parameter θ_m that results in the smallest increase in loss. This increase in loss is quantified as the sensitivity-based statistic s_ε, as defined in Equation 9:

$$s_\varepsilon = \frac{(\theta_{\text{sft}}^m - \theta_{\text{pre}}^m)^2}{2\,[\mathbf{H}^{-1}]_{mm}}. \tag{31}$$

We calculate this statistic for all parameters, then rank them in descending order based on the magnitude of s_ε.

The proof of Theorem 4.3 is finished.

□

D. More Details about Experiments

D.1. Implementation Details

Standard Fine-tuning. In the case of InstructBLIP, our fine-tuning procedure adhered to the official codebase [1] guidelines with a batch size of 12 for each task, a maximum of 5 epochs, and a learning rate of 1e-5. Similarly, for LLaVA, we followed the official fine-tuning protocols [2], setting the maximum epoch to 1. The initial learning rate was configured at 2e-4 for fine-tuning on Flickr30k and 1e-4 for GQA, with a batch size of 64. All experiments were conducted on 8 NVIDIA V100 GPU with 32GB of memory.

[1] https://github.com/salesforce/LAVIS

[2] https://github.com/haotian-liu/LLaVA

Computation Complexity of Model Tailor. In the implementation of Model Tailor, we have employed an efficient computational technique called SparseGPT (Frantar & Alistarh, 2023) to calculate the inverse Hessian matrix, a critical component in the OBS calculation. The computational complexity of calculating the inverse Hessian as described in SparseGPT involves three main components: (1) Initial Hessian Computation: The initial Hessian matrix is computed with a time complexity of $\mathcal{O}(nd_{\text{col}}^2)$, where n is the number of input samples and d_{col} is the dimensionality of the matrix column. By choosing n to be a small multiple of d_{col}, we achieve stable results while maintaining efficiency. (2) Hessian Inversion Iteration: Iterating through the inverse Hessian requires $\mathcal{O}(d_{\text{col}}^3)$ time complexity, which is manageable even for larger models. (3) Reconstruction Process: The pruning or reconstruction process, based on the inverse Hessian, is bounded by the complexity of $\mathcal{O}(d_{\text{col}}^3 + d_{\text{row}}^2 d_{\text{col}})$, where d_row represents the dimensionality of the matrix row. This ensures that even for models with extensive parameters, the process remains computationally feasible. In summary, the overall computational complexity aligns with $\mathcal{O}(d_{\text{hidden}}^3)$, considering a Transformer model's hidden dimensionality d_{hidden}. This demonstrates a substantial efficiency improvement over exact reconstruction methods, confirming that our approach is computationally practical even for very large models.

Combination with LoRA. We adhered to the standard LoRA fine-tuning paradigm provided by the official code repository[3] of LLaVA. For fine-tuning on Flickr30k, we set the initial learning rate at 2e-4, and for GQA, the initial learning rate was set at 1e-4. All other hyperparameter settings remained consistent with the original setup of LLaVA. After the fine-tuning process on LLaVA-1.5 using LoRA, we integrate the incremental updates from LoRA throughout the entire fine-tuned model. This means that our Model Tailor method is applied across the full model, encompassing both the MLP mapper and the entirety of the LLM, which totals approximately 6.7 billion parameters. Therefore, to manage the volume of selected parameters effectively and ensure it remains substantial but not excessive, we employed a sparsity level of 5%, which corresponds to approximately 335 million parameters. This approach ensures that Model Tailor works in conjunction with the enhancements provided by LoRA, facilitating a comprehensive adaptation across the model's expansive parameter space.

D.2. Evaluation Metrics

To rigorously assess the effectiveness of our method in mitigating catastrophic forgetting on MLLMs, we utilize two key evaluation metrics: the Average Performance and the H-score. The H-score, in particular, is a novel metric designed to provide a balanced evaluation between original and target tasks. It is defined as the harmonic mean of the average performance on the original tasks, $\text{Avg}(P_{\text{origin}})$, and the average performance on the target tasks, $\text{Avg}(P_{\text{target}})$. The formula for the H-score is as follows:

$$P_H = \frac{2 \times \text{Avg}(P_{\text{origin}}) \times \text{Avg}(P_{target})}{\text{Avg}(P_{\text{origin}}) + \text{Avg}(P_{\text{target}})}. \tag{32}$$

The rationale behind the introduction of the H-score is to circumvent the potential overemphasis on the performance of original tasks, particularly as their number increases.

D.3. More Results

Quantitative Results on Multi-Task Setting. Table 5 and Table 6 present the quantitative outcomes of Model Tailor within a multi-task context. The data delineates a clear enhancement in performance on pre-trained tasks when integrating masks and compensatory values from two distinct tasks. This amalgamation of knowledge fosters a generalized proficiency, underscoring the method's improved adaptability and robustness across varied tasks.

Quantitative results of Model Tailor combined with LoRA. Table 7 details the quantitative performance of Model Tailor when amalgamated with LoRA on the LLaVA-1.5 model. The results demonstrate remarkable performance boosts, regardless of whether the model was fine-tuned on Flickr30k or OKVQA. This indicates the potentiated benefits of combining Model Tailor's approach with LoRA's fine-tuning capabilities.

Visualization of decorator. Figure 6 visualizes the parametric shifts pre and post compensation. The vertical axis quantifies the absolute difference in cumulative sums between each parameter and its corresponding pre-trained value. In essence, taller bars within the histogram signify a greater deviation towards the fine-tuned model's parameters. The illustration evidences the decorator's role in nudging the model parameters closer to the fine-tuned state, thereby compensating for the performance impact on the target task due to the removal of certain parameters (while maintaining pre-trained values).

[3] https://github.com/haotian-liu/LLaVA

Table 5: **H-score and average performance on InstructBLIP (Vicuna-7B) with a sparsity ratio $\rho = 10\%$ under multi-task setting.** The optimal and sub-optimal results are denoted respectively by boldface and underlining for emphasis.

Method	Pretrain tasks							Target task			
	COCO	NoCaps-in	NoCaps-near	NoCaps-out	OKVQA	AOKVQA	VQAv2	GQA	Flickr30k	Avg	Hscore
Flickr30k-Fintuned	139.8	115.8	124.3	123.4	**57.63**	57.84	76.19	48.58	**94.4**	93.10	**83.12**
GQA-Fintuned	141.8	**116.5**	121.8	120.7	56.15	58.33	76.28	**53.98**	82.7	92.03	80.79
Hybrid-Flickr30k-GQA	**141.9**	116.3	**125.4**	**125.5**	57.09	**58.64**	**76.69**	52.00	90.0	**93.72**	**83.12**

Table 6: **H-score and average performance on LLaVA-1.5 (Vicuna-7B) with a sparsity ratio $\rho = 10\%$ under multi-task setting.** The optimal and sub-optimal results are denoted respectively by boldface and underlining for emphasis.

Method	Pretrain tasks								Target task			
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	OKVQA	Flickr30k	Avg	Hscore
Flockr30k-fine-tuned	73.21	52.49	42.28	67.15	43.89	82.88	63.40	56.15	7.05	**75.4**	62.72	64.80
OKVQA-fine-tuned	76.25	60.39	**46.49**	69.51	54.88	85.44	63.32	54.21	**38.1**	0.01	65.18	54.01
Hybrid-Flickr30k-OKVQA	**76.88**	**61.29**	45.88	**72.41**	53.76	**85.96**	**63.77**	**57.72**	30.2	59.7	**65.71**	**69.52**

Table 7: **Combination with LoRA on LLaVa1.5 (Vicuna-7b) with a sparsity ratio $\rho = 5\%$.** The optimal and sub-optimal results are denoted respectively by boldface and underlining for emphasis.

Method	Pre-trained tasks								Target task		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	Flickr30k	Avg	Hscore
LoRA	76.35	58.64	50.25	65.13	55.18	**84.91**	64.43	53.09	62.5	63.39	62.99
Ours + LoRA	**78.59**	**61.39**	**53.2**	**69.61**	**58.01**	84.22	**65.03**	**57.82**	81.4	**67.70**	**72.89**

Method	Pre-trained tasks								Target task		
	VQAv2	GQA	VizWiz	SQA	TextVQA	POPE	MM-Bench	MM-Bench-CN	OKVQA	Avg	Hscore
LoRA	71.21	52.87	37.09	28.15	54.36	67.81	35.91	29.47	59.25	48.45	52.48
Ours + LoRA	**72.44**	**54.97**	**40.09**	**57.56**	**55.73**	67.73	**56.01**	**46.39**	**59.82**	**56.75**	**58.04**

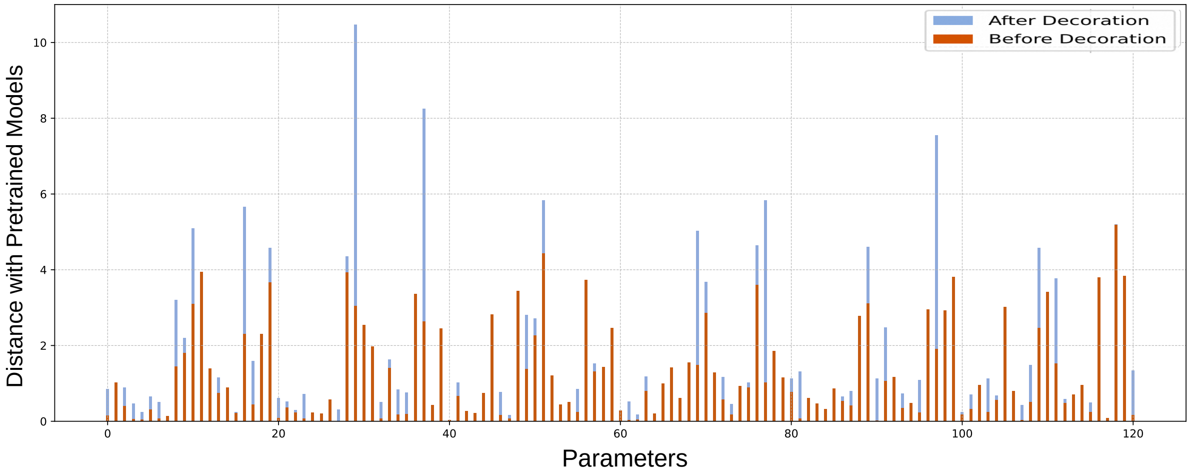


Figure 6: **Visualization of Decorator's Effectiveness.** This bar graph contrasts the parameter distances from their pre-trained counterparts before and after the application of the patch decorator. The decorator moves parameters closer to their fine-tuned states, demonstrates the decorator's capacity to counterbalance the exclusion of certain parameters.