MTLoRA: A Low-Rank Adaptation Approach for Efficient Multi-Task Learning

Ahmed Agiza*
Brown University
Providence, RI
ahmed_agiza@brown.edu

Marina Neseem*
Brown University
Providence, RI
marina_neseem@brown.edu

Sherief Reda
Brown University
Providence, RI
sherief_reda@brown.edu

Abstract

Adapting models pre-trained on large-scale datasets to a variety of downstream tasks is a common strategy in deep learning. Consequently, parameter-efficient fine-tuning methods have emerged as a promising way to adapt pre-trained models to different tasks while training only a minimal number of parameters. While most of these methods are designed for single-task adaptation, parameter-efficient training in Multi-Task Learning (MTL) architectures is still unexplored. In this paper, we introduce MTLoRA, a novel framework for parameter-efficient training of MTL models. MTLoRA employs Task-Agnostic and Task-Specific Low-Rank Adaptation modules, which effectively disentangle the parameter space in MTL fine-tuning, thereby enabling the model to adeptly handle both task specialization and interaction within MTL contexts. We applied MTLoRA to hierarchical-transformer-based MTL architectures, adapting them to multiple downstream dense prediction tasks. Our extensive experiments on the PASCAL dataset show that MTLoRA achieves higher accuracy on downstream tasks compared to fully fine-tuning the MTL model while reducing the number of trainable parameters by $3.6\times$*. Furthermore, MTLoRA establishes a Pareto-optimal trade-off between the number of trainable parameters and the accuracy of the downstream tasks, outperforming current state-of-the-art parameter-efficient training methods in both accuracy and efficiency. Our code is publicly available.*[1]

1. Introduction

General-purpose vision and language models, particularly those trained on large-scale datasets, show remarkable adaptability to a wide range of downstream tasks [22, 30]. However, individually fine-tuning all parameters of these models for every downstream task poses significant efficiency challenges. This approach becomes increasingly inefficient as the number of tasks grows, especially in environments constrained by computational resources.

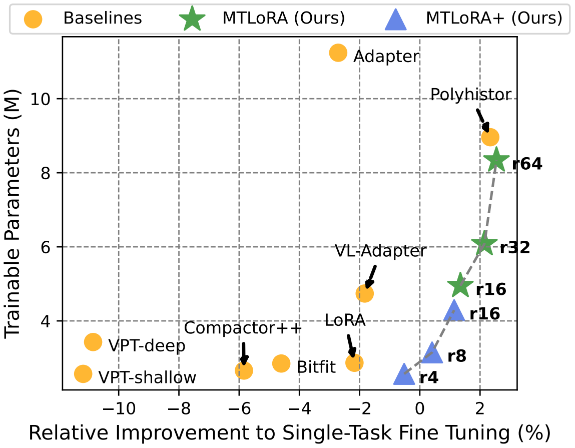


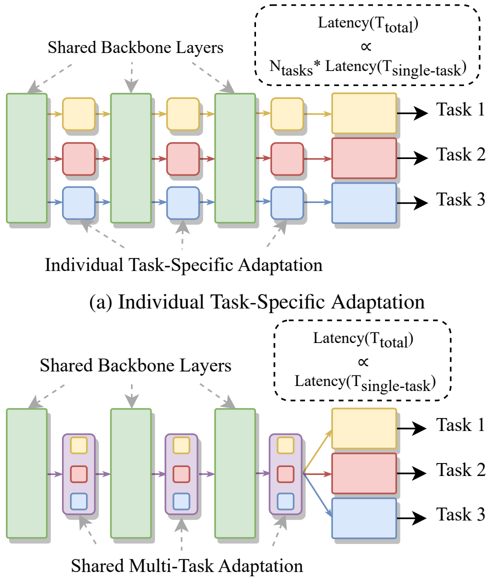
Figure 1. *MTLoRA* versus state-of-the-art parameter-efficient training approaches using Swin-Tiny vision transformer as a backbone. **r** represents the different ranks for the low-rank decomposition modules inside MTLoRA.

Therefore, there is a need to develop resource-efficient fine-tuning techniques [12, 13, 21, 23]. These methods aim to optimize training efficiency by limiting the number of trainable parameters, all while attempting to preserve or enhance task-specific fine-tuning. Most existing parameter-efficient adaptation methods are primarily tailored for single-task adaptation, and they may lose their effectiveness when applied to multi-task learning (MTL) scenarios. This is attributed to the inherent complexity of MTL, where the goal is to optimize the performance of a single model across a spectrum of tasks, introducing an additional layer of complexity. Moreover, focusing solely on individual task adaptation overlooks the potential benefits of cross-task knowledge sharing. Such knowledge sharing in an MTL context can significantly enhance the performance of each task [4, 37].

To realize multi-task adaptation using existing methods [21, 23], individual modules specific to the different tasks have to be added and adapted to one downstream task at a time as shown in Figure 2a. This approach enables cus-

*The first two authors contributed equally to this work.

[1] https://github.com/scale-lab/MTLoRA.git


Shared Backbone Layers
Latency(T$_{total}$)
∝
N$_{tasks}$* Latency(T$_{single-task}$)
Task 1
Task 2
Task 3
Individual Task-Specific Adaptation
Shared Backbone Layers
Latency(T$_{total}$)
∝
Latency(T$_{single-task}$)
Task 1
Task 2
Task 3
Shared Multi-Task Adaptation

(a) Individual Task-Specific Adaptation

(b) Shared Multi-Task Adaptation (Ours)

Figure 2. *(a) Individual Task Adaptation* results in parallel execution paths for each task, resulting in inference and training time that scales linearly with the number of tasks. On the other hand, (b) *Shared Multi-Task Adaptation* maintains inference and training time close to the single task model since only the decoders are executed separately.

tomization and improvement for each task's unique needs, especially useful when tasks have unique characteristics or require specialized knowledge [25, 27]. However, this strategy incurs a significant drawback in terms of efficiency during both training and inference. Due to the task-specific nature of the fine-tuning, the model must be trained and inferred separately for each task. This results in a proportional increase in computational cost and time with the number of tasks. For instance, adapting a model to five distinct tasks would require five separate training passes. Similarly, during inference, the backbone needs to be executed five times, once for each task, leading to a linear escalation in inference and training duration as the task count increases.

Our research diverges from conventional parameter-efficient adaptation methods by concentrating on parameter-efficient training specifically for multi-task learning (MTL) architectures. In MTL models, a single shared backbone is trained to simultaneously extract feature representations for various downstream tasks [5, 19, 33]. MTL offers significant efficiency advantages since the shared backbone is executed only once, as shown in Figure 2b, leading to more resource-efficient training and inference processes where latency does not increase linearly with the number of tasks. Despite these advantages, the aspect of parameter-efficient training in MTL architectures remains largely unexplored.

The primary challenge in fine-tuning MTL models efficiently lies in addressing task conflicts during fine-tuning. These conflicts arise when different tasks have competing demands or induce divergent updates in the model. Hence, the focal point for many MTL architectures [14, 18] is to balance these conflicting updates. Consequently, a pivotal question emerges: *How can we efficiently adapt a single shared backbone to serve multiple tasks without sacrificing the individual performance of each task?*

In pursuit of this objective, we introduce *MTLoRA* -- a novel framework designed for parameter-efficient fine-tuning of MTL models. *MTLoRA* addresses the challenges of fine-tuning a shared backbone to effectively serve multiple downstream tasks, particularly under the constraints of conflicting task requirements. This is accomplished through a strategic combination of *Task-Agnostic* and *Task-Specific* low-rank decomposition modules. By fine-tuning these modules, *MTLoRA* successfully untangles the parameter space involved in MTL fine-tuning, enabling the model to balance between learning shared features and those specific to individual tasks. Remarkably, *MTLoRA* demonstrates superior accuracy in downstream tasks compared to fully fine-tuning the entire MTL model while requiring the training of significantly fewer parameters. This enhanced performance is attributed to *MTLoRA*'s ability to facilitate positive knowledge sharing during fine-tuning, thereby improving the effectiveness of learning each downstream task. **Our contributions** can be summarized as follows:

- To the best of our knowledge, *MTLoRA* is the first to address the problem of parameter-efficient training of multi-task learning models. *MTLoRA* effectively balances between learning both shared and task-specific features during parameter-efficient fine-tuning.
- We design novel *Task-Agnostic* and *Task-Specific* low-rank adaptation modules leveraging them to adapt a shared vision-transformer backbone to multiple downstream dense prediction tasks.
- We observe that adding low-rank adaptation to the patch-merging layers in vision transformers, a practice not previously explored, significantly improves the accuracy-efficiency trade-off during fine-tuning MTL models. We highlight that observation by introducing *MTLoRA+*.
- We apply *MTLoRA* and *MTLoRA+* to a hierarchical-transformer-based MTL architecture. *MTLoRA* demonstrates superior accuracy in downstream tasks compared to fully fine-tuning the entire MTL model while training significantly less number of parameters. In addition, *MTLoRA* dominates state-of-the-art parameter-efficient training approaches, as shown in Figure 1.

The rest of the paper is organized as follows. We review the related work in Section 2. Then, we introduce our MTLoRA framework in Section 3. Next, we show the setup and evaluation of MTLoRA in Section 4. Finally, we conclude in Section 5.

2. Related Work

Multi-task Learning: Multi-task learning is commonly used to learn various related tasks simultaneously [24–26]. The typical design of a multi-task architecture includes an encoder to distill feature representations from input frames and a set of task-specific decoders for generating predictions unique to each downstream task [37]. An important aspect to consider within these multi-task architectures is the mechanism of information sharing. The two main strategies are soft sharing and hard sharing. Soft parameter sharing involves each task having its own set of backbone parameters, with the primary objective of facilitating cross-task information exchange. On the other hand, hard parameter sharing employs a shared set of parameters within the backbone, with each task employing independent decoders for output generation [1, 17, 28]. Further classification of these architectures takes into account the stage at which task interactions occur - leading to the categorization into encoder-focused and decoder-focused frameworks [33]. Encoder-focused architectures centralize information exchange within the encoder stage [20, 31], whereas, in decoder-focused architectures, tasks exchange information during the decoding stage. Notably, some models adopt a more integrative approach, allowing for cross-task information sharing to occur at encoder and decoder stages [25].

Parameter-Efficient Training for Single-Task Models: Parameter-efficient training (PEFT) has become increasingly important, especially when dealing with large-scale pre-trained models [9, 12, 13, 36] since traditional fine-tuning methods, which involve adjusting a significant portion of a model's parameters for specific tasks, can be resource-intensive. Two common techniques in this domain are adapters [9, 36] and Low-Rank Adaptation (LoRA) [7, 12]. *Adapters* are lightweight modules inserted between the layers of a pre-trained model, which allows for targeted modifications to the model's behavior without altering the original pre-trained weights. This approach is beneficial as it reduces the number of parameters that need to be fine-tuned, thus lowering the computational burden. Adapters have shown effectiveness in various tasks, providing a flexible and efficient way to adapt large models to specific tasks or datasets. However, one limitation of adapters is the additional parameters they introduce, which can lead to increased computational requirements during inference. On the other hand, *LoRA* offers a different approach to PEFT. LoRA involves modifying the weight matrices of a pre-trained model using low-rank decomposition. This method allows for fine-tuning the model's behavior while maintaining the original structure and size of the weight matrices. The key advantage of LoRA is that it does not introduce additional parameters during the model's runtime. Instead, it updates the pre-existing weights to enhance the model's performance on new tasks with minimal increase in computational requirements. LoRA has been successfully applied in various fields, including NLP [2, 3, 7, 12] and computer vision [11], demonstrating its versatility and effectiveness. However, these methods, while efficient, only focus on single-task models.

Parameter-Efficient Training for Multi-Task Models: In a multi-task setting, PEFT is more challenging as the model must cater to the needs of multiple tasks simultaneously, often leading to increased complexity and potential for task interference. Consequently, some recent studies have proposed new solutions to extend the benefits of PEFT for multi-task adaptation. One such approach is the Hypernetworks [23], which uses shared networks to generate adapter parameters for all layers conditioned on the task, thus allowing for the sharing of information across different tasks while enabling task-specific adaptation through task-specific adapters. Building on top of it, Polyhistor [21] explores PEFT in the domain of dense vision tasks, specifically on hierarchical vision transformers. Polyhistor proposes two ideas: decomposing hypernetworks into low-rank matrices and using custom kernels to scale fine-tuning parameters to the different transformer blocks. However, these two approaches rely on separate execution of the model for each task to apply its adapter, which does not benefit from the MTL's potential for efficient training or inference.

3. Methodology

Problem Setting: Given a general-purpose transformer-based backbone pre-trained on large-scale image datasets (e.g., ImageNet [6]), our goal is to efficiently adapt it to several downstream tasks in a Multi-Task Learning (MTL) architecture setting. We are considering the common MTL architecture with one shared encoder and multiple task-specific decoders, as shown in Figure 2b. Following the existing works in parameter-efficient training, the criteria of parameter-efficient MTL training include the accuracy of downstream tasks and the number of training parameters.

Method Overview: Our approach for efficiently adapting MTL models to various downstream tasks consists of two novel aspects: (1) efficiently share homogeneous information across tasks via a pool of task-agnostic and task-specific low-rank matrices and (2) efficiently allow multi-scale task-specific feature sharing between the shared-encoder and task-specific decoders of the MTL architecture.

This section is organized as follows. We start with an overview of the used MTL architecture in Subsection 3.1. Then, we propose our parameter-efficient task-specific adaptation method in Subsection 3.2. In Subsection 3.3, we propose our multi-scale task-specific efficient feature-sharing method. Finally, we explore the effect of fine-tuning the non-attention modules in Subsection 3.4.

3.1. MTL Architecture Overview

We first establish a Multi-Task Learning (MTL) framework designed for experimentation with parameter-efficient adaptation in MTL contexts. Aligning with established MTL methodologies [31, 34], our model comprises three main components: a shared hierarchical encoder, task-specific inter-scale fusion modules, and a pool of task-specific decoders. We adopt an off-the-shelf hierarchical vision transformer as the shared encoder [22], which extracts visual features from input frames for all downstream tasks. The hierarchical structure of the encoder allows for capturing visual features at various scales, providing a comprehensive representation of the input data. The extracted multi-scale visual features are then fused and processed by various task-specific decoders to execute the downstream tasks. Our MTL framework is designed to accommodate different vision transformer and decoder architectures, which makes our parameter-efficient adaptation approach suitable for a wide range of MTL architectures.

To effectively adapt the MTL architecture to various downstream tasks, we draw inspiration from the low-rank adaptation (LoRA) technique commonly employed in Language Models [12], traditionally used for single-task adaptation. Our primary inquiry is: *'How does fine-tuning low-rank matrices perform when optimizing for multiple visual downstream tasks?'*. Previous studies in low-rank adaptation mainly aim to identify a unique set of low-rank matrices for adapting the encoder to an individual downstream task [15, 23, 29], as illustrated in Figure 2a. This approach requires running the entire model separately for each of the tasks, which is inefficient for real-time applications. In contrast, our research seeks to develop a single set of low-rank adaptation matrices that are applicable across multiple downstream tasks, as depicted in Figure 2b. This methodology enables a single execution of the backbone for all tasks. Multi-task learning often presents challenges, such as the "conflicting gradients problem" [14]. This issue becomes more pronounced in low-rank adaptation due to the limited number of trainable parameters.

3.2. Low-Rank Adaptation for MTL Architectures

Low-rank decomposition modules are increasingly used to adapt pre-trained models for various tasks [12]. These modules, incorporated into layers that involve matrix multiplication, are notably used in the attention layers of transformer-based models. The function of these modules can be mathematically described as follows:

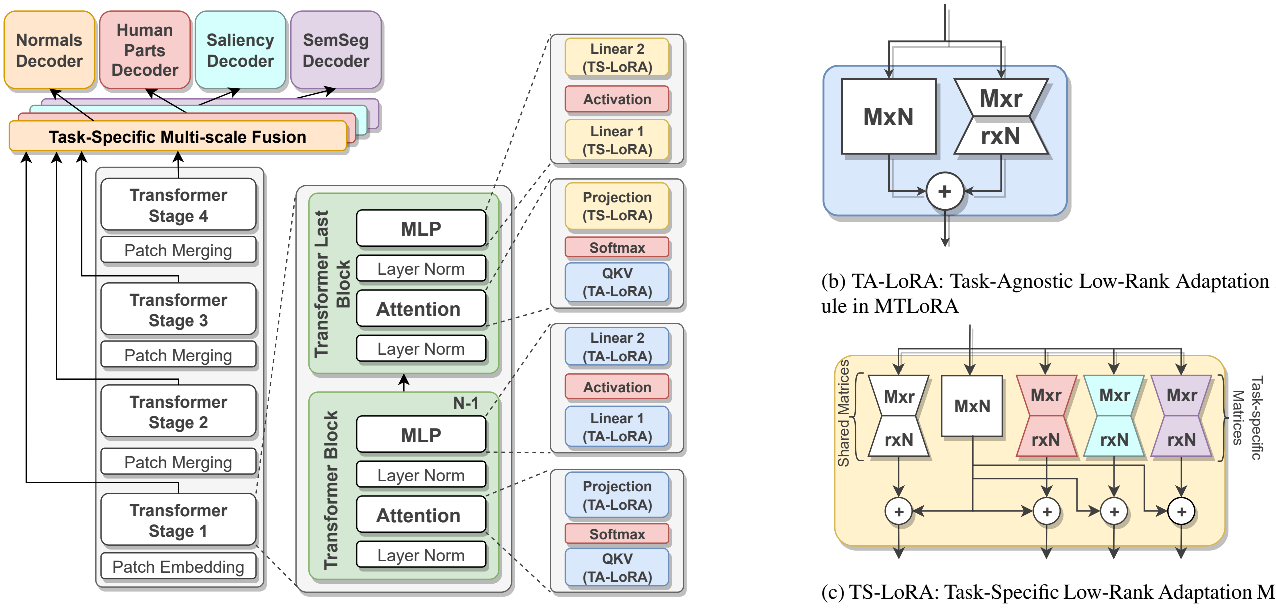
$$Output_{Layer_i} = W_i x + b_i + \alpha B_i A_i x \qquad (1)$$

Here, W_i and b_i are the original weights and biases of the layer. A and B are the rank decomposition matrices, and x is the input to the layer i. α is the adaptation scale which controls the deviation of the tuned model from the original model. During training, only parameters of A, B, and potentially b_i are trained, significantly reducing the memory footprint and leading to faster training. During inference, the A and B matrices can be merged into W_i, ensuring that the introduction of low-rank decomposition does not add any extra latency to the inference process. For Hierarchical Vision transformers, several locations within the architecture are identified as suitable for the application of low-rank matrices, enhancing task adaptability as shown in Figure 3a.

- **QKV Computation in Attention Layers**: The Query, Key, Value (QKV) computation, the main components of the attention mechanism, represents a prime candidate for low-rank adaptation. Fine-tuning this computation allows for modifications to the attention mechanism, making it more suited for specific downstream tasks, which can improve the model's ability to process visual inputs in a task-specific manner.
- **Projection Layer**: The projection layer in transformers is responsible for projecting the attention layer's output back to the original feature space. Fine-tuning the projection layer allows the attention output to be projected into the task's feature space, yielding better performance on downstream tasks.
- **Feed Forward Layers in the MLP block**: These layers, consisting of two dense layers (FC1 and FC2) with nonlinear activation in-between, transform the attention output into the final feature representation. Fine-tuning these layers dictates the model's capacity to effectively generate task-specific final feature representations to be processed by subsequent stages or task-specific decoders.

Adopting low-rank decomposition modules in these layers provides controllable knobs to trade-off fine-tuning efficiency (i.e., Number of trained parameters) and adaptation quality (i.e., Performance on the downstream tasks). We consider two variants of low-rank decomposition modules as shown in Figure 3: (1) *Task-Agnostic Low-Rank Adaptation* module to capture shared features among the various tasks, and (2) *Task-Specific Low-Rank Adaptation* module that can learn task-specific features.

Task-Agnostic Low-Rank Adaptation (*TA-LoRA*): In our framework, we utilize task-agnostic low-rank adaptation modules, which employ low-rank decomposition to adjust the corresponding weights, as detailed in Equation 1. The *TA-LoRA* modules are designed to identify and leverage shared features across multiple downstream tasks, thereby facilitating knowledge sharing. We have integrated *TA-LoRA* modules into the transformer blocks of the Hierarchical Vision Transformer backbone, with the exception of the final block in each stage, as illustrated in Figure 3a. Specifically, these *TA-LoRA* modules are applied to adapt key computational layers within the transformer blocks, namely the QKV Layer, the Projection Layer, and the MLP block. The



(a) Low-Rank Adaptation matrices are applied to all the blocks of the shared backbone.

(b) TA-LoRA: Task-Agnostic Low-Rank Adaptation Module in MTLoRA

(c) TS-LoRA: Task-Specific Low-Rank Adaptation Module in MTLoRA

Figure 3. *MTLoRA* framework overview. Task-Agnostic LoRA modules (*TA-LoRA*) are placed at each transformer block, excluding the last ones in each stage where our Task-Specific LoRA (*TS-LoRA*) modules are placed to capture task-specific fine-tuning at different scales.

inclusion of *TA-LoRA* modules aims to promote a balanced and synergistic learning process. This approach ensures fine-tuning that is unbiased towards any specific task, preventing overfitting. Additionally, to address the challenge of conflicting gradients in MTLoRA, the final block of each stage incorporates our novel *Task-Specific Low-Rank Adaptation* modules, which aim to capture task-specific features as explained in the following paragraph.

Task-Specific Low-Rank Adaptation (*TS-LoRA*): One of the main challenges in multi-task low-rank adaptation is to disentangle the feature learning space in order to solve the conflicts between the various downstream tasks. To achieve that, we propose our novel *TS-LoRA* modules. *TS-LoRA* incorporates separate task-specific low-rank matrices in addition to the shared low-rank matrices as shown in Figure 3c. These modules are designed to operate in two distinct modes. First, when a *TS-LoRA* module follows a layer with a *TA-LoRA* module (for instance, in the projection layer), it processes the shared input to derive task-specific representations. Conversely, in scenarios where a layer with a *TS-LoRA* module succeeds another with a similar module (as observed in MLP feed-forward layers), it processes task-specific inputs to produce corresponding task-specific outputs as follows:

$$Output_{layer_i/task_j} = W_i x + b_i + \alpha_i B_{task_j} A_{task_j} x \quad (2)$$

Here, $Output_{layer_i/task_j}$ represents the $task_j$'s specialized output at layer i. W_i and b_i are the original weights and biases of the layer. x is input to $layer_i$. B_{task_j} and A_{task_j} are the *TS-LoRA* matrices for task j. These modules fine-tune the model according to the specific needs of each task. The outputs of the *TS-LoRA* modules are directed toward the corresponding task-specific fusion modules and decoders, as shown in Figure 3. This allows the encoder to generate task-specific feature representations at various scales. Since these *TS-LoRA* matrices are only connected to their corresponding task-specific decoders in the computation graph, the backward propagation only updates those matrices according to their corresponding task's loss.

In MTLoRA, the usage of both *TA-LoRA* and *TS-LoRA* modules is key to achieving an optimal balance between generalization and specialization within the MTL model. The *TA-LoRA* modules are designed to capture generalized information throughout the model, ensuring that a fundamental level of generality is maintained across various tasks. In contrast, the *TS-LoRA* modules are used to encapsulate unique updates that are tailored to each specific task. This dual-module approach ensures that while the model efficiently processes shared features relevant across multiple tasks, it also possesses the capacity to cater to the specific demands of individual tasks.

3.3. Multi-Scale Task-Specific Feature Sharing in Encoder-Decoder MTL Architecture

Multi-scale feature propagation within the encoder-decoder architecture has been shown to enhance performance in vision tasks [31, 38] where the input data is captured at various scales, providing various levels of abstractions. Typically, a hierarchical vision transformer processes input through multiple stages, with each stage generating features at a different scale. Merging features from these different scales results in a more comprehensive feature representation. In conventional setups, features at various scales are often fused together to create a unified, shared multi-scale feature set applicable to all tasks. However, our *TS-LoRA* module allows for a unique specialization at each scale for every task since it generates task-specific features at the end of every transformer stage as shown in Figure 3a. This enables the creation of task-specific multi-scale features which pushes the model to fine-tune the features at each scale according to the requirements of each task. Our learnable task-specific multi-scale fusion layers use a residual blocks-based architecture to combine the features at different scales (i.e., receptive fields) in an informative way for every downstream task.

3.4. Fine-tuning Non-Attention Modules

Several studies in the domain of parameter-efficient training have highlighted the benefits of unfreezing some low training-cost modules [9], such as layer normalization, which can positively impact the model's performance without significantly increasing the number of trainable parameters. Hence, we explore the effect of unfreezing different modules within MTLoRA. In addition to training the shared *TA-LoRA* and the task-specific *TS-LoRA* modules, we unfreeze the patch embedding layer, the patch merging layer, the layer normalization, and the position bias in the attention layer. We provide insights about the effect of freezing each of those layers on the accuracy-efficiency trade-off in Subsection 4.4. Additionally, we explore adding low-rank decomposition modules to the *patch merging* module instead of completely unfreezing it. This allows for further reduction in training parameters; we denote this lighter version as *MTLoRA+* referring to those extra low-rank decomposition modules added outside the transformer blocks.

4. Experimental Results

4.1. Implementation Details

Dataset: We evaluate our method on the PASCAL MTL dataset [8]. Following other papers in MTL literature [31, 33, 34], we use the PASCAL-Context split that has annotations for various dense prediction tasks such as semantic segmentation, human part detection, surface normals estimation, and saliency distillation. It has 4,998 images in the training split and 5,105 in the validation split.

Evaluation metrics: Following common multi-task learning evaluation practices [31], the semantic segmentation, saliency estimation, and human part segmentation tasks are evaluated using mean intersection over union (mIoU). We use the root mean square error (rmse) in the predicted angles to evaluate the surface normals task. We also measure the overall performance Δm as the average per-task reduction in performance compared to the single-task baseline st:

$$\Delta m = \frac{1}{T}\sum_{i=1}^{T}(-1)^{l_i}(M_i - M_{st,i})/M_{st,i} \tag{3}$$

where l_i = 1 if a lower value means better for performance measure M_i of task i, and 0 otherwise. The single-task performance is measured for a fully converged model that uses the same backbone network only for that task.

Implementation: *MTLoRA* is implemented using PyTorch, and the code is publicly available on GitHub. Our main artifact is an easily pluggable *MTLoRALinear* layer that encapsulates our *TS-LoRA* and *TA-LoRA* modules, enabling the model to adapt to different tasks by using task-specific low-rank matrices. We use rank 4 for the task-specific matrices while we explore different ranks for the shared matrices. We adopt the publicly available *Swin Transformer* backbone [22], which was pre-trained on the ImageNet dataset [6] as our shared encoder. Then, we attach simple task-specific decoders for different dense tasks. Specifically, we use a simple decoder similar to the one in HR-Net [32], which includes linear and bilinear upsampling layers to efficiently perform dense vision tasks, and we adapt the number of output dimensions to different tasks. The number of decoder parameters is only 6% of the overall MTL model's parameters when using *Swin-Tiny* as a backbone. We run each experiment on a single NVIDIA V100 GPU.

Training: To train our multi-task learning model, we use a loss function equal to the weighted sum of the losses of the various downstream tasks as follows:

$$Loss_{MTL} = \sum_{i}^{T}\omega_{task\_i} \times L_{task\_i} \tag{4}$$

where ω_{task_i} and L_{task_i} are the weight and the loss of the various tasks in the MTL model, respectively. Specifically, we use the standard per-pixel cross-entropy for semantic segmentation and human part segmentation, $L1$ loss for surface normals estimation, and balanced cross-entropy for saliency detection. We also adopt the task weights used by Vandenhende *et. al.* [31].

4.2. Baselines

To evaluate the performance of *MTLoRA*, we compare its accuracy and the number of training parameters to other parameter-efficient training methods. We first compare to *Single task* baselines where each task has a separate model and all parameters are fine-tuned to minimize the loss on the corresponding task. We also build a multi-task learning model with a shared encoder and task-specific decoders and evaluate the post-training accuracy when only decoders are fine-tuned (*MTL - Tuning Decoders Only*) and when the full model is fine-tuned (*MTL - Full Fine-Tuning*) to reduce the overall tasks losses as shown in equation 4.

Table 1. Results - MTLoRA versus SOTA parameter efficient training methods. The table summarizes the number of trainable parameters in each method. It also includes the accuracy of the downstream tasks as well as the average MTL model's accuracy (Δm). The last column indicates whether or not the model allows all the tasks to be executed simultaneously. The symbols ↑ and ↓ indicate higher and lower is better, respectively. **bold** numbers highlights how *MTLoRA* dominates the full finetuning while training $3.6\times$ less parameters.

Method	**SemSeg** ($mIoU\uparrow$)	**Human Parts** ($mIoU\uparrow$)	**Saliency** ($mIoU\uparrow$)	**Normals** ($rmse\downarrow$)	$\Delta m(\%)$	**Trainable Parameters** (M)	**Single Inference For All Tasks**
Single Task	67.21	61.93	62.35	17.97	0	112.62	×
MTL - Tuning Decoders Only	65.09	53.48	57.46	20.69	-9.95	1.94	✓
MTL - Full Fine Tuning	67.56	60.24	65.21	16.64	+2.23	30.06	✓
Adapter [10]	69.21	57.38	61.28	18.83	-2.71	11.24	×
Bitfit [35]	68.57	55.99	60.64	19.42	-4.60	2.85	×
VPT-shallow [15]	62.96	52.27	58.31	20.90	-11.18	2.57	×
VPT-deep [15]	64.35	52.54	58.15	21.07	-10.85	3.43	×
Compactor [16]	68.08	56.41	60.08	19.22	-4.55	2.78	×
Compactor++ [16]	67.26	55.69	59.47	19.54	-5.84	2.66	×
LoRA [12]	70.12	57.73	61.90	18.96	-2.17	2.87	×
VL-Adapter [29]	70.21	59.15	62.29	19.26	-1.83	4.74	×
HyperFormer [23]	71.43	60.73	65.54	17.77	+2.64	72.77	×
Polyhistor [21]	70.87	59.15	65.54	17.77	+2.34	8.96	×
MTLoRA ($r = 16$)	68.19	58.99	64.48	17.03	+1.35	4.95	✓
MTLoRA ($r = 32$)	67.74	59.46	64.90	16.59	+2.16	6.08	✓
MTLoRA ($r = 64$)	67.9	59.84	65.40	16.60	**+2.55**	**8.34**	✓
MTLoRA+ ($r = 4$)	68.12	57.77	63.14	17.60	-0.52	2.57	✓
MTLoRA+ ($r = 8$)	68.54	58.30	63.57	17.41	+0.29	3.15	✓
MTLoRA+ ($r = 16$)	68.28	58.70	64.323	17.034	+1.19	4.29	✓

As mentioned earlier, *MTLoRA* is the first to achieve parameter-efficient training for multi-task learning models. Therefore, to evaluate our method against state-of-the-art methods, we compare *MTLoRA* to other single-task parameter-efficient training methods where a task-wise module is added for each task using the setup provided by Liu *et al.* [21]. Specifically, we compare to the following baselines: (1) Adapter [10] where task-specific bottleneck modules are placed into transformer layers (2) Bitfit [35] where only biases, patch merging layers, and patch projection layers are fine-tuned. (3) VPT [15] where tunable embeddings (i.e., 50 embeddings per layer) are inserted in the first input layer (VPT-shallow) and all layers (VPT-deep). (4) Compacter [16], which decomposes the fast matrix into two low-rank vectors, and Compacter++, which only places modules after MLP layers. (5) LoRA [12], where the low-rank decomposition is applied on attention layers with rank r = 4 and the adapter output scale (i.e., 4), which matches our *MTLoRA* hyperparameter. (6) VL-Adapter [29], which shares an adapter across different tasks. (7) Hyperformer [23], where a hyper-network is used to produce the weights for adapters for various tasks. (8) Polyhistor [21], where decomposed hyper-networks are used to share information across different tasks while still necessitating separate training and inference paths for each task.

4.3. Quantitative Analysis

For a fair comparison, *MTLoRA*, *MTLoRA+*, as well as all other baselines, are all based on the *Swin-Tiny* variant of the Swin Transformers family of models [22]. Table 1 shows the per-task accuracy, the overall MTL accuracy based on Equation 3, the number of trainable parameters of *MTLoRA* and *MTLoRA+* compared to our baselines. The last column indicates whether or not the corresponding parameter-efficient training method allows all the tasks to be executed simultaneously. As mentioned earlier in Figure 2, the ability to execute all tasks in a single inference path is essential for applications where efficiency and latency are critical. As shown in Figure 1, *MTLoRA* and *MTLoRA+* offer a Pareto for the trade-off between the number of trainable parameters and the accuracy of the downstream tasks.

4.4. Ablation

Effect of task-specific modules in MTLoRA: To show the effectiveness of the task-specific Low-rank-decomposition modules in *MTLoRA*, we compare the performance of *MTLoRA* and *MTLoRA+* to a similar setup with only task-agnostic low-rank decomposition modules. The results of this comparison, as shown in Figure 4, clearly demonstrate the impact of adding task-specific adaptation modules. Notably, the integration of these modules results in a substantial improvement in the accuracy-efficiency trade-off during parameter-efficient fine-tuning. This enhancement indicates the ability of the task-specific modules to effectively untangle the parameter space involved in MTL. Consequently, this leads to positive knowledge sharing during the fine-tuning process, significantly boosting the performance of each downstream task.

Effect of Various Backbone Adaptation Locations: As mentioned in Subsection 3.2, we insert low-rank decom-

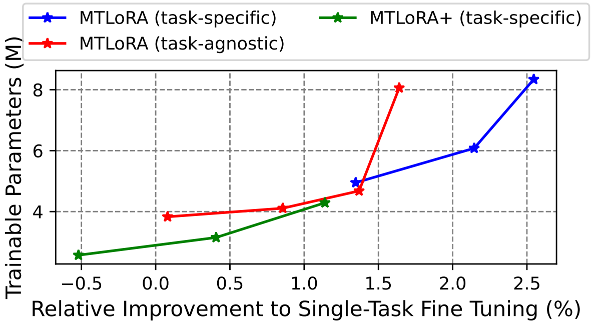


Figure 4. Accuracy versus trainable parameters of MTLoRA with task-agnostic vs task-specific adaptation modules.

Table 2. Effect of removing the various low-rank decomposition matrices in *MTLoRA* from the different locations in the backbone vision transformer. *None* refers to the default of MTLoRA, where all the low-rank modules are adopted.

	None	*No FC1*	*No FC2*	*No Proj*	*No QKV*
Δm	+2.55	+1.93%	+1.65%	+1.65%	+1.26%
Trainable (M)	8.34	6.81	6.81	7.73	7.21

position modules in four different locations in the Hierarchical Transformer encoder: 1) the feed-forward layers in the MLP block (FC1 and FC2), 2) the QKV layer, and 3) the projection layer. We analyze the effect of removing the low-rank decomposition modules from each of those layers to get insights about the effectiveness of adapting each of those weights. Table 2 shows the overall MTL accuracy (Δm) versus the number of trainable parameters when the low-rank decomposition modules are removed from each of the four locations. Those results are from MTLoRA applied on a Swin-Tiny backbone where all low-rank decomposition modules have a rank of 32. We can see that adapting the QKV weights is the most important since removing its low-rank modules causes the most accuracy degradation. On the other hand, removing the adaptation from the first linear layer of the MLP block seems the least effective in comparison. However, the table shows that each low-rank decomposition module offers a significant improvement in the overall performance of the multiple downstream tasks. Removing some of them can be used to achieve a different accuracy efficiency trade-off during training.

Effect of Freezing Non-Attention Modules: As mentioned earlier in SubSection 3.4, besides fine-tuning the low-rank decomposition matrices, we unfreeze the patch embedding layer, the patch merging layer, the layer normalization, and the position bias in the attention layer. We analyze the effect of freezing these extra modules to provide insights into the accuracy-efficiency trade-off associated with each component. Table 3 shows the overall MTL accuracy (Δm) versus the number of trainable parameters when each of those modules is frozen. We can notice that unfreezing all those modules yields the highest accuracy.

Table 3. Effect of freezing the different modules outside the transformer block.

	Patch Embed	*Layer Norm*	*Position Bias*	*Patch Merging*
Accuracy (W/ ST)	+2.02	+2.06	+1.99	+1.74
Training Parameters (M)	8.34	8.32	8.32	6.80

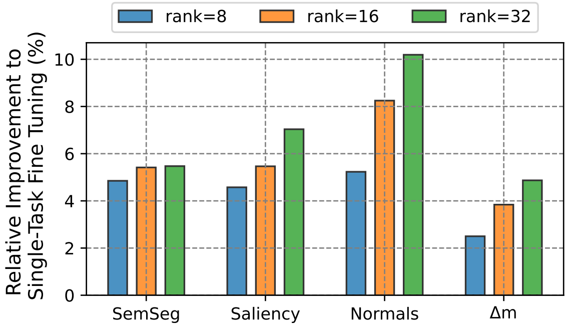


Figure 5. Performance of *MTLoRA* on various downstream tasks when applied to a *Swin-Base* model pre-trained on *ImageNet-22K*

Results with Larger Backbones and Pre-training Datasets: To analyze the efficacy of *MTLoRA* scaled backbone and pre-training datasets. We apply *MTLoRA* on *Swin-Base* that was pre-trained on *ImageNet22k* dataset. Figure 5 shows the improvement in the accuracy of *MTLoRA* compared to single-task models. We can see that MTLoRA scales well with a larger backbone, providing significant improvement compared to the single-task models while training significantly fewer parameters. More analyses are included in the Appendix.

5. Conclusion

In conclusion, this work introduced *MTLoRA*, an innovative framework designed to enable parameter-efficient training for Multi-Task Learning (MTL) models. Central to *MTLoRA* are the Task-Agnostic and Task-Specific Low-Rank Adaptation modules, which are instrumental in effectively disentangling the parameter space during MTL fine-tuning. Those modules allow the fine-tuning to balance both task specialization and interaction within MTL environments. We have demonstrated the application of *MTLoRA* in hierarchical-transformer-based MTL architectures, tailoring them to a variety of downstream dense prediction tasks. Our experiments show that *MTLoRA* not only surpasses the accuracy of fully fine-tuned MTL models but also achieves this with a substantially lower number of trained parameters (3.6× reduction in trainable parameters). Additionally, *MTLoRA* provides Pareto-optimality in the trade-off between the number of trainable parameters and accuracy compared to existing state-of-the-art parameter-efficient training approaches.

References

[1] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. Adversarial training for multi-context joint entity and relation extraction. *arXiv preprint arXiv:1808.06876*, 2018. 3

[2] Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023. 3

[3] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023. 3

[4] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020. 1

[5] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3, 6

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. 3

[8] Mark Everingham, Luc Van Gool, Christopher Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010. 6

[9] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023. 3, 6

[10] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021. 7

[11] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 817–825, 2023. 3

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1, 3, 4, 7

[13] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023. 1, 3

[14] Adrián Javaloy and Isabel Valera. Rotograd: Gradient homogenization in multitask learning. *arXiv preprint arXiv:2103.02631*, 2021. 2, 4

[15] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 4, 7

[16] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021. 7

[17] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 3

[18] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021. 2

[19] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 2

[20] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 3

[21] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *Advances in Neural Information Processing Systems*, 35:36889–36901, 2022. 1, 3, 7

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1, 4, 6, 7

[23] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021. 1, 3, 4, 7

[24] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1860, 2019. 3

[25] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016. 2, 3

[26] Marina Neseem, Ahmed Agiza, and Sherief Reda. Adamtl: Adaptive input-dependent inference for efficient multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4729–4738, 2023. 3

[27] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4822–4829, 2019. 2

[28] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 3

[29] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022. 4, 7

[30] 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

[31] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 527–543. Springer, 2020. 3, 4, 5, 6

[32] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020. 6

[33] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018. 2, 3, 6

[34] Hanrong Ye and Dan Xu. Inverted pyramid multi-task transformer for dense scene understanding. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 514–530. Springer, 2022. 4, 6

[35] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021. 7

[36] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023. 3

[37] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34 (12):5586–5609, 2021. 1, 3

[38] Quan Zhou, Wenbing Yang, Guangwei Gao, Weihua Ou, Huimin Lu, Jie Chen, and Longin Jan Latecki. Multi-scale deep context convolutional neural networks for semantic segmentation. *World Wide Web*, 22:555–570, 2019. 5