Resolving Token-Space Gradient Conflicts: Token Space Manipulation for Transformer-Based Multi-Task Learning

Supplementary Material

A. Additional Related Works

Multi-Task Architectures. Various multi-task architectures can be categorized based on how the parameters or features of the sharing network are distributed among tasks. The widely used shared trunk structure comprises a common encoder shared by multiple tasks and a dedicated decoder for each task [11, 41, 53, 71]. A tree-like architecture, with multiple division points for each task group, offers a more generalized structure [4, 25, 40, 56]. The cross-talk architecture employs separate symmetrical networks for each task, utilizing feature exchange between layers at the same depth for information sharing between tasks [20, 61]. The prediction distillation model [16, 57, 61, 72] incorporates cross-task interactions at the end of the shared encoder, while the task switching network [19, 42, 54, 55] changes network parameters depending on the task.

B. Experimental Settings

B.1. Datasets

We evaluate our method on three benchmarks: NYUD-v2, PASCAL-Context, and Taskonomy. NYUD-v2 contains 4 vision tasks: Our evaluation is based on depth estimation, semantic segmentation, surface normal prediction, and edge detection. PASCAL-Context contains 5 tasks: We evaluate semantic segmentation, human parts estimation, saliency estimation, surface normal prediction, and edge detection. We used 11 tasks for Taskonomy: We evaluate Depth Euclidean (DE), Depth Zbuffer (DZ), Edge Texture (ET), Keypoints 2D (Key2D), Keypoints 3D (Key3D), Normal (N), Principal Curvature (PC), Reshading (R), Segment Unsup 2d (S2D), and Segment Unsup 2.5D (S25D).

B.2. Implementation Details

For experiments, we adopt ViT [14] pre-trained on ImageNet-22K [12] as the transformer encoder. The models are trained for 60,000 iterations on both NYUD [51] and PASCAL [17] datasets with batch size 6. We use Adam optimizer with learning rate 2×10^{-5} and 1×10^{-6} of a weight decay with a polynomial learning rate schedule. Following the previous works [66, 67], the cross-entropy loss is used for semantic segmentation, human parts estimation, and saliency, edge detection. Surface normal prediction and depth estimation use L1 loss.

B.3. Design and Implementation Strategy

To improve efficiency, we perform SVD only once early in training to estimate the feature space for conflict analysis. Gradient conflicts are measured in a pairwise manner across tasks, and the average number of conflicts in each space is used to guide token expansion. Based on this, we statically allocate a small number of task-specific tokens (six in our setup) as learnable parameters, independently applied at each layer. These tokens are fixed during training and do not adapt dynamically. For NYUD-v2 and PASCAL-Context, we use the full training sets to compute gradient statistics, while for Taskonomy, covariance is estimated using 100 randomly sampled mini-batches. The assignment of Token Modulation (TM) and Token Expansion (TE) is determined by a manually chosen activation ratio, which we analyze in Fig. 6. Rather than activating all components uniformly, TM and TE are selectively applied to layers with the highest conflict levels, either individually or jointly, based on their effectiveness in reducing task interference.

B.4. Evaluation

For semantic segmentation, we utilize mean Intersection over Union (mIoU). Surface normal prediction performance is measured by the mean angular distance between the predicted output and ground truth. Depth estimation is evaluated using Root Mean Squared Error (RMSE). For saliency estimation and human part segmentation, we employ mIoU. Edge detection is assessed using the optimal-dataset-scale F-measure (odsF). For Taskonomy, we adopt RMSE for principal curvature and L1 distance for the remaining tasks.

C. Additional Experiments

Comparison with Multi-Task Optimization. In Tabs. 8 to 10, we further evaluate the proposed DTME-MTL against previous multi-task optimization approaches using different backbone sizes. Our method demonstrates significant improvements in multi-task performance with minimal increases in parameters. Specifically, DTME-MTL results in a parameter increase of 0.089% for ViT-L, 0.23% for ViT-S, and 0.46% for ViT-T.

Analysis on the Modulator Configuration. In Tab. 11, we show the performance difference based on the configuration of the token modulators. Specifically, we compared the outcomes obtained when employing affine transformation and batch normalization, which could be considered as the most common and straightforward approaches. Through experi-

Table 8. Comparison with multi-task optimization approaches on Taskonomy across 11 different tasks with ViT-L. Non-converged results are indicated with a dash.

Task	DE	DZ	EO	ET	Key2D	Key3D	N	PC	R	S2D	S25D	
Metric	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist.	RMSE ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	$\triangle_m \uparrow (\%)$
ST	0.0141	0.0146	0.0992	0.1716	0.1631	0.0801	0.2133	0.7134	0.1342	0.1688	0.1419	0.00
GD	0.0153	0.0156	0.1196	0.1757	0.1729	0.0896	0.2215	0.7451	0.1576	0.1826	0.1537	-8.92
GradDrop	0.0170	0.0195	0.1235	0.1757	0.1753	0.0909	0.2818	0.7679	0.1663	0.1916	0.1543	-17.07
MGDA	-	-	-	-	-	-	-	-	-	-	-	-
UW	0.0152	0.0155	0.1195	0.1755	0.1728	0.0897	0.2356	0.7436	0.1569	0.1830	0.1538	-9.36
DWA	0.0153	0.0156	0.1197	0.1757	0.1730	0.0897	0.2214	0.7441	0.1576	0.1827	0.1537	-8.96
PCGrad	0.0152	0.0156	0.1192	0.1749	0.1699	0.0893	0.2310	0.7475	0.1577	0.1825	0.1480	-8.63
CAGrad	0.0155	0.0156	0.1175	0.1756	0.1649	0.0860	0.2421	0.7544	0.1591	0.1854	0.1554	-9.32
IMTL	0.0151	0.0156	0.1194	0.1755	0.1726	0.0895	0.2199	0.7432	0.1569	0.1824	0.1533	-8.57
Align-MTL	0.0150	0.0155	0.1136	0.1733	0.1633	0.0862	0.2512	0.8029	0.1643	0.1803	0.1445	-8.78
Nash-MTL	0.0151	0.0154	0.1138	0.1732	0.1644	0.0863	0.2507	0.7656	0.1544	0.1833	0.1452	-7.95
FAMO	0.0153	0.0157	0.1196	0.1757	0.1730	0.0897	0.2221	0.7444	0.1575	0.1830	0.1534	-8.99
DTME-MTL	0.0127	0.0130	0.1088	0.1731	0.1665	0.0852	0.1654	0.6890	0.1389	0.1661	0.1404	+2.41

Table 9. Comparison with multi-task optimization approaches on Taskonomy across 11 different tasks with ViT-S. Non-converged results are indicated with a dash.

Task	DE	DZ	EO	ET	Key2D	Key3D	N	PC	R	S2D	S25D	
Metric	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist.	RMSE ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	$\triangle_m \uparrow (\%)$
ST 0.0255	0.0255	0.1285	0.1727	0.1653	0.0918	0.3973	0.8562	0.1864	0.1824	0.1647	0.00	
GD	0.0244	0.0243	0.1501	0.1778	0.1844	0.1009	0.4105	0.9087	0.2325	0.2032	0.1822	-8.04
GradDrop	0.0253	0.0253	0.1533	0.1785	0.1865	0.1021	0.4399	0.9246	0.2408	0.2063	0.1791	-10.42
MGDA	-	-	-	-	-	-	-	-	-	-	-	-
UW	0.0242	0.0242	0.1498	0.1778	0.1847	0.1007	0.4064	0.9079	0.2312	0.2033	0.1822	-7.74
DWA	0.0242	0.0242	0.1500	0.1778	0.1844	0.1008	0.4097	0.9071	0.2316	0.2032	0.1822	-7.84
PCGrad	0.0248	0.0248	0.1501	0.1755	0.1761	0.1001	0.4306	0.9181	0.2371	0.2023	0.1772	-8.12
CAGrad	0.0254	0.0255	0.1516	0.1738	0.1698	0.0983	0.4535	0.9282	0.2442	0.2068	0.1849	-9.74
IMTL	0.0236	0.0237	0.1456	0.1756	0.1760	0.0988	0.4151	0.9055	0.2222	0.2010	0.1794	-5.74
Align-MTL	0.0266	0.0264	0.1499	0.1736	0.1700	0.0986	0.4659	0.9868	0.2604	0.2030	0.1780	-11.51
Nash-MTL	0.0235	0.0235	0.1432	0.1745	0.1718	0.0975	0.4230	0.9225	0.2268	0.1985	0.1775	-5.41
FAMO	0.0243	0.0243	0.1499	0.1778	0.1846	0.1008	0.3841	0.9080	0.2321	0.2027	0.1816	-7.31
DTME-MTL	0.0196	0.0200	0.1372	0.1754	0.1712	0.0958	0.3129	0.8333	0.1955	0.1907	0.1698	+3.62

Table 10. Comparison with multi-task optimization approaches on Taskonomy across 11 different tasks with ViT-T. Non-converged results are indicated with a dash.

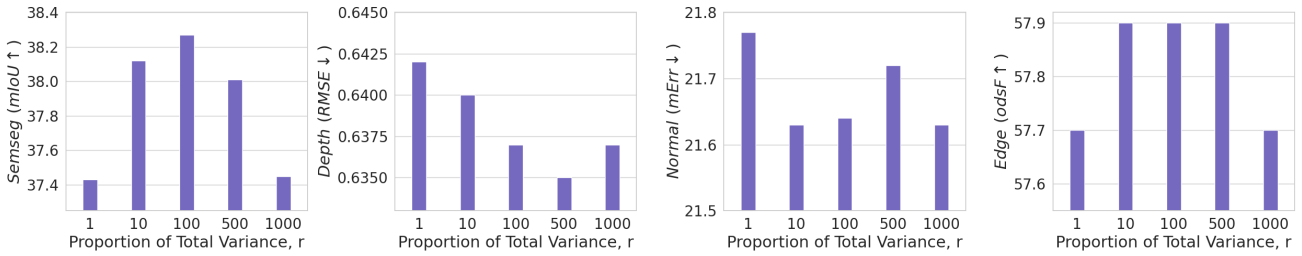
Task	DE	DZ	EO	ET	Key2D	Key3D	N	PC	R	S2D	S25D	
Metric	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist.	RMSE ↓	L1 Dist. ↓	L1 Dist. ↓	L1 Dist. ↓	$\triangle_m \uparrow (\%)$
ST	0.0250	0.0256	0.1388	0.1755	0.1670	0.0958	0.3856	0.9066	0.2132	0.1878	0.1722	0.00
GD	0.0266	0.0278	0.1593	0.1794	0.1865	0.1047	0.4752	0.9467	0.2568	0.2081	0.1897	-11.10
GradDrop	0.0276	0.0284	0.1624	0.1807	0.1884	0.1064	0.4741	0.9611	0.2658	0.2108	0.1860	-12.67
MGDA	-	-	-	-	-	-	-	-	-	-	-	-
UW	0.0266	0.0277	0.1593	0.1795	0.1865	0.1045	0.4757	0.9466	0.2567	0.2080	0.1896	-11.07
DWA	0.0266	0.0274	0.1593	0.1794	0.1866	0.1045	0.4743	0.9465	0.2567	0.2080	0.1897	-10.95
PCGrad	0.0273	0.0285	0.1596	0.1768	0.1807	0.1043	0.4785	0.9689	0.2644	0.2080	0.1854	-11.55
CAGrad	0.0290	0.0305	0.1641	0.1747	0.1731	0.1051	0.4884	0.9870	0.2828	0.2136	0.1945	-14.64
IMTL	0.0263	0.0272	0.1558	0.1772	0.1810	0.1025	0.4730	0.9525	0.2458	0.2065	0.1868	-9.24
Align-MTL	-	-	-	-	-	-	-	-	-	-	-	-
Nash-MTL	0.0261	0.0270	0.1536	0.1762	0.1766	0.1017	0.4590	0.9649	0.2496	0.2039	0.1846	-8.28
FAMO	0.0266	0.0275	0.1592	0.1795	0.1865	0.1047	0.4746	0.9466	0.2566	0.2080	0.1898	-10.97
DTME-MTL	0.0236	0.0241	0.1494	0.1765	0.1790	0.0998	0.4138	0.8921	0.2290	0.1959	0.1824	-2.88

Table 11. We compare task performance based on the configuration of the modulator. Specifically, we compare the performance of tasks using an affine transformation against those using a batch normalization layer as configurations for the modulator.

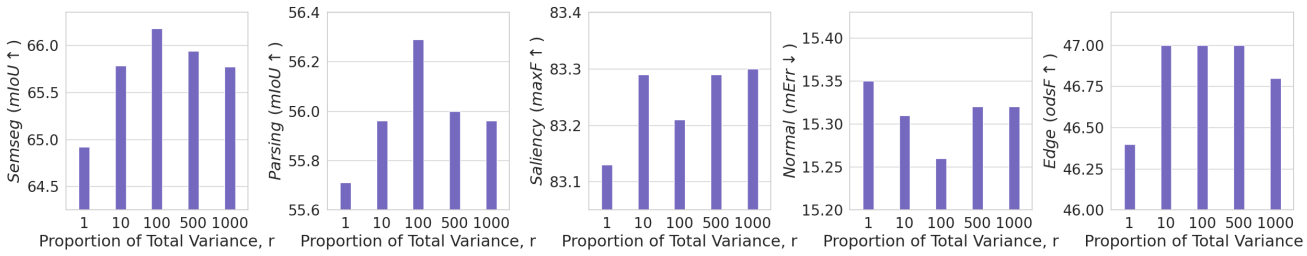
Model	NYUD-v2				PASCAL-Context				
	Semseg mIoU ↑	Depth RMSE ↓	Normal mErr ↓	Edge odsF ↑	Semseg mIoU ↑	Parsing mIoU ↑	Saliency maxF ↑	Normal mErr ↓	Edge odsF ↑
TM+TE (Affine)	**38.27**	**0.6370**	**21.64**	**57.90**	**66.18**	**56.29**	**83.21**	**15.26**	**47.00**
TM+TE (Batch Norm)	37.42	0.6550	23.16	56.10	60.80	53.29	82.59	15.73	44.90

Table 12. We assess task performance by comparing scenarios where we freeze the backbone network after expansion (w/ Freeze) and where we don't (w/o Freeze).

Model	NYUD-v2				PASCAL-Context				
	Semseg mIoU ↑	Depth RMSE ↓	Normal mErr ↓	Edge odsF ↑	Semseg mIoU ↑	Parsing mIoU ↑	Saliency maxF ↑	Normal mErr ↓	Edge odsF ↑
TM+TE (w/ Freeze)	34.80	0.6730	22.48	56.00	58.34	52.96	82.86	15.63	43.20
TM+TE (w/o Freeze)	**38.27**	**0.6370**	**21.64**	**57.90**	**66.18**	**56.29**	**83.21**	**15.26**	**47.00**

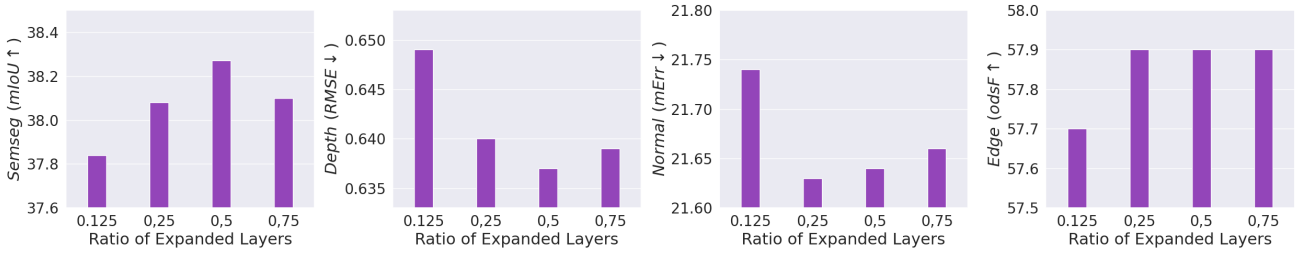


(a) Results on NYUD-v2.

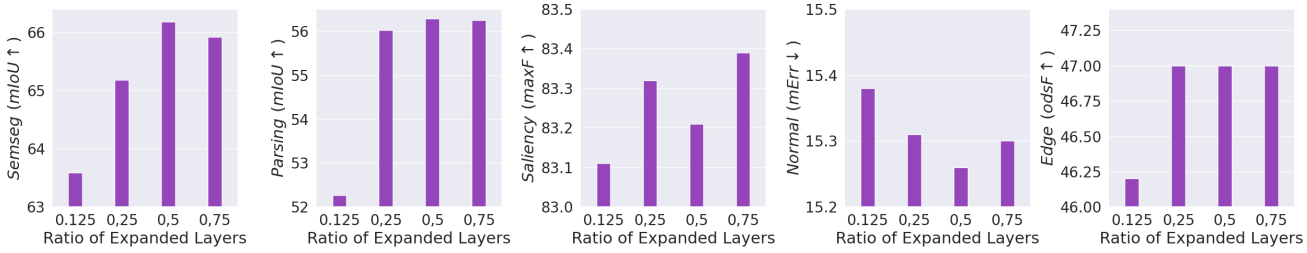


(b) Results on PASCAL-Context.

Figure 5. We assess the performance of tasks based on the proportion of total variance r. The results are displayed for both (a) NYUD-v2 and (b) PASCAL-Context.



(a) Results on NYUD-v2.



(b) Results on PASCAL-Context.

Figure 6. The performance of tasks based on the ratio of the number of expanded layers to the total number of layers. The results are displayed for both (a) NYUD-v2 and (b) PASCAL-Context.

ments, we find that affine transformations consistently exhibit better performance across all tasks compared to batch normalization layers used as modulators for both datasets.

Analyzing Performance Differences with Backbone Network Freezing. In Tab. 12, we examine the performance variation based on whether we freeze the existing backbone network components when training the expanded network after implementing the proposed dynamic token modulation and expansion. The results indicate that training networks without freezing the existing backbone network com-

ponents leads to significantly better performance compared to training networks with freezing. We guess that allowing modifications to the learned token space after expansion helps the network to dynamically partition the token space for each task.

Influence of r on SVD Approximation. In Fig. 5, we illustrate how the proportion of total variance r impacts the approximation of a token's range and null space. We assess the performance of tasks across five values of r (1, 10, 100, 500, 1000). Our results suggest that the value of r has minimal impact on task performance, implying that there is less need for extensive tuning of the r parameter to optimize performance. In our other experiments, we chose r as 100 for training.

Impact of the Number of Layers Expanded by DTME-MTL. DTME-MTL expands a subset of transformer layers selected based on the severity of gradient conflicts. In Fig. 6, we analyze how varying the number of expanded layers affects task performance. The x-axis denotes the ratio of expanded layers to the total number of layers. We observe that applying TM+TE to approximately 25%–50% of the layers yields consistent performance gains across tasks while maintaining parameter efficiency. Performance improves as more high-conflict layers are expanded, but begins to degrade when expansion exceeds 50%, especially when low-conflict layers are included. This suggests that over-expansion can be detrimental. Table 2 further confirms that using a moderate expansion ratio (50%) avoids overfitting, whereas Fig. 6 highlights that indiscriminate expansion into less conflicting layers harms performance. These findings underscore the importance of both the *extent* and *location* of TM+TE application.

Effect of Swapping Conflict Types. In Tab. 13, we present the results of an experiment on NYUD-v2 where we intentionally swap the conflict types targeted by each method. Specifically, Token Expansion (TE) is applied to layers with severe range space conflict, and Token Modulation (TM) is applied to layers with severe null space conflict—opposite to our standard configuration. This reversal leads to a clear performance drop, confirming that each method is most effective when applied to the type of conflict it is designed to resolve. These results support our design choice of assigning TM to range space conflict and TE to null space conflict.

Table 13. Performance comparison across selection strategies.

Method	Random	Reverse	Swap	Standard
$\triangle_m \uparrow (\%)$	-2.966	-6.167	-2.608	+0.044

D. Additional Analysis

Further Justification for Targeted TM/TE Assignment. Prior work [46] suggests that fine-tuning from a pretrained model tends to remain in the same loss basin, preserving the structure of the pretrained feature space. Accordingly, we view the token space during fine-tuning as constrained by the span of the pretrained features. If the conflict lies within this span (i.e., the range space), it can be resolved by rotating the token space—achievable via a modulator, since affine transformations include rotation. However, if the conflict resides in the null space, it lies outside the span and cannot be sufficiently addressed by modulation alone. In such cases, expanding the token space with task-specific tokens helps relax this constraint. We theoretically support this view in Propositions 1 and 2 (with proofs in Appendix E), which analyze how each method addresses conflict in its respective subspace. This is further validated empirically: we measure the reduction in gradient conflicts by comparing the start and end of training in each space (Tab. 14). The results show that Token Modulation (TM) is more effective in reducing conflicts in the range space, while Token Expansion (TE) is more effective in the null space. This consistency between theoretical analysis and empirical behavior supports our design choice to selectively apply TM and TE based on the dominant type of conflict in each layer.

Token-Level vs. Parameter-Level Conflict Handling. Parameter-space conflicts reflect an aggregate gradient across all tokens, which makes it difficult to localize or disentangle the source of interference. In contrast, token-level conflicts can be measured for each individual token, allowing more localized and fine-grained analysis. This granularity enables our method to selectively modulate or expand tokens based on where the conflict occurs. Furthermore, by decomposing the token space into range and null components—depending on whether the pretrained model already spans those directions—we adaptively apply Token Modulation (TM) or Token Expansion (TE) to address conflicts. Such space-aware conflict resolution is fundamentally infeasible in parameter space, where task interference is entangled across layers and tokens.

Comparison with LoRA in Multi-Task Inference. As shown in Tab. 1, the baseline (ST) corresponds to full fine-tuning and serves as an upper bound on performance. While LoRA [26] is a parameter-efficient method, assigning a separate LoRA module for each task leads to disjoint sets of task-specific weights. Even when merged into the base model, these configurations require separate forward passes per task, negating the efficiency benefits of multi-task learning (MTL). In contrast, our method maintains shared weights across tasks and allows all outputs to be computed jointly in a single batched tensor operation on GPU. This enables highly parallel inference with only a 13.4% overhead per task, whereas the inference time in LoRA scales linearly with the number of tasks.

Table 14. Reduction in gradient conflict numbers (NYUD-v2).

Method	Num($g_{\mathcal{R},i} \cdot g_{\mathcal{R},i} \leq 0$)	Num($g_{\mathcal{N},i} \cdot g_{\mathcal{N},i} \leq 0$)
TM	11.60 % $\downarrow$	8.92 % $\downarrow$
TE	4.64% $\downarrow$	15.44 % $\downarrow$

E. Theoretical Analysis

E.1. Proof of Proposition 1

Proposition 1. *When the input token $\mathcal{T}_{in}$ for input sample $\mathcal{X}_l$ spans the range space of $\tilde{\mathcal{T}}_s$, optimizing the token modulators $\{\mathcal{M}_i\}_{i=1}^{\mathcal{K}}$ reduces gradient conflicts in the row space of $\tilde{\mathcal{T}}_s$ and leads to a reduction in the multi-task loss.*

Proof. Let the loss function $\mathcal{L}_i$ be a function of the shared parameters Θ_s, the token modulator $\mathcal{M}_i$, and the input data $\mathcal{X}_l$. Since transformers convert input data into tokens, we consider the loss to be a function of one of the input tokens, $\mathcal{T}_{in}$, rather than $\mathcal{X}^l$. To represent the updating step during optimization, we use the superscript t for current variables, such as Θ_s^t, and $\mathcal{M}_i^t$, and $t+1$ for the next-step variables, such as Θ_s^{t+1}, and $\mathcal{M}_i^{t+1}$.

In cases where the input token $\mathcal{T}_{in}$ spans the row space of $\tilde{\mathcal{T}}_s$, this can be expressed as follows:

$$\mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T\nabla_{\mathcal{T}_{in}}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_{in})\simeq 0 \tag{5}$$

Since the row space and null space are perpendicular to each other, with their dimensions summing to the entire space, the following holds according to Eq. (5):

$$\sum_{i=1}^{\mathcal{K}}\nabla_{\mathcal{T}_{in}}\mathcal{L}_i=\sum_{i=1}^{\mathcal{K}}(\mathcal{U}_{\mathcal{R}}\mathcal{U}_{\mathcal{R}}^T+\mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T)\nabla_{\mathcal{T}_{in}}\mathcal{L}_i\simeq\sum_{i=1}^{\mathcal{K}}(\mathcal{U}_{\mathcal{R}}\mathcal{U}_{\mathcal{R}}^T)\nabla_{\mathcal{T}_{in}}\mathcal{L}_i \tag{6}$$

Let the token modulator $\mathcal{M}_i$ be a $p\times p$ matrix that manipulates the input token $\mathcal{T}_{in}$.

$$\sum_{i=1}^{\mathcal{K}}\nabla_{\mathcal{T}_{in}}\mathcal{L}_i=\sum_{i=1}^{\mathcal{K}}(\mathcal{U}_{\mathcal{R}}\mathcal{M}_i^t)(\mathcal{U}_{\mathcal{R}}\mathcal{M}_i^t)^T\cdot\nabla_{\mathcal{M}_i^t}\mathcal{L}_i\cdot\nabla_{\mathcal{T}_{in}}\mathcal{M}_i^t \tag{7}$$

The total multi-task loss can be represented using a Taylor expansion. Assuming $\eta\ll 1$, we can ignore the second-order terms of η:

$$\sum_{i=1}^{\mathcal{K}}\mathcal{L}_i(\Theta_s^{t+1},\mathcal{M}_i^{t+1},\mathcal{T}_s)=\sum_{i=1}^{\mathcal{K}}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)+\sum_{i=1}^{\mathcal{K}}\nabla_{\Theta_s^t}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)(\Theta_s^{t+1}-\Theta_s^t) \tag{8}$$

$$+\sum_{i=1}^{\mathcal{K}}\nabla_{\mathcal{M}_i^t}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)(\mathcal{M}_i^{t+1}-\mathcal{M}_i^t) \tag{9}$$

$$=\sum_{i=1}^{\mathcal{K}}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)-\eta|\sum_{i=1}^{\mathcal{K}}\nabla_{\Theta_s}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)|^2 \tag{10}$$

$$-\eta\sum_{i=1}^{\mathcal{K}}|\nabla_{\mathcal{M}_i^t}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_s)|^2 \tag{11}$$

By optimizing the modulator $\mathcal{M}_i^t$ so that $|\nabla_{\mathcal{M}_i^t}\mathcal{L}_i(\Theta_s^t,\mathcal{M}_i^t,\mathcal{T}_{in})|$ approaches zero for each task $i=1,2,\ldots,\mathcal{K}$, we can alleviate gradient conflicts in the row space of $\tilde{\mathcal{T}}_s$ (as Eq. (7) also approaches zero) and reduce the overall multi-task loss, since Eq. (11) is always greater than or equal to zero. □

E.2. Proof of Proposition 2

Proposition 2. *When the input token $\mathcal{T}_{in}$ for input sample $\mathcal{X}_l$ spans the null space of $\tilde{\mathcal{T}}_s$, token expansion using $\{\mathcal{T}_i\}_{i=1}^{\mathcal{K}}$ alleviates the increase in multi-task loss caused by gradient conflicts in the null space of $\tilde{\mathcal{T}}_s$.*

Proof. Let the loss function $\mathcal{L}_i$ be a function of the shared parameters Θ_s^t, the task-specific token $\mathcal{T}_i^t$, and the input data $\mathcal{X}^t$. Similarly, since transformers convert input data into tokens, we consider the loss to be a function of one of the input tokens, $\mathcal{T}_{in}^t$, rather than $\mathcal{X}^t$. To represent the updating step during optimization, we use the superscript t for current variables, such as Θ_s^t, $\mathcal{T}_{in}^t$ and $\mathcal{M}_i^t$, and $t+1$ for the next-step variables, such as Θ_s^{t+1}, $\mathcal{T}_{in}^{t+1}$ and $\mathcal{M}_i^{t+1}$.

In the case where the input token $\mathcal{T}_{in}^t$ spans the null space of $\tilde{\mathcal{T}}_s$, this can be expressed as follows:

$$\sum_{i=1}^{\mathcal{K}}\mathcal{U}_{\mathcal{R}}\mathcal{U}_{\mathcal{R}}^T\nabla_{\mathcal{T}_{in}^t}\mathcal{L}_i(\Theta_s^t,\mathcal{T}_{in}^t,\mathcal{T}_i^t)\simeq 0 \tag{12}$$

The derivative of the task-specific loss $\mathcal{L}_i$ with respect to the expanded token, including the input token $\mathcal{T}_{in}^t$ and the learnable task-specific tokens $\mathcal{T}_i^t$, is given as follows:

$$\sum_{i=1}^{\mathcal{K}} \nabla_{[\mathcal{T}_{in}^t, \mathcal{T}_i^t]} \mathcal{L}_i \tag{13}$$

$$= \sum_{i=1}^{\mathcal{K}} \left(\begin{bmatrix} \mathcal{U}_{\mathcal{R}} & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & \mathcal{U}_{\mathcal{R},i} \end{bmatrix} \begin{bmatrix} \mathcal{U}_{\mathcal{R}} & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & \mathcal{U}_{\mathcal{R},i} \end{bmatrix}^T + \begin{bmatrix} \mathcal{U}_{\mathcal{N}} & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & 0_{\mathcal{K}\times\mathcal{K}} \end{bmatrix} \begin{bmatrix} \mathcal{U}_{\mathcal{N}} & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & 0_{\mathcal{K}\times\mathcal{K}} \end{bmatrix}^T \right) \begin{bmatrix} \nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i \\ \nabla_{\mathcal{T}_i^t} \mathcal{L}_i \end{bmatrix} \tag{14}$$

$$= \sum_{i=1}^{\mathcal{K}} \begin{bmatrix} \mathcal{U}_{\mathcal{R}}\mathcal{U}_{\mathcal{R}}^T + \mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & \mathcal{U}_{\mathcal{R},i}\mathcal{U}_{\mathcal{R},i}^T \end{bmatrix} \begin{bmatrix} \nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i \\ \nabla_{\mathcal{T}_i^t} \mathcal{L}_i \end{bmatrix} \tag{15}$$

$$\simeq \sum_{i=1}^{\mathcal{K}} \begin{bmatrix} \mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T & 0_{d\times\mathcal{K}} \\ 0_{\mathcal{K}\times d} & \mathcal{U}_{\mathcal{R},i}\mathcal{U}_{\mathcal{R},i}^T \end{bmatrix} \begin{bmatrix} \nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i \\ \nabla_{\mathcal{T}_i^t} \mathcal{L}_i \end{bmatrix} \tag{16}$$

$$= \sum_{i=1}^{\mathcal{K}} \begin{bmatrix} (\mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T)\nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i \\ (\mathcal{U}_{\mathcal{R},i}\mathcal{U}_{\mathcal{R},i}^T)\nabla_{\mathcal{T}_i^t} \mathcal{L}_i \end{bmatrix} \tag{17}$$

The total multi-task loss can be expressed as follows:

$$\mathcal{L}_i(\Theta_s^{t+1}, \mathcal{T}_{in}^{t+1}, \mathcal{T}_i^{t+1}) = \mathcal{L}_i(\Theta_{in}^t, \mathcal{T}_s^t, \mathcal{T}_i^t) + \nabla_{\Theta_s^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_s^t, \mathcal{T}_i^t)(\Theta_s^{t+1} - \Theta_s^t) \tag{18}$$

$$+\nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_s^t, \mathcal{T}_i^t)(\mathcal{T}_{in}^{t+1} - \mathcal{T}_{in}^t) \tag{19}$$

$$+\nabla_{\mathcal{T}_i^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_s^t, \mathcal{T}_i^t)(\mathcal{T}_i^{t+1} - \mathcal{T}_i^t) \tag{20}$$

$$= \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) - \eta \nabla_{\Theta_s^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_s^t, \mathcal{T}_i^t) \cdot \sum_{i=1}^{\mathcal{K}} \nabla_{\Theta_s^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) \tag{21}$$

$$-\eta(\mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T)\nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) \cdot \sum_{i=1}^{\mathcal{K}} (\mathcal{U}_{\mathcal{N}}\mathcal{U}_{\mathcal{N}}^T)\nabla_{\mathcal{T}_{in}^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) \tag{22}$$

$$-\eta(\mathcal{U}_{\mathcal{R},i}\mathcal{U}_{\mathcal{R},i}^T)\nabla_{\mathcal{T}_i^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) \cdot (\mathcal{U}_{\mathcal{R},i}\mathcal{U}_{\mathcal{R},i}^T)\nabla_{\mathcal{T}_i^t} \mathcal{L}_i(\Theta_s^t, \mathcal{T}_{in}^t, \mathcal{T}_i^t) \tag{23}$$

The increase in multi-task loss caused by gradient conflicts in the null space (as described in Eq. (22)) cannot be reduced since the shared token $\mathcal{T}_{in}^t$ is not a learnable parameter. Instead, task-specific tokens $\mathcal{T}_i^t$ can be added to mitigate the increase in multi-task loss due to null space gradient conflicts by optimizing the learnable parameters $\{\mathcal{T}_i\}_{i=1}^{\mathcal{K}}$ as described in Eq. (23). □