TSP-Transformer: Task-Specific Prompts Boosted Transformer for Holistic Scene Understanding

Shuo Wang[1] Jing Li[2] Zibo Zhao[1] Dongze Lian[3]
Binbin Huang[1] Xiaomei Wang[4] Zhengxin Li[1] Shenghua Gao[1,5,6†]
[1]ShanghaiTech University [2]Xiaohongshu Inc. [3]National University of Singapore
[4]Fudan University [5]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[6]Shanghai Engineering Research Center of Energy Efficient and Custom AI IC
{wansghuo2022, zhaozb, liandz, huangbb, lizhx, gaoshh}@shanghaitech.edu.cn
lijing1@alumni.shanghaitech.edu.cn, 17110240025@fudan.edu.cn

Appendices

Appendix. A More Implementation Details

Data Processing. For a fair comparison with ATRC [1], InvPT [6] and TaskPrompter [7], we follow their data processing pipeline. On PASCAL-Context [2], we pad the image to the size of 512 × 512, while on NYUD-v2 [5], we randomly crop the input image to the size of 448 × 576. We use typical data augmentation including random scaling, cropping, horizontal flipping and color jittering.

Implementation Details of Encoder Feature Fusion. For ViT [3] encoders, we follow InvPT [6] implementation choosing 3 layers based on the depth and unfolding their output spatially, and then use transposed convolution to upsample the resolution of feature maps to match the spatial resolution in the corresponding decoder stage before the further transformation. Specifically, for ViT-base encoder, using the output token sequences of layer 3, 6, and 9, while for ViT-large encoder using output token sequences of layer 6, 12, and 18. The encoder feature fusion module procures multi-task encoder features of different scales from the preceding layers. The kernel size and stride of the transposed convolution for the feature at the first scale are 4, and those at the second scale are 2. The fused coarse multi-task encoder feature and the fine multi-task feature after intermediate supervision participate in cross-attention operations to forward in the decoder.

Appendix. B More Experimental Results and Analysis

Shared encoder vs. unshared encoder. We advocate that different tasks are closely related, and a shared encoder makes various tasks share the same low-level features (layers 1-12) and different but task-specific high-level features(layers 13-24). Other than using a shared encoder, we also report the performance based on task-specific encoders where different encoders with task-specific prompts are learned for different tasks. It means that our encoder has a different branch for each task, and the different tasks in the encoder stage are completely independent, without any interaction between tasks and no shared parameters in the encoder.

Method	Semseg (IoU)↑	Depth (RMSE)↓	Normal (mErr)↓	Boundary (odsF)↑
unshared	54.03	0.5121	18.86	**78.00**
shared	**55.39**	**0.4961**	**18.44**	77.50

Table 1. Ablation for shared encoder on NYUD-v2. Performance with the shared encoder is better for all the tasks except for boundary detection.

Results on NYUD-v2 with different encoder design strategies are reported in Table 1. It shows that the results with the shared encoder are better for all the tasks except for boundary detection. The possible reason is that semantic segmentation, depth estimation, and surface normal estimation are more closely related tasks. In contrast, its relation to other tasks is not that strong for boundary detection. The difference in performance gain is the task competition problem in training. Thus learned task-specific encoder may not be a wrong choice for boundary detection.

Prompts Initialization. Prompt tuning first emerged in the field of NLP, and the research on prompt initialization is an important field. Visual prompt tuning [4] also studies the initialization method of the prompts, but it is only a tuning setting. Compared with Visual prompt tuning, the model parameters that can be optimized vary greatly in our multi-task learning method. The conclusions drawn above do not necessarily apply to our multi-task transformer network.

method	SemSeg (IoU)↑	Depth (RMSE)↓	Normal (mErr)↓	Boundary (odsF)↑
zeros	**55.39**	**0.4961**	**18.44**	**77.50**
random	54.31	0.5069	18.67	77.40
ones	54.61	0.4962	18.66	**77.50**

Table 2. Performance with different prompt initialization strategies on NYUD-v2. It's impressive that our default initialization zeros, generally works the best.

We compare the performance using the above initialization strategy against the default zeros initialization in Table 2. As shown in Table 2, it's impressive that our default initialization zeros, works the best in general.

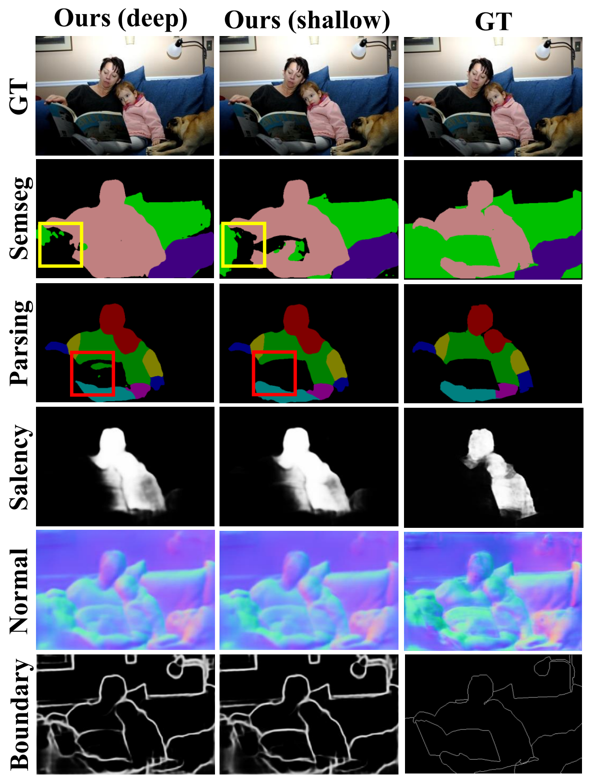

Figure 1. Qualitative analysis of different model variants (shallow: $1-12$ and deep: $13-24$) on PASCAL-Context. Results of different model variants are shown by Table 3.

Dataset size and task numbers. We conduct extensive experiments on NYUD-v2 and PASCAL- Context for performance evaluation, but our experiments reveal inconsistent patterns between the two datasets. Compared to the NYUD-v2 dataset, the PASCAL-Context dataset has a larger data size, a greater number of tasks, and a distinct data distribution. We believe these factors account for the discrepancies observed in some experimental results.

For NYUD-v2, there is no significant difference between the task prompts positions with the same layer range length in the shallow $(1-12)$ and deep $(13-24)$ layers of the encoder as shown in Table 3 (a). In fact, the results are slightly better in the task prompts positions with deeper layers. However, this is not the case with another PASCAL-Context dataset as shown in Table 3 (b). The positions has a more substantial impact on performance, particularly for certain higher-level scene understanding tasks such as Semseg and Parsing. A potential reason for this discrepancy is that high-level tasks demand more task-specific information, particularly as the size of data increases, task prompts positions with shallow layers leading to a finer decoupling of features from encoder.

Computation cost. Table 4 shows the computation cost of our proposed model, detailing the GFLOPs and the number of model parameters across different model variants, along with corresponding performances on the NYUD-v2 and PASCAL-Context datasets. Our model variants demanding higher computational resources do not necessarily guarantee superior overall performance, indicating the

Layers with prompts	Semseg (IoU)↑	Depth (RMSE)↓	Normal (mErr)↓	Boundary (odsF)↑
w/o prompt	53.56	0.5183	19.04	**78.10**
1-12	54.96	**0.4948**	18.72	77.40
13-24	**55.39**	0.4961	**18.44**	77.50

(a) The performance with prompts positions on NYUD-v2

Layers with prompts	Semseg (IoU)↑	Parsing (IoU)↑	Saliency (maxF)↑	Normal (mErr)↓	Boundary (odsF)↑
w/o prompt	79.03	67.61	84.81	14.15	73.00
1-12	**81.48**	**70.64**	**84.86**	**13.69**	**74.80**
13-24	80.64	69.53	84.67	13.83	74.20

(b) The performance with prompts positions on PASCAL-Context

Table 3. The performance with prompts positions on two different datasets. Unlike the NYUD-v2 dataset, the placement depth of the task prompts substantially impacts performance, particularly for higher-level scene understanding tasks such as Semseg and Parsing on PASCAL-Context.

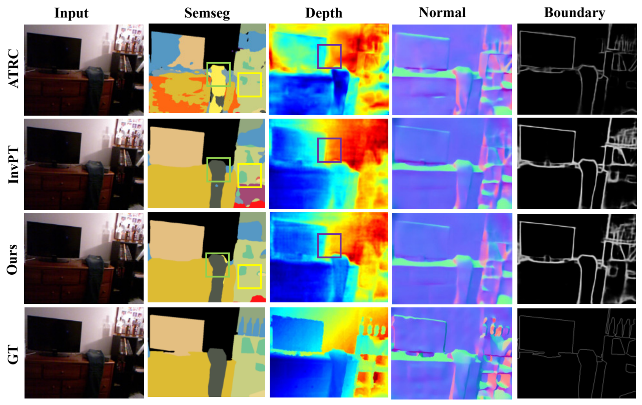

Figure 2. Qualitative comparison with state-of-the-art method on NYUD-v2. Our method generates more accurate predictions.

Layers with prompts	Prompt token numbers	Semseg (IoU)↑	Depth (RMSE)↓	Normal (mErr)↓	Boundary (odsF)↑	GFlops (G)	Number of parameters (M)
w/o prompt (InvPT)	0	53.56	0.5183	19.04	**78.10**	597.67	402.09
24	1	53.97	0.5038	18.63	77.50	654.17	402.10
13-24	5	55.39	0.4961	**18.44**	77.50	1146.24	402.34
1-12	5	54.96	0.4948	18.72	77.40	1681.11	402.34
1-24	5	**55.80**	**0.4898**	18.63	77.60	1685.13	402.58

The performance and compute cost with different prompts inserting positions and prompt token numbers for each task on NYUD-v2.

Layers with prompts	Prompt token numbers	Semseg (IoU)↑	Parsing (IoU)↑	Saliency (maxF)↑	Normal (mErr)↓	Boundary (odsF)↑	GFlops (G)	Number of parameters (M)
w/o prompt (InvPT)	0	79.03	67.61	84.81	14.15	73.00	668.29	422.93
24	1	80.08	69.12	84.46	13.85	74.10	744.94	422.94
13-24	5	80.64	69.53	84.67	13.83	74.20	1412.52	423.24
1-12	5	81.48	70.64	84.86	**13.69**	**74.80**	2138.61	423.24
1-24	5	**81.63**	**70.69**	**84.90**	13.81	74.70	2143.65	423.54

The performance and compute cost with different prompts inserting positions and prompt token numbers for each task on PASCAL-Context.

Table 4. On our most efficient model variant, which introduces only one task prompt token for each task on the last transformer encoder layer, the performance improvement is also significant, accompanied by a slight increase in the number of parameters and GFlops. It can also be seen that the shared vanilla layer placed in the shallow layer (layers with prompts: 24, 13-24) can drastically reduce the computational load in terms of GFlops, and there is almost a negligible increase in the number of parameters across all of our model variants.

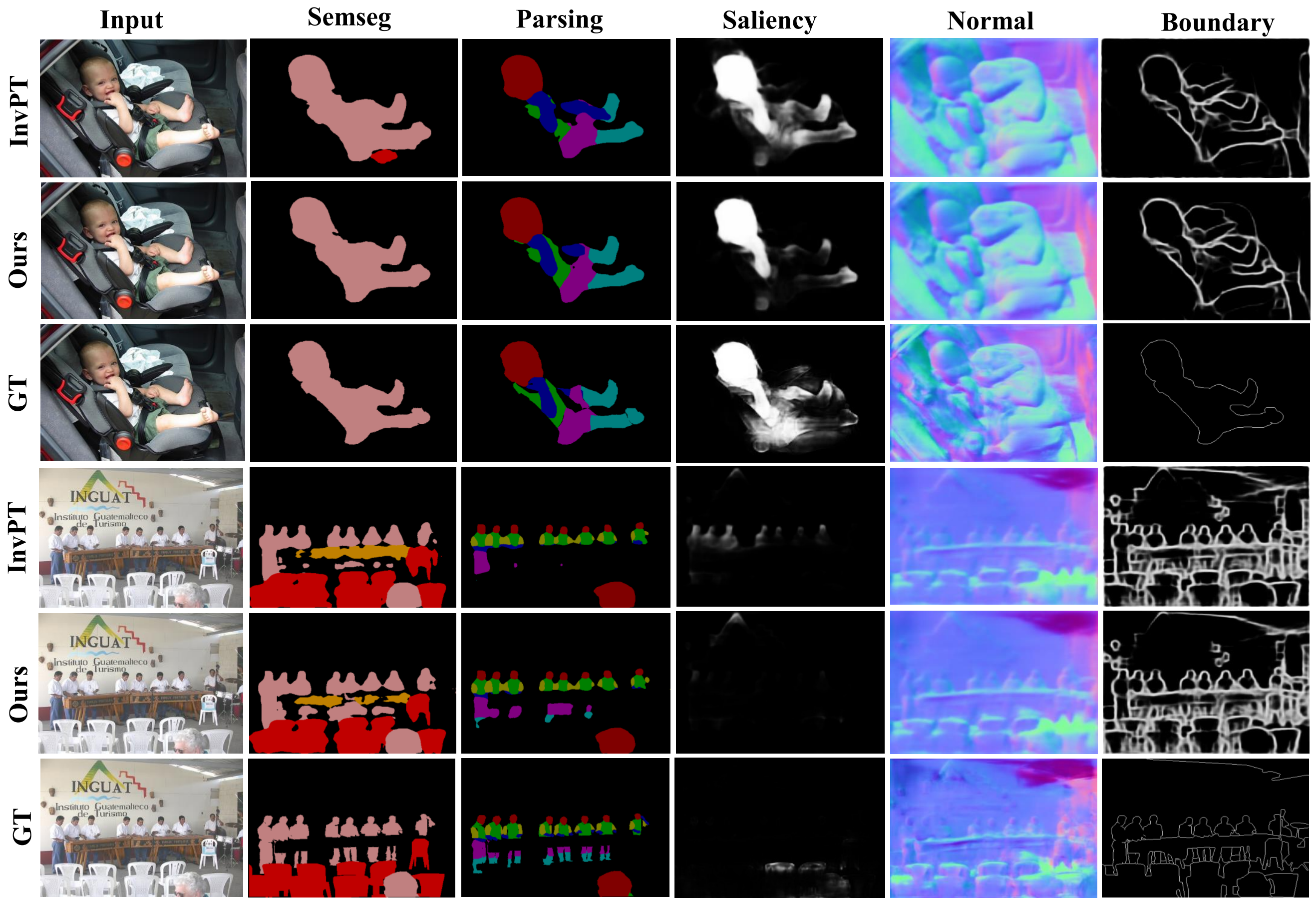

Figure 3. Qualitative comparison with state-of-the-art method on PASCAL-Context. Our method generates more accurate predictions.

existence of a trade-off between performance and computation. The most efficient model variant, which introduces only one task prompt token to each task on the last transformer encoder layer, demonstrates significant performance improvement with a modest increase in parameter count and GFLOPs. Furthermore, our results show that placing the shared vanilla layer in the shallow layer can drastically reduce the computational load in terms of GFLOPs. Moreover, across all our model variants, we observe only a nearly negligible increase in the number of parameters.

More qualitative results. We show more prediction results by our method and InvPT on the NYUD-v2 and PASCAL-Context dataset in Fig. 2 and Fig. 3. It is clear that our method produces significantly better results than InvPT, especially on semantic segmentation, depth estimation and human parsing.

References

[1] David Brüggemann, Menelaos Kanakis, Anton Obukhov, Stamatios Georgoulis, and Luc Van Gool. Exploring relational context for multi-task dense prediction. In *Proc. IEEE/CVF Int. Conf. Computer Vision*, pages 15869–15878, 2021. 1

[2] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1971–1978, 2014. 1

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[4] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022. 1

[5] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conf. Computer Vision*, pages 746–760. Springer, 2012. 1

[6] Hanrong Ye and Dan Xu. Inverted pyramid multi-task transformer for dense scene understanding. *arXiv preprint arXiv:2203.07997*, 2022. 1

[7] Hanrong Ye and Dan Xu. Taskprompter: Spatial-channel multi-task prompting for dense scene understanding. In *The Eleventh Int. Conf. Learning Representations*, 2023. 1