DiTASK: Multi-Task Fine-Tuning with Diffeomorphic Transformations

Supplementary Material

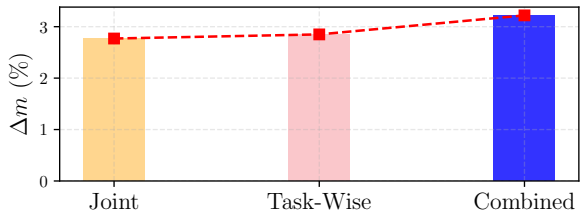

Figure 7. Effect of task-specific and task-agnostic components

A. DiTASK Adaptation Process

In this section, we present the pseudocode for our DiTASK. We repeat this adaptation process in every training iteration for every transformer stage in the encoder, as shown in Figure 4.

DiTASK $(\mathbf{W}, \boldsymbol{\theta}_j, \{\boldsymbol{\theta}_k\}_{k=1}^K, f, \mathbf{x}, \{\mathbf{x}_k\}_{k=1}^K)$

1. Compute the Singular Value Decomposition (SVD) of $\mathbf{W}$:
   A. $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, where: $\mathbf{U} \in \mathbb{R}^{c_2 \times c_2}, \boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_p), \mathbf{V} \in \mathbb{R}^{c_1 \times c_1}$.

// Joint Adaptation

2. $\boldsymbol{\Sigma}_J = \text{diag}(f^{\boldsymbol{\theta}_j}(\sigma_1), \cdots, f^{\boldsymbol{\theta}_j}(\sigma_p))$.
3. Construct $\mathbf{W}_J = \mathbf{U}\boldsymbol{\Sigma}_J\mathbf{V}^\top$.
4. $\mathbf{h} = \mathbf{W}_J\,\mathbf{x}$

// Task-Specific Adaptation

5. For $k = 1, \cdots, K$ do
   A. $\mathbf{x}_k = \mathbf{x}$ if not last block, else $\mathbf{x}_k$
   B. $\boldsymbol{\Sigma}_k = \text{diag}(f^{\boldsymbol{\theta}_k}(\sigma_1), \cdots, f^{\boldsymbol{\theta}_k}(\sigma_p))$.
   C. Construct $\mathbf{W}_k = \mathbf{U}\boldsymbol{\Sigma}_k\mathbf{V}^\top$.
   D. $\mathbf{h}_k = \mathbf{W}_k\,\mathbf{x}_k$
6. **Return** $\mathbf{h}, \{\mathbf{h}_k\}_{k=1}^K$

B. Gradient Analysis

We analyze the memory requirements for low-rank adaptation methods, such as LoRA, and compare them with DiTASK in terms of gradient storage.

LoRA adapts a pre-trained weight matrix $\mathbf{W} \in \mathbb{R}^{c_2 \times c_1}$ using two learnable low rank matrices $\mathbf{B} \in \mathbb{R}^{c_2 \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times c_1}$. During backpropagation, gradients need to be stored for both $\mathbf{B}, \mathbf{A}$, and the input $\mathbf{x}$, resulting in a memory requirement that scales with $rc_2(c_2 + c_1) + c_1c_2$. This scaling depends directly on the rank r, which can make LoRA memory-intensive when r is large or when the dimensions c_1 and c_2 are significant.

In contrast, DiTASK leverages the singular value decomposition (SVD) of $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, where $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \cdots, \sigma_p), p = \min(c_1, c_2)$. DiTASK adapts $\mathbf{W}$ by learning transformations on the singular values $\boldsymbol{\Sigma}$ using a small set of parameters $\boldsymbol{\theta}$. This reduces the gradient storage requirement to $\mathcal{N}_\mathcal{P} + p + c_1c_2$, where $\mathcal{N}_\mathcal{P}$ is the number of intervals over which the CPA velocity field is defined. Unlike LoRA, DiTASK avoids gradients tied to low-rank matrices, significantly reducing memory usage for tasks with high-rank requirements or large input dimensions.

By operating directly on the singular values, DiTASK achieves a more memory-efficient adaptation strategy while retaining the ability to make task-specific updates. This efficiency makes it particularly advantageous for large-scale models.

C. Experimental and Implementation Details

Training. We train using AdamW [29] optimizer with StepLR scheduler for 300 epochs on 8 NVIDIA A6000 GPUs (batch size 64 per GPU).

Code. Our implementation closely follows the codebase of MTLoRA [1] (MIT License), which we modified for our requirements. We refactored the code to allow distributed training.

Hyperparameters. The hyperparameters specific to DiTASK are the tessellation size $\mathcal{N}_\mathcal{P}$ for each joint and task-wise transformations. In all our experiments, we perform a hyperparameter grid search using Weights & Biases framework [4]. All our experiments were performed on a single node with 8 NVIDIA A6000 Ada GPUs using distributed data-parallel (DDP) training

- **PASCAL MTL:** We used the tessellation size of CPAB transformations in $\{16, 32, 64, 128\}$, dropout in $\{0.0, 0.05, 0.5\}$, learning rate in $\{0.005, 0.0005, 0.00005\}$, warmup epochs for StepLR in $\{20, 30, 40\}$, weight decay in $\{0.05, 0.005, 0.0005, 0.00005\}$ and a total training epochs of 300 with a batch size of 64 per GPU.
- **NYUD:** We used the tessellation size of CPAB transformations in $\{16, 32, 64, 128\}$, dropout in $\{0.0, 0.05, 0.5\}$, learning rate in $\{0.005, 0.0005\}$, warmup epochs for StepLR in $\{10, 20, 30\}$, weight decay in $\{0.05, 0.005, 0.0005, 0.00005\}$ and a total train-

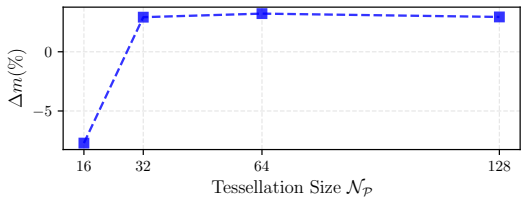

Figure 8. Effect of tessellation size using DITASK's performance on PASCAL MTL

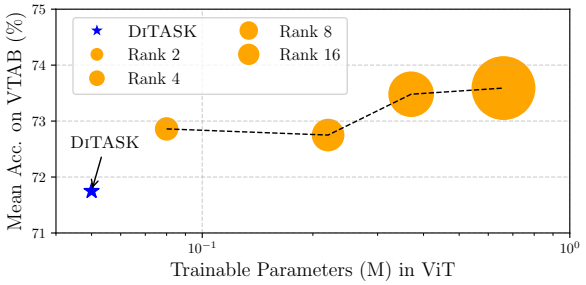

Figure 9. Pareto optimal curve on VTAB benchmark using DITASK and LoRA. DITASK achieves competitive performance using $\sim 10\times$ fewer parameters.

ing epochs of 100 with a batch size of 64 per GPU.

Evaluation Metrics. We follow the evaluation protocol of MTLoRA [1]:

- Task-specific metrics: MIOU for segmentation tasks and RMSE for surface normals and depth estimation.
- Average relative improvement across tasks:

$$\Delta m = \frac{1}{K}\sum_{k=1}^{K}(-1)^{l_k}\frac{(M_k - M_{st,k})}{M_{st,k}}, \qquad (7)$$

where M_k is the performance on task k, $M_{st,k}$ is the single-task baseline. $l_k = 1$ for metrics where lower is better, and 0 otherwise.

D. Additional Ablations

CPAB Parameterization. The CPAB transformations are parameterized by the number of subintervals $\mathcal{N}_\mathcal{P}$ of the domain Ω. From Figure 8, we observe that a moderate-sized $\mathcal{N}_\mathcal{P} = 32$ provides strong and stable performance.

Pre-training Scale. Models pre-trained on ImageNet-21k mostly outperform their ImageNet-1k counterparts (Table 5), suggesting that models pre-trained on larger datasets learn novel representations and, by preserving them, make DITASK more effective.

Table 5. MTL Performance using DITASK on PASCAL for varying pre-training dataset scale

Task ↓ / Dataset →	ImageNet-1k	ImageNet-21k
SEMSEG	70.09	69.06
HUMAN PARTS	59.03	62.02
SALIENCY	64.55	65.00
NORMALS	17.47	17.10

VTAB Benchmark. To understand the single-task generalization capabilities of our method, we compare DITASK and LoRA [19] using the ViT [12] backbone on the VTAB Benchmark [59]. Shown in Figure 9, DITASK achieves competitive performance with an order of magnitude fewer learnable parameters, presenting itself as a strong fine-tuning method for vision transformers.

Table 6. MTL Performance of selected baselines vs. DITASK using Pyramid Vision Transformer (PVT) and Swin-Tiny backbones with different parameter budgets.

Method	$\Delta m(\%)$	**Trainable Backbone Parameters** (M)
PVT + LoRA ($r = 4$)	-1.35	2.41
Swin-Tiny + LoRA ($r = 4$)	-2.17	0.93
Swin-Tiny + LoRA ($r = 8$)	+4.93	1.31 ($\times 4$)
PVT + MTLoRA ($r = 64$)	+1.2	8.69
Swin-Tiny + MTLoRA ($r = 16$)	+1.35	3.01
Swin-Tiny + MTLoRA ($r = 32$)	+2.16	4.14
Swin-Tiny + MTLoRA ($r = 64$)	+2.55	6.40
PVT + DITASK	**+3.01**	1.96
Swin-Tiny + DITASK	**+3.22**	1.61
(Single Task) Swin-Tiny + DITASK	**+5.33**	1.61 ($\times 4$)

Additional Experiments. To understand the robustness of our design for different backbones, we evaluate DITASK using the Pyramid Vision Transformer (PVT) [51] backbone against various parameter budgets of MTLoRA and LoRA. From Table 6, we conclude that our DITASKachieves 2.5× improvement in multi-task performance over best-performing baseline using 4.4× fewer trainable backbone parameters.

E. Qualitative Comparison

Figure 10 shows semantic segmentation results on the PASCAL MTL [13] dataset, comparing MTLoRA and DITASK against the ground truth. DITASK consistently produces sharper and more accurate segmentations. In the first row, it captures the structure and boundaries of bicycles more precisely, whereas MTLoRA oversmooths the outputs, failing to recover fine details. Similarly, in the third row, DITASK segments smaller objects, such as zebras, with greater detail, avoiding omissions observed in MTLoRA. For complex indoor scenes, such as rows five and six, DITASK distinguishes multiple objects effectively and maintains segmentation coherence, whereas MTLoRA generates fragmented outputs. These results highlight DITASKs ability to adapt pre-trained weights through diffeomorphic transforma-

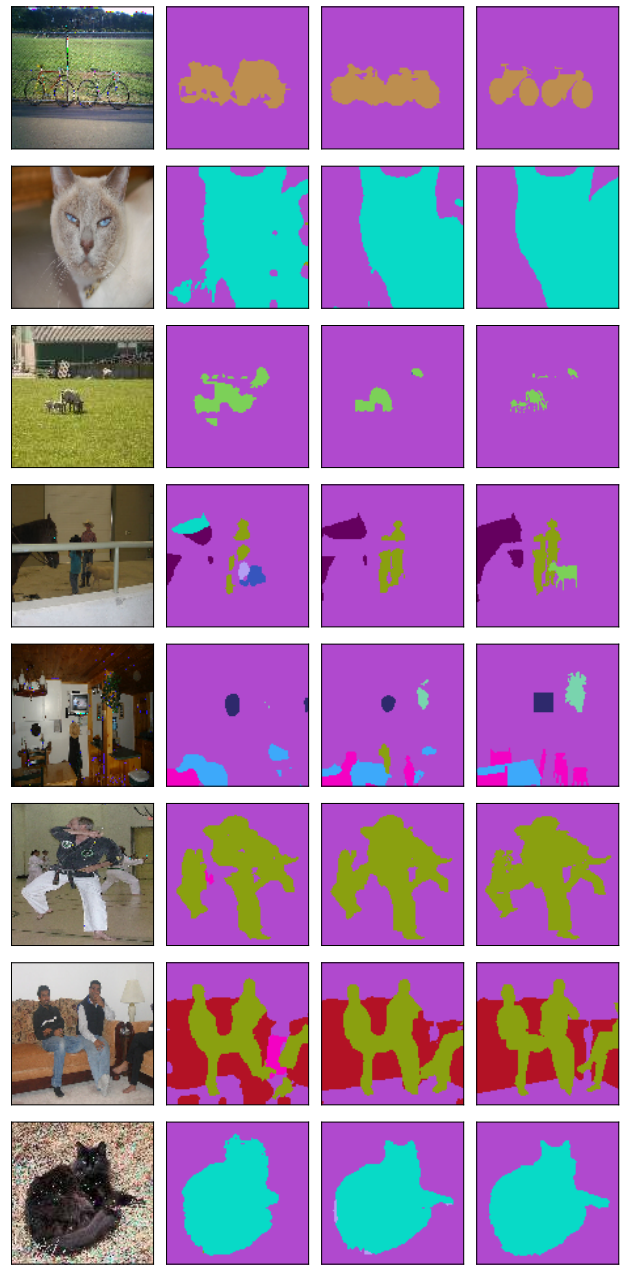

Figure 10. Qualitative comparison of semantic segmentation on representative samples from the PASCAL MTL dataset with MTLoRA and our DiTASK

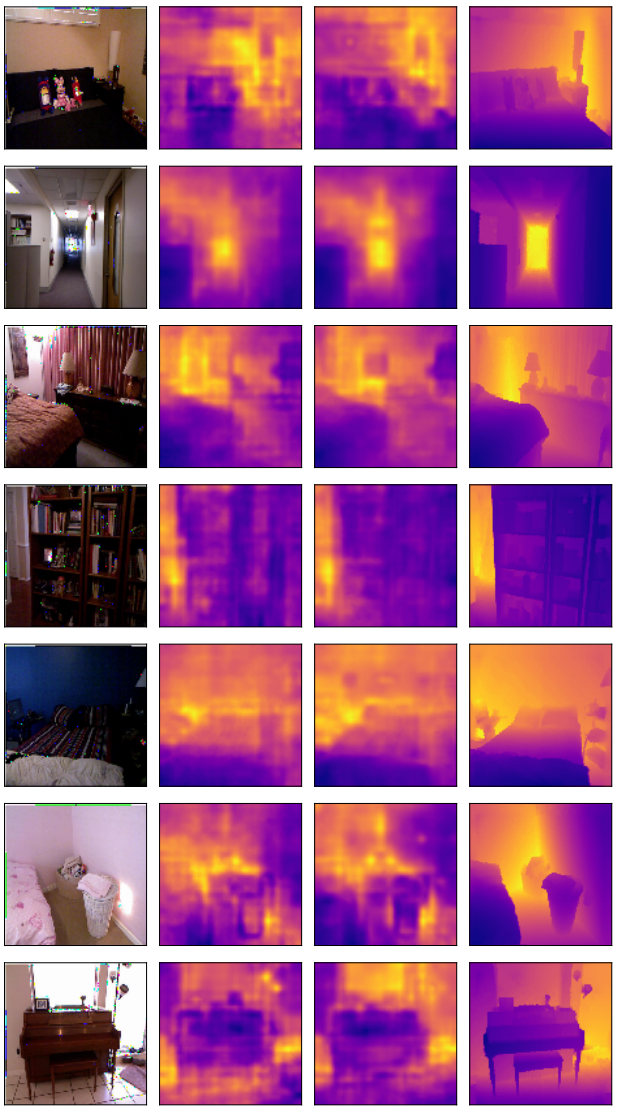

Figure 11. Qualitative comparison of depth estimation on representative samples from the NYUD MTL dataset with MTLoRA and our DiTASK

tions, enabling accurate segmentation across diverse object categories.

Figure 11 demonstrates depth estimation results on the NYUD MTL [41] dataset, comparing MTLoRA and DiTASK. DiTASK captures fine-grained depth variations and preserves object boundaries more effectively than MTLoRA. In the second row, DiTASK separates the hallways foreground and background accurately, closely matching the ground truth, while MTLoRA produces oversimplified and blurred outputs. In the fourth row, DiTASK preserves depth discontinuities and object structures, where MTLoRA fails to capture these transitions. Even in challenging scenes, such as the last row, DiTASK achieves detailed and consistent depth predictions, outperforming MTLoRA. These results val-

idate the effectiveness of DITASKs singular value transformations in producing precise, task-specific depth estimates.

F. Additional Related Work

Hard Parameter Sharing and Task Dynamics in Multi-Task Learning. Hard parameter sharing is a widely used approach in multi-task learning (MTL), where most layers of a neural network are shared among tasks, while task-specific layers are restricted to the output heads. This technique, introduced in [7], is computationally efficient but presents challenges due to task interference, where conflicting task gradients degrade performance. Approaches like PCGrad [57] mitigate this by enforcing gradient orthogonality, but they do not always address the full extent of task competition for limited shared parameters. Despite these challenges, task synergies can be harnessed through careful parameter modulation, allowing shared features to benefit related tasks. In this context, methods like our DITASK enhance positive transfer by preserving crucial pre-trained feature structures, enabling both task-agnostic and task-specific adaptations through diffeomorphic transformations.

Paradigms in Multi-Task Learning: Model Design and Optimization Strategies. Research on MTL can be categorized into two main paradigms: optimization-driven and model design-based strategies. Optimization approaches focus on balancing task-specific loss functions or modifying task gradients to reduce interference, as seen in works like PCGrad [57]. In contrast, model design-based methods, such as adapters [17] and LoRA [19], introduce parameter-efficient layers that balance shared and task-specific features. However, these methods often restrict updates to low-rank subspaces, limiting adaptability. MTLoRA [1] extends LoRA by incorporating task-specific subspaces, yet still faces trade-offs between task isolation and synergy. Our DITASK addresses these limitations by preserving full-rank features and enabling dynamic, parameter-efficient adaptations, achieving superior performance on MTL benchmarks.