Virtual Task Prompting for Multi-Task Scene Understanding

Zedong Wang* **Yucheng Wang*** **Dan Xu†**
The Hong Kong University of Science and Technology
{zwangmw, ywangls}@connect.ust.hk danxu@cse.ust.hk

Abstract

Multi-task scene understanding models typically rely on modified vision encoders or complex decoders for cross-task interaction, which limits their flexibility and compatibility with off-the-shelf vision foundation models (VFMs). We introduce **V**irtual **T**ask **P**rompting (VTP), which frames multi-task prompting as retrieving task-adaptive compositions from a shared virtual state, where both the state and routing process are optimized in an end-to-end manner. Concretely, VTP maintains an expanded set of virtual tasks alongside the backbone. At each layer, learnable routing matrices read task-specific compositions as input to interact with image tokens and write the processed results back, forming explicit pathways that capture the intricacies among tasks outside the encoder. Since VTP keeps the backbone clean, it integrates seamlessly with pretrained VFMs, advancing the state of the art on Pascal-Context and NYUD-v2. Stress test on Taskonomy with 13 heterogeneous tasks further shows VTP's robustness in extreme settings. Ablations and analysis indicate that VTP's gains stem from adaptive routing rather than increased capacity. The project page is available at `https://jacky1128.github.io/VTP/`.

1 Introduction

Multi-Task Learning (MTL)[7] trains a single model simultaneously over multiple correlated tasks to learn robust, generalizable representations [22]. It has driven rapid progress in computer vision [15, 28, 59], particularly in scene understanding for autonomous driving [13, 23] and embodied AI [8, 21]. As Transformers [56, 16] become the de facto choice for vision tasks, multi-task scene understanding models have followed suit, which typically fall into two groups: (i) Hard-parameter sharing (HPS) employs a shared encoder for task-agnostic representations, followed by specifically-designed decoders, *e.g.*, cross-attention [69, 72], state space models [31], or mixture-of-experts [70], for decoding-stage interaction. (ii) Soft-parameter sharing (SPS) [53] instead maintains multiple parallel backbones, enabling cross-task message passing throughout the entire network architecture.

TaskPrompter [71] sets an alternative to both. It appends learnable task vectors for each task that are concatenated with input image tokens to interact directly through Vision Transformers (ViTs) [16, 54], which combines the parameter efficiency of HPS with the input-to-output inter-task communication in SPS, while replacing complex decoders with lightweight prompt-guided prediction heads. It shows the potential to capture both task-specific and agnostic information at once in a streamlined model.

However, multi-task prompting has yet to deliver on its promise. As shown in Figure 1(a), in order to compensate for its simplified architecture, TaskPrompter [71] partitions task vectors into spatial and channel components for increased capacity, which induces inevitable modifications to the vision encoder (*e.g.*, extra channel attention, channel-wise sliding window) and complicates the decoding with dual-branch processing. More importantly, since most existing vision foundation models (VFMs)

*Equal contribution. †Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

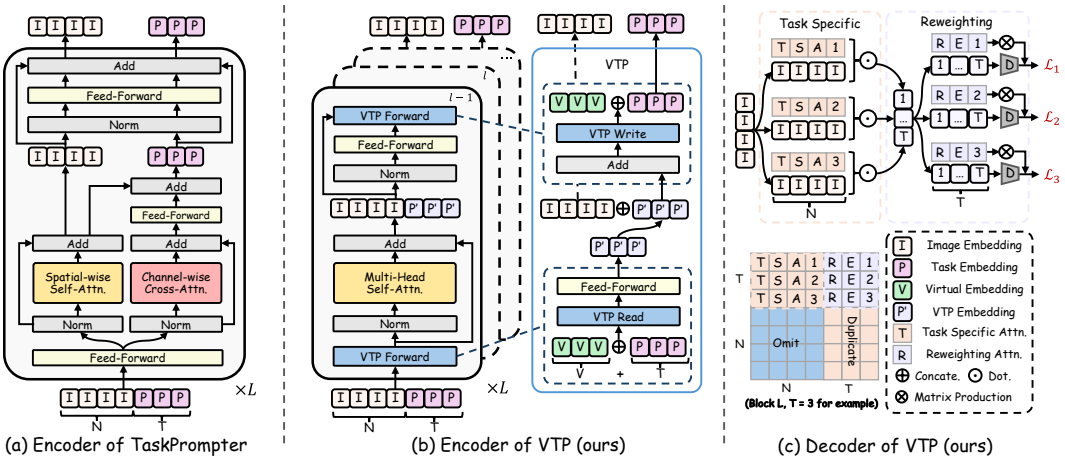


Figure 1: **Overview of Virtual Task Prompting (VTP).** (a) TaskPrompter [71] modifies ViT encoder to compensate for the limited capacity. (b) VTP instead maintains $T+V$ virtual states alongside the encoder, where the model read them to T task vectors to interact with image tokens, and then write the results back to $T+V$ bypass, enriching task representations while maintaining a vanilla ViT block. (c) VTP keeps the prompt-guided decoding pipeline but unifies the spatial-channel parts into one.

standardize the macro design of ViTs for scaling up and open-source their corresponding weights to empower the community [45, 52], such heuristic re-engineering of vision encoder limits the method's flexibility, making it difficult to leverage the power of off-the-shelf VFMs (*e.g.*, weight mismatches). As a result, the MTL community remains largely restricted to the ImageNet priors [16, 69, 70], which hinders both its research progress and real-world applications under limited jointly annotated data.

To push beyond the status quo, we ground this work in a specific question: *how to achieve effective task prompting with sufficient capacity while maintaining a vanilla encoder backbone?* The prompt setup and learning process first caught our attention. Previous studies assign each task a single fixed vector updated via backpropagation through the entire model. Intuitively, however, correlated tasks may benefit from sharing a common vector, whereas a complex task might require complementary knowledge from distinct vectors to capture its intricacies. A natural remedy is to increase the vector count for capacity (*e.g.*, pairing two per task), yet doubling or quadrupling the vectors yields negligible or even negative gains as shown in Table 3. To dig deep, further analysis in Figure 4(a) reveals that the task vectors undergo severe mode collapse across layers, converging to near-identical correlation patterns that might mislead the training into subpar representations, resulting in inefficiency and non-generalizability. Thus, we argue that if we can derive an adaptive prompting method that learns to encode the most expressive patterns for each task to interact with image tokens at different layers.

As such, we propose Virtual Task Prompting (VTP), which achieves effective scene understanding for up to 13 tasks without modifying the encoder's internal topology. Inspired by hyper-connections [82, 61], we cast multi-task prompting as retrieving specific task vector compositions from a larger virtual state, where the vectors can be diverse or partially overlapping, and both the state and the retrieving process are end-to-end optimizable. Concretely, VTP maintains a set of states of *virtual task* alongside the encoder, expanding the task embedding space from T to $T+V$. At each ViT block, learnable routing matrices adaptively reads T task compositions from the $T+V$ virtual state for interaction with image tokens, then writes the results back, providing explicit learning pathways that encourages task specialization and cross-task sharing to emerge jointly outside encoder (Sec. 3.2.1). Each task's representation is thus no longer a fixed vector like static weights but a learned composition drawn from the richer virtual state, where different tasks may share overlappings or access distinct ones. Since VTP operates outside the encoder backbone, it enables seamless integration with diverse VFMs.

We validate VTP on three axes. (i) Standard benchmarks (Sec. 4.2): On Pascal-Context [9] and NYUD-v2 [51], our VTP shows competitive results ($+2.77\%$ and $+1.95\%$). Integrating VFMs [52, 57] further yields a leap ($+5.15\%$ and $+3.80\%$). (ii) Stress Test (Sec. 4.3): On Taskonomy [77] with heterogeneous 13 tasks, VTP offers consistent gains (up to $+9.58\%$), showcasing robustness as the task count grows. (iii) *Analysis*: Naively scaling vectors yields negligible gains, indicating that VTP's gains stem not only from its added capacity. It also shows lower parameter count (342M vs. 401M), higher throughput (Table 3), and more discriminative task patterns (Figure 4) than the prior baseline.

Our contributions can be summarized as follows:

- We introduce Virtual Task Prompting (VTP), a prompting-based architecture that eliminates the need for inner modifications to the vision encoder for multi-task scene understanding.
- By keeping the encoder backbone clean, VTP integrates seamlessly with different off-the-shelf VFMs (*e.g.*, DINOv3 [52] and ViT-5 [57]), substantially advancing the state-of-the-art on standard Pascal-Context [9] and NYUD-v2 [51] without additional modifications.
- We stress-test VTP on Taskonomy [77] with 13 heterogeneous tasks covering both classification and dense regression, where it offers consistent gains. Ablations show that VTP's effectiveness stems from its adaptive read-write routing, not merely increased capacity.

2 Related Work

Multi-task learning seeks to build shared representations that capture both task-specific and cross-task structure. Existing studies pursue this typically from two orthogonal paths: *training algorithms* that re-balance objectives or gradients, and *model architectures* that operate how task representations are built and interact. VTP falls into the latter, enriching task representations through adaptive prompting.

Multi-Task Optimization. Loss scaling balances task objectives by adjusting their weights, progressing from uniform weighting [78] and homoscedastic uncertainty [27] to strategies driven by loss trajectories [37], geometric means [12], equal-rate reduction [33], and progress alignment [49]. Gradient manipulation instead modifies task gradients w.r.t. shared parameters, encompassing magnitude balancing [10], conflict-resolving projection [74], multi-objective Pareto optimization [14] with extensions to worst-case improvement [34] and Nash bargaining [44], principal component alignment [48], and utility maximization [4]. Hybrid methods combine both families [35, 32, 36], while others operate on gradients w.r.t. shared features to reduce cost or semantic disparity [47, 24, 60].

Multi-Task Model Architectures. (i) *CNN-based models*: promote cross-task sharing through multimodal distillation [63], affinity-based task pairing [79], multi-scale interaction [55], and learnable fusion across task-specific streams [43, 46, 17]. However, CNNs are constrained to local receptive fields, which motivates the shift to global attention. (ii) *Transformer-based models*: Transformers [56] offer global modeling and scalable pretrained features, both benefiting MTL. InvPT [69, 72] employs cross-attention for joint modeling of global positions and multiple tasks. MTFormer [64] applies cross-task attention with contrastive learning, MQTransformer [65] proposes query attention to model inter-task dependencies. DeMT [66] combines deformable convolutions with query-based Transformers via shared gating. While effective, these architectures focus on increasingly complex decoders without designs for task-intrinsic priors [80]. (iii) *MoE and state-space models*: Mixture-of-experts (MoE) naturally suit MTL by routing shared experts in a task-conditioned manner [70, 11, 67], though they mainly target decoder components and lack global cross-task sharing. Recent state-space models such as MTMamba [31] and PAME [76] leverage selective scanning for linear-complexity cross-task interaction, but remain rely on specially-tailored encoder or decoder designs for multi-task learning.

Across all these lines, enriching task interaction typically requires modifying the encoder's internals or introducing complex decoders, coupling methods to specific backbones and hindering adaptation of off-the-shelf VFMs [45, 52]. VTP departs from this by operating alongside the vision encoder, preserving full VFM compatibility while providing sufficient capacity for expressive task prompting.

Adapters and Prompting. MTLoRA [1] disentangles shared and task-specific parameters via low-rank adapters. Visual Prompt Tuning [25, 73] introduces learnable prompts into ViT layers for task transfer, with extensions to language and graphs [18, 75]. In the multi-task setting, a growing line of work employs parameter-efficient fine-tuning (PEFT) by inserting lightweight modules into frozen model backbones for dense prediction [5], including hypernetwork-generated adapters [38], shared projections with task-specific extraction [62], task-conditional adapters [30, 26], prompt adaptation or tuning [50], input-conditioned filters [2], and diffeomorphic singular-value modulation [41]. Among these studies, multi-task prompting-based architectures [71, 39, 19] are most related to VTP, as they adopt task prompting with ViTs for joint training. We discuss their limitations that inspire VTP in Sec. 3.1. Unlike the above PEFT and adaptation methods that freeze the backbone and prioritize parameter efficiency, VTP as a multi-task framework jointly trains the entire model with task vectors and targets multi-task performance through adaptive read-write routing of a richer virtual task state.

3 Methodology

This section presents Virtual Task Prompting (VTP). We first introduce the overall framework of multi-task prompting architectures, covering the encoding and decoding pipelines with notation in Sec. 3.1. We then discuss VTP's task-adaptive read-and-write routing mechanism, which maintains a $T{+}V$ virtual state alongside the encoder and feeds into a streamlined decoder in Sec. 3.2. Finally, we conclude with the multi-task joint training objective after VTP's output prediction in Sec. 3.3.

3.1 Overall Framework

Encoding. Given an input RGB image, a patch embedding layer produces $N{=}H{\times}W$ visual tokens $\mathcal{X}^0 = \{x_1^0, x_2^0, \ldots, x_N^0\} \in \mathbb{R}^{N\times C}$, where H and W are spatial resolution after patchifying and C denotes the embedding dimension. To jointly address T correlated tasks, a set of T learnable vectors $\mathcal{P}^0 = \{p_1^0, p_2^0, \ldots, p_T^0\} \in \mathbb{R}^{T\times C}$ is introduced, with one for each task. The task vectors and visual tokens are concatenated to derive a joint input sequence $\mathcal{Z}^0$, as illustrated in Figure 1:

$$\mathcal{Z}^0 = \big[\mathcal{P}^0;\ \mathcal{X}^0\big] = \big[p_1^0, \ldots, p_T^0;\ x_1^0, \ldots, x_N^0\big] \ \in\ \mathbb{R}^{(T+N)\times C}, \tag{1}$$

where $[\,\cdot\,;\,\cdot\,]$ denotes token-wise concatenation. The joint sequence is processed through L ViT [16] layers. At each model layer $l \in \{1, \ldots, L\}$, an encoder block $\mathcal{F}_l(\cdot)$, typically comprising multi-head self-attention, a feed-forward network, and residual connections, updates the joint sequence as:

$$\big[\mathcal{P}^l;\ \mathcal{X}^l\big] = \mathcal{F}_l\big(\big[\mathcal{P}^{l-1};\ \mathcal{X}^{l-1}\big]\big)\,, \tag{2}$$

where $\mathcal{P}^l \in \mathbb{R}^{T\times C}$ and $\mathcal{X}^l \in \mathbb{R}^{N\times C}$ indicate the output task vectors and image tokens within layer l, respectively. Self-attention over all $T{+}N$ tokens yields a $(T{+}N) \times (T{+}N)$ affinity map, allowing each task to absorb relevant cues from the input image and to exchange information with other tasks.

Decoding. After the final layer L, the processed task vectors $\mathcal{P}^L = \{p_t^L\}_{t=1}^T$ and visual tokens $\mathcal{X}^L$ drive task-specific predictions. For each task $t \in \{1, \ldots, T\}$, we define the task-specific affinities $\mathcal{M}^L \in \mathbb{R}^{T\times N}$ (colored orange in Figure 1(c)), where each row $\mathcal{M}^L_{t,:}$ is computed as:

$$\mathcal{M}^L_{t,:} = \mathrm{softmax}\left(\frac{\mathcal{X}^L\, {p_t^L}^\top}{\sqrt{C}}\right) \ \in\ \mathbb{R}^N, \tag{3}$$

which captures how task vector p_t^L attends to the image and highlights the regions that are most relevant to task t. Similarly, we can derive the cross-task affinities $\mathcal{R}^L \in \mathbb{R}^{T\times T}$ (blue in Figure 1(c)), encoding inter-task interactions during training. The decoder $\mathcal{D}_t(\cdot,\cdot,\cdot)$ then produces output $\hat{\mathcal{Z}}_t$ as:

$$\hat{\mathcal{Z}}_t = \mathcal{D}_t\big(\mathcal{M}^L_{t,:},\ \mathcal{R}^L_{t,:},\ \mathcal{X}^L\big) \ \in\ \mathbb{R}^{N\times C}, \tag{4}$$

where $\mathcal{M}^L_{t,:}$ routes shared features to task t and $\mathcal{R}^L_{t,:}$ retrieves complementary information from other tasks based on their affinities. These two operations encode task-specific and agnostic information in a prompt-guided manner, followed by a lightweight convolutional head for the final predictions.

Limitations. This framework combines the parameter efficiency of HPS with the task sharing of SPS. However, (i) each task is represented by a single fixed vector updated solely via backpropagation, which limits its expressiveness (with analysis in Sec 4.4). (ii) To compensate, prior work [71] splits task vectors to spatial and channel ones in Eq. (1) for disentangled processing, inducing modifications to both the encoder $\mathcal{F}_l(\cdot)$ in Eq. (2) (*e.g.*, extra channel attention) and decoder $\mathcal{D}_t(\cdot,\cdot,\cdot)$ in Eq. (4) (*e.g.*, dual-branch prompting). As shown in Figure 1(a), such re-engineering couples the method to specific encoder structures, inevitably hindering its compatibility with off-the-shelf pretrained VFMs.

3.2 Virtual Task Prompting

The limitations above point to key demands for multi-task prompting architectures. (i) The encoder $\mathcal{F}_l(\cdot)$ should stay clean to preserve compatibility with VFMs. (ii) A single vector per task provides insufficient representation capacity. A natural remedy is to increase its scale, yet as shown in Table 3, doubling or quadrupling vectors yields negligible or even negative gains. Our analysis in Sec. 4.4 further shows that the vectors undergo severe mode collapse across layers, converging to near-identical non-discriminative patterns. This motivates a third, (iii) an end-to-end trainable approach to encode and optimize task vectors rather than treating them as static weights updated via backpropagation.

We address these with Virtual Task Prompting (VTP), which operates outside model backbone as a bypass before and after each ViT block. Concretely, VTP introduces V additional vectors as *virtual tasks* alongside the encoder, enlarging the embedding space to $T{+}V$. To achieve discriminative task representations, we propose a read–and-write mechanism for *virtual tasks* inspired by hyper-connections [82, 29, 61], where learnable routing matrices adaptively read T compositions from the $T{+}V$ virtual state at each layer l for interaction with image tokens as in Eq. 2, then write the results back, providing explicit learning pathways and more expressive task features across different layers.

3.2.1 Encoding

As shown in Figure 1(b), virtual task state $\mathcal{P}^0 \in \mathbb{R}^{(T+V)\times C}$ are initialized as learnable embeddings which operate outside the encoder. Their roles are not prescribed but optimized in an end-to-end manner. Some may encode shared structure across correlated tasks, while others capture task-specific patterns. At each ViT layer $l \in \{1, \ldots, L\}$, VTP executes a read–and–write cycle as follows.

State Read. To ensure the encoder receive T task vectors in the standard $(T{+}N) \times C$ dimension, a learnable routing matrix $\Phi^l \in \mathbb{R}^{T\times(T+V)}$ accesses virtual state $\mathcal{P}^{l-1}$ by reading T vectors from it:

$$\tilde{\mathcal{P}}^l = \Phi^l \, \mathcal{P}^{l-1} \;\in\; \mathbb{R}^{T\times C}, \tag{5}$$

where each read vector $\tilde{p}_t^l$ is a learned task-specific composition over the virtual state $\mathcal{P}^l$, encoding richer information than a static vector per task. Note that different tasks may draw on shared states for common structure or access distinct ones for specialization, which is determined by the learnable Φ^l. The T vectors are then concatenated with image tokens and processed by encoder $\mathcal{F}_l(\cdot)$ as:

$$\Big[\hat{\mathcal{P}}^l;\ \mathcal{X}^l\Big] = \mathcal{F}_l\Big(\Big[\tilde{\mathcal{P}}^l;\ \mathcal{X}^{l-1}\Big]\Big), \tag{6}$$

where $\hat{\mathcal{P}}^l \in \mathbb{R}^{T\times C}$ and $\mathcal{X}^l \in \mathbb{R}^{N\times C}$ are the processed task vectors and visual tokens. Because the Read step compresses the virtual state to exactly T vectors, the input retains the standard shape $\mathbb{R}^{(T+N)\times C}$, and $\mathcal{F}_l$ — self-attention, feed-forward network, and residual connections — remains entirely unchanged, preserving direct compatibility with off-the-shelf VFM weights.

State Write. After the encoder processes T vectors, two learnable routing matrices write the updated information back into $T{+}V$ virtual state to maintain the expanded capacity across layers as:

$$\mathcal{P}^l = \Psi^l \, \hat{\mathcal{P}}^l \;+\; \Gamma^l \, \mathcal{P}^{l-1} \;\in\; \mathbb{R}^{(T+V)\times C}, \tag{7}$$

where $\Psi^l \in \mathbb{R}^{(T+V)\times T}$ writes the encoder-processed task vectors into selected virtual slots, and $\Gamma^l \in \mathbb{R}^{(T+V)\times(T+V)}$ carries forward information from the previous virtual state $\mathcal{P}^{l-1}$. The two terms balance what to update from the current layer with what to retain from preceding ones, enabling virtual task vectors to accumulate and populate discriminative task information across layers.

As such, VTP imposes task-adaptive selection to the $T{+}V$ vectors that distills which task patterns are essential enough for encoder interaction, counteracting the mode collapse that arises when vectors are updated through backpropagation alone. Throughout this cycle, the routing operates only on the task-vector stream with $\mathcal{O}((T+V)^2C)$ complexity independent of N. Since $T, V \ll N$ in dense scene understanding, this routing overhead is negligible relative to the encoder cost (as in Sec. 4.5).

3.2.2 Decoding

Since VTP operates on unsplit task vectors, the decoding is simplified compared to the dual-branch design of TaskPrompter [71]. As illustrated in Figure 1(c), for each task t, we got its task-specific affinity $\mathcal{M}^L_{t,:}$ (Eq. 3) and cross-task affinity $\mathcal{R}^L_{t,:}$ during training, then decode the output $\hat{\mathcal{Z}}_t$ as:

$$\hat{\mathcal{Z}}_t = \mathcal{D}_t\big(\mathcal{M}^L_{t,:},\ \mathcal{R}^L_{t,:},\ \mathcal{X}^L\big) = \sum_{k=1}^{T} \mathcal{R}^L_{t,k}\ \big(\underbrace{\mathcal{M}^L_{k,:} \odot \mathcal{X}^L}_{\text{task-specific}}\big), \tag{8}$$

where the inner term extracts task-specific features for each task k by prompting with $\mathcal{M}^L_{k,:}$, and the outer summation aggregates these task-specific features across tasks with the cross-task affinities $\mathcal{R}^L_{t,:}$, allowing each task to retrieve complementary information from other tasks. Note that each row of the output is a weighted combination of all tasks' feature maps. A lightweight prediction head, a $3{\times}3$ convolutional block followed by a linear projection, is then applied to $\hat{\mathcal{Z}}_t$ to produce the final prediction for task t. Following prior work [69, 71], we select the output features at K evenly spaced encoder layers and apply the same prompting-based decoding process in Eq. 8 to each of them.

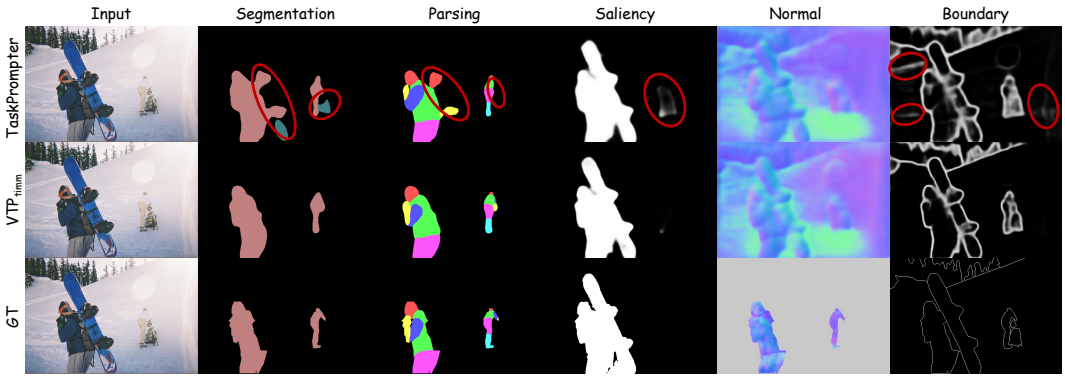


Figure 2: **Qualitative results on Pascal-Context [9].** VTP$_{\text{timm}}$ produces precise fine-grained object segmentation (bags), body-part parsing (human arms) with tighter saliency (suppress the spurious background). Key regions are with red circles. VTP$_{\text{vit-5}}$ and VTP$_{\text{dinov3}}$ and more results on Figure 5.

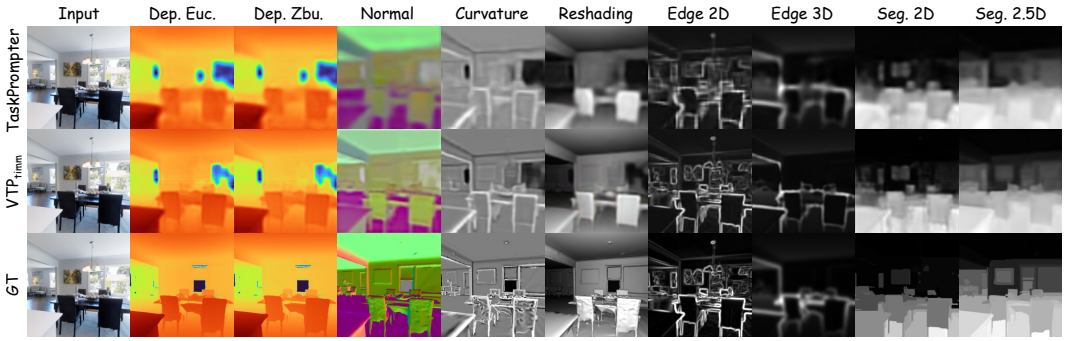


Figure 3: **Qualitative results on Taskonomy [77] with 9 dense tasks.** VTP$_{\text{timm}}$ produces apparently more accurate predictions than the baseline. VTP$_{\text{vit-5}}$ and VTP$_{\text{dinov3}}$ and more results on Figure 8.

3.3 Joint Training

VTP comprises three main components: (i) a standard ViT encoder $\mathcal{F}_l$ that processes both image and task tokens, (ii) the read-write matrices that maintains and routes virtual states as a bypass, and (iii) a streamlined decoder $\mathcal{D}_t$ with prediction heads. We combine per-task objectives into a single loss:

$$\mathcal{L}_{\text{MTL}} = \sum_{t=1}^{T} \lambda_t \cdot \mathcal{L}_t(\theta_s, \theta_t), \tag{9}$$

where θ_s denotes shared encoder parameters, θ_t the decoder parameters for task t, and λ_t scalar weights balancing each task's contribution. Since all task losses backpropagate through the shared encoder, VTP routing matrices Φ^l, Ψ^l, and Γ^l receive gradient signals from every task simultaneously, shaping the read–write pathways to reflect cross-task structure in an end-to-end learnable manner.

4 Experiments

We evaluate VTP on three MTL benchmarks, covering both indoor and outdoor scenarios: (i) Pascal-Context [9] (5 tasks, outdoor scenes), (ii) NYUD-v2 [51] (4 tasks, indoor scenes), and (iii) Taskonomy [77] (13 heterogeneous tasks, stress test). We then present ablations (Sec. 4.5), qualitative results (Figure 2), and empirical analysis of learned task vectors (Sec. 4.4) and efficiency (Sec. 4.5).

4.1 Datasets and Evaluation

Datasets. (i) **Pascal-Context** [9] contains 4,998/5,105 train/test images labeled for semantic segmentation (Semseg), human parsing (Parsing), and boundary detection (Edge). Following previous studies [40], we use pseudo labels for surface normal estimation (Normal) and saliency detection

Table 1: **State-of-the-art comparison** on NYUD-v2 [51] (left) and PASCAL-Context [9] (right). All Transformer-based models use a ViT-L backbone. VTP_{timm} is with ImageNet-pretrained weights, while $\text{VTP}_{\text{dinov3}}$ and $\text{VTP}_{\text{vit-5}}$ employ DINOv3 [52] and ViT-5 [57] without any modification. Δ_p (%) denotes the average MTL gains relative to TaskPrompter [71] (Δ_p=0). ↑ (↓) indicates higher (lower) values are better. The best and second-best results per task are in **bold** and underline, respectively.

Method	Semseg mIoU↑	Depth RMSE↓	Normal mErr↓	Edge odsF↑	Δ_p (%)↑
CNN-based					
Cross-Stitch [43]	36.34	0.6290	20.88	76.38	−17.94
PAP [79]	36.72	0.6178	20.82	76.42	−17.13
PSD [81]	36.69	0.6246	20.87	76.42	−17.54
PAD-Net [63]	36.61	0.6270	20.85	76.38	−17.68
MTI-Net [55]	45.97	0.5365	20.27	77.86	−7.80
ATRC [6]	46.33	0.5363	20.18	77.94	−7.48
Transformer-based					
InvPT [69]	53.56	0.5183	19.04	78.10	−1.74
InvPT++ [72]	53.85	0.5096	18.67	78.10	−0.69
TaskPrompter [71]	55.30	0.5152	18.47	78.20	0.00
TaskExpert [70]	55.35	0.5157	18.54	78.40	−0.03
MQTransformer [65]	54.84	0.5325	19.67	78.20	−2.67
TSP-Transformer [58]	55.39	0.4961	18.44	77.50	+0.78
MLoRE [67]	55.96	0.5076	18.33	78.43	+0.93
Other architectures					
MTMamba [31]	55.82	0.5066	18.63	78.70	+0.60
TaskDiffusion [68]	55.65	0.5020	18.43	78.64	+0.99
PAME [76]	56.63	0.5102	18.52	78.10	+0.74
VTP_{timm}	57.21	0.4960	18.41	78.43	+1.95
$\text{VTP}_{\text{vit-5}}$	56.58	0.4915	**18.18**	78.65	+2.27
$\text{VTP}_{\text{dinov3}}$	**59.55**	**0.4820**	18.52	**79.25**	**+3.80**

Method	Semseg mIoU↑	Parsing mIoU↑	Saliency maxF↑	Normal mErr↓	Edge odsF↑	Δ_p (%)↑
CNN-based						
ASTMT [40]	68.00	61.10	65.70	14.70	72.40	−11.69
PAD-Net [63]	53.60	59.60	65.80	15.30	72.50	−16.51
MTI-Net [55]	61.70	60.18	84.78	14.23	70.80	−8.76
ATRC [6]	62.69	59.42	84.70	14.20	70.96	−8.67
ATRC-ASPP [6]	63.60	60.23	83.91	14.30	70.86	−8.57
ATRC-BMTAS [6]	67.67	62.93	82.29	14.24	72.42	−6.65
Transformer-based						
InvPT [69]	79.03	67.61	84.81	14.15	73.00	−1.60
InvPT++ [72]	80.22	69.12	84.74	13.73	74.20	+0.06
TaskPrompter [71]	80.89	68.89	84.83	13.72	73.50	0.00
TaskExpert [70]	80.64	69.42	84.87	13.56	73.30	+0.28
MQTransformer [65]	78.93	67.41	83.58	14.21	73.90	−1.81
TSP-Transformer [58]	81.48	70.64	84.86	13.69	74.80	+1.06
MLoRE [67]	81.41	70.52	84.90	13.51	75.42	+1.45
Other architectures						
MTMamba [31]	81.11	72.62	84.14	14.14	**78.80**	+1.80
TaskDiffusion [68]	81.21	69.62	84.94	13.55	74.89	+0.94
PAME [76]	81.38	70.39	84.51	13.36	75.30	+1.50
VTP_{timm}	82.88	73.85	85.52	13.56	75.12	+2.77
$\text{VTP}_{\text{vit-5}}$	82.48	73.28	85.60	**13.32**	75.35	+2.94
$\text{VTP}_{\text{dinov3}}$	**85.62**	**79.25**	**85.96**	13.85	76.80	**+5.15**

Table 2: **Taskonomy results over 13 tasks.** Δ_p (%) denotes the average MTL performance gains relative to TaskPrompter [71] baseline (Δ_p=0). ↑ (↓) indicates the higher (lower) values are better. The best and second-best results for each task are highlighted in **bold** and underline, respectively.

Method	Obj.Cls. Acc↑	Scn.Cls. Acc↑	Dep.Euc. RMSE↓	Dep.Zbu. RMSE↓	Normal mErr↓	Curv. L2↓	Reshad. L1↓	Edge 2D L1↓	Edge 3D L1↓	Keyp. 2D L1↓	Keyp. 3D L1↓	Seg. 2D L1↓	Seg. 2.5D L1↓	Δ_p (%)↑
InvPT [69]	57.01	65.50	6.251	6.384	29.850	0.9120	0.1810	0.0985	0.0835	0.1150	0.0835	0.2035	0.1590	−2.34
TaskPrompter [71]	58.37	66.76	6.150	6.268	29.046	0.8947	0.1754	0.0961	**0.0819**	0.1107	0.0814	0.1991	0.1566	0.00
VTP_{timm}	58.78	**70.96**	5.757	5.823	**27.505**	0.8650	0.1750	0.0804	0.0948	0.0834	0.0805	**0.1927**	0.1570	+4.50
$\text{VTP}_{\text{vit-5}}$	**61.73**	69.90	5.460	**5.620**	33.060	0.8494	0.1703	0.0795	0.0862	**0.0710**	0.0796	0.2020	0.1608	+5.53
$\text{VTP}_{\text{dinov3}}$	60.21	68.08	**5.440**	5.630	28.356	**0.8288**	**0.1447**	**0.0683**	0.0859	0.0719	**0.0747**	0.1992	**0.1515**	**+9.58**

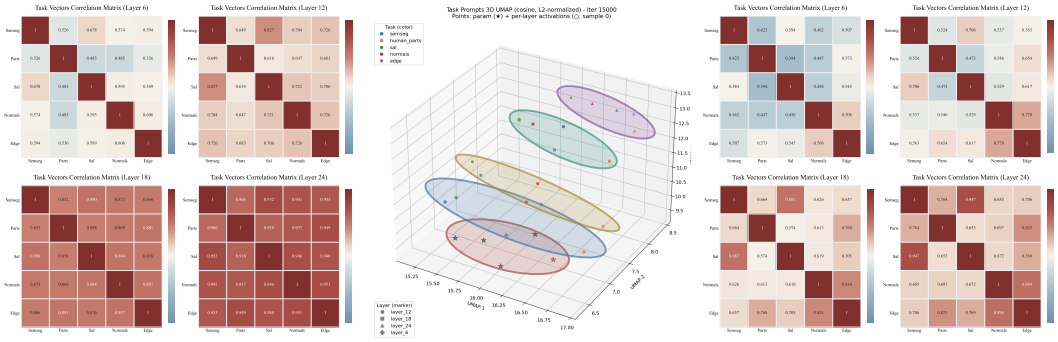
(Saliency), totaling 5 tasks. (ii) **NYUD-v2** [51] provides 1,449/795 train/test images with 4 tasks spanning semantics and geometry: Semseg, monocular depth estimation (Depth), Normal, and Edge. (iii) **Taskonomy** [77] comprises 381.8K images with 25/5 buildings for train/test for its tiny split, with 13 tasks: object and scene classification (Obj./Scn. Cls.), Euclidean and z-buffer depth (Dep. Euc./Zbu.), normal, curvature (Curv.), reshading (Reshad.), 2D/3D edge and keypoint (Edge/Keyp. 2D/3D), and semseg (Seg. 2D/2.5D). It is challenging for jointly training heterogeneous tasks.

Evaluation. Following established protocols [71, 69], we report standard per-task metrics: mIoU (Semseg, Parsing), mErr (Normal), maxF/odsF (Edge), Acc (classification), and L_1/L_2 distances (regression). Depth is tested with RMSE, absolute/relative error, and threshold (1.25, 1.25^2, 1.25^3). To summarize overall MTL performance, we report the popular average per-task relative gain Δ_p:

$$\Delta_p = \frac{1}{T}\sum_{t=1}^{T}(-1)^{\sigma_t}\frac{M_{m,t}-M_{b,t}}{M_{b,t}} \tag{10}$$

where $M_{m,t}$ and $M_{b,t}$ are performance of method m and the baseline on task t, and $\sigma_t \in \{0,1\}$ flips the sign for lower-is-better metrics (*e.g.*, mErr, L_1, L_2). Please refer to Appendix for more details.

Implementation Details. Following prior work [69, 71], all models are trained for $40{,}000$ iterations using . Adam with learning rate 2×10^{-5}, weight decay 1×10^{-6}, a batch size of 4 and a polynomial schedule. Regression tasks are trained with L_1 loss, while classification tasks (Semseg, Parsing, Saliency, Edge) using cross-entropy. Task vectors are initialized from $\mathcal{N}(\mu{=}1, \sigma{=}1)$. Φ^l and Γ^l are initialized as block identity matrices to ensure pass-through reading and full carry-forward, while Ψ^l is with a cyclic pattern ($\Psi^l_{i,j}{=}1$ if $i{=}j \bmod T$, else 0) so that each task writes back to its natural state, reducing to standard prompting for stability, with routing structure emerging during training.



(a) Layer-wise Correlation of TaskPrompter's Task Vectors (b) UMAP of VTP's Layer-wise Task Vectors (c) Layer-wise Correlation of VTP's Task Vectors

Figure 4: **Task correlation and UMAP analysis.** Pairwise cosine similarity among task vectors at selected layers for **(a)** TaskPrompter collapses into uniformly high correlation with negligible variation in deep layers. **(b)** UMAP of VTP's task vectors shows clear layer-wise clusters, confirming that vectors form an internally cohesive latent space encoding layer-wise discriminative information. **(c)** VTP exhibits structured task affinities, capturing adaptive inter-task differentiation across depths.

4.2 Pascal-Context and NYUD-v2 Results

Table 1 compares VTP against CNN-based, Transformer-based, and other architectures on Pascal-Context [9] and NYUD-v2 [51]. We report three VTP variants: VTP_{timm} with ImageNet-pretrained ViT, and $\text{VTP}_{\text{dinov3}}$ / $\text{VTP}_{\text{vit-5}}$ that integrate DINOv3 [52] and ViT-5 [57] without any modification.

VTP without VFMs. VTP_{timm} surpasses existing methods on Δ_p on both benchmarks. On Pascal-Context, it achieves $+2.77\%$ (previous best: $+1.80\%$), leading on segmentation (82.88 vs. 81.48) and parsing (73.85 vs. 72.62), while also improving saliency (85.52 maxF). On NYUD-v2, it reaches $+1.95\%$, nearly doubling the prior best of $+0.99\%$, with the highest segmentation (57.21 vs. 56.63) results. Boundary detection on Pascal-Context remains below MTMamba (78.80), likely due to the Mamba-based decoder that excels at fine-grained spatial details. Overall, these results confirm that VTP's adaptive prompting is effective on its own, independent of stronger pretrained vision encoders.

VTP with VFMs. VFM integration yields a substantial leap, showcasing the practical value of VTP's clean encoder. On Pascal-Context, $\text{VTP}_{\text{dinov3}}$ achieves $+5.15\%$, nearly doubling VTP_{timm}, with segmentation rising from 82.88 to 85.62 and parsing from 73.85 to 79.25.On NYUD-v2, it reaches $+3.80\%$, roughly $4\times$ the prior non-VTP best, leading on segmentation (59.55), depth (0.4820), and boundary (79.25). VFMs show diverse strengths: $\text{VTP}_{\text{vit-5}}$ achieves the best normal results (18.18, 13.32), while $\text{VTP}_{\text{dinov3}}$ dominates semantic tasks. Interestingly, $\text{VTP}_{\text{dinov3}}$ slightly degrades normal estimation compared to VTP_{timm} (13.85 vs. 13.56 on Pascal-Context; 18.52 vs. 18.41 on NYUD-v2), suggesting that DINOv3's self-supervised pretraining favors semantic over geometric representations.

4.3 Stress-Testing on Taskonomy

To examine VTP's robustness under extreme MTL conditions, we evaluate on Taskonomy [77] with 13 heterogeneous tasks, from sparse classification (object and scene) to dense regression (curvature, reshading). Due to limited compute and the absence of popular benchmarking, we train and compare against two baselines [69, 71]. The goal here is not to establish a new leaderboard, but to stress-test whether our VTP with different vision encoders scales gracefully as task count and diversity grow.

As shown in Table 2, VTP_{timm} achieves $+4.50\%$ over TaskPrompter, with notable gains appearing on scene classification (70.96 vs. 66.76), depth (5.757 vs. 6.150), normals (27.505 vs. 29.046), and 2D edges (0.0804 vs. 0.0961). $\text{VTP}_{\text{dinov3}}$ reaches $+9.58\%$ overall, more than doubling VTP_{timm}, with the best results on 7 of 13 tasks including depth, curvature, and reshading. $\text{VTP}_{\text{vit-5}}$ achieves $+5.53\%$ with a complementary profile, leading on object classification and 2D keypoints. Not all tasks benefit uniformly. VTP_{timm} regresses on 3D edges (0.0948 vs. 0.0819), and $\text{VTP}_{\text{vit-5}}$ degrades on normals (33.060 vs. 29.046) despite leading this on Pascal-Context and NYUD-v2, likely due to distributional differences in Taskonomy's annotations. Despite these variations, the consistent overall gains across 13 tasks of varying granularity show that VTP generalizes well beyond the common 4–5 task setting.

Table 3: **Ablation (left) and efficiency (right) results.** Left: Δ_{pascal} and Δ_{nyu} denote the MTL gains Δ_p (%) on Pascal-Context and NYUD-v2 relative to TaskPrompter [71]. V: virtual task number. VTPs share similar parameters and FLOPs. Throughput results are measured on a single L40 GPU.

Configuration	Δ_{pascal}	Δ_{nyu}
(a) Capacity controls		
TaskPrompter	0.00	0.00
+ 2× decoder parameters	−0.18	−0.07
+ 2× vectors per task	+0.09	+0.05
+ 4× vectors per task	−0.14	−0.02
(b) Virtual task number V		
VTP (V=0)	+1.52	+1.28
VTP (V=2, default)	**+2.77**	+1.95
VTP (V=4)	+2.41	**+1.97**
VTP (V=8)	+2.43	+1.84

Method	Δ_p (%)↑	Parameters (M)↓	FLOPs (G)↓	Throughput (imgs/s)↑
PAD-Net [63]	−16.51	**330**	773	7.684
MTI-Net [55]	−8.76	851	774	6.790
InvPT [69]	−1.60	423	669	10.813
MQTransformer [65]	−1.81	504	712	9.045
TaskExpert [70]	+0.28	420	622	11.526
TaskPrompter [71]	0.00	401	497	14.394
VTP_{timm}	+2.77	342	487	**16.318**
$\text{VTP}_{\text{vit-5}}$	+2.94	337	**487**	13.258
$\text{VTP}_{\text{dinov3}}$	**+5.15**	336	488	16.153

4.4 Qualitative results and analysis

Figure 2 compares VTP with TaskPrompter on Pascal-Context (more in Appendix D), where VTP_{timm} yields sharper boundaries and VFM variants further refine details in Figure 5. To understand how these task vectors evolve across layers, we analyze pairwise cosine similarity of task vectors and visualize them via UMAP [42] in Figure 4. Please refer to Appendix E for more setting details.

Mode collapse of the baseline. In Figure 4 (a), TaskPrompter's task vectors progressively over-align with depth. While shallow layers exhibit moderate, structured inter-task similarity, this erodes rapidly. The mean off-diagonal correlation reaches 0.87 with a min–max range of 0.04 by layer 18 and 0.94 by layer 24. This indicates that, without explicit learning pathways beyond pure backpropagation, task vectors degenerate into near-identical directions regardless of their specific semantic relationships.

Discriminative task patterns of VTP. In contrast, VTP exhibits structured, depth-related patterns (Figure 4(c)). Shallow layers correlate geometric tasks (normals, edges), while deeper layers increasingly align semantic pairs (segmentation and saliency), showing that VTP encodes discriminative features from the $T+V$ virtual state into T vectors across depths. Empirical analysis in Figure 4 (b) further corroborates this observation, where different layers establish their distinct UMAP clusters.

4.5 Ablations and Hyperparameter Sensitivity

Ablation Study. Table 3(a) isolates the gains from added capacity and VTP itself. Doubling decoder parameters slightly hurts Δ_p (-0.18%), and naively increasing vectors yields negligible or negative returns ($+0.09\%$ for $2\times$, -0.14% for $4\times$). In contrast, VTP (V=2) achieves $+2.77\%$, confirming that gains stem from the method more than increased capacity. Table 3(b) varies the virtual task V. Even V=0 offers $+1.52\%$, since it includes the full Eqs. 5–7, showing that layer-wise routing alone is crucial, consistent with the mode collapse in Figure 4. Adding virtual tasks contributes: V=2 yields $+0.97\%$ over V=0, while V=4 and V=8 perform comparably ($+2.41\%$, $+2.43\%$), showing that a small V suffices for capacity. We use V=2 as default. We conduct more validations in Table 5.

Efficiency. The complexity comprises three parts: (i) Self-attention over $T+N$ tokens ($\mathcal{O}((T+N)^2C)$, scaling with N); (ii) VTP routing on task stream ($\mathcal{O}((T+V)^2C)$, independent of N); and (iii) decoder affinities ($\mathcal{O}(T(N+T)C)$, linear in N). Since $T, V \ll N$, the routing cost is negligible relative to encoder when $T, V \ll N$. Table 3 (right) shows that VTP uses fewer parameters (342M vs. 401M) and FLOPs (487G vs. 497G) than TaskPrompter with higher throughput, as it removes extra encoder designs. Thus, VTP variants share similar costs, with differences stemming from the VFMs.

5 Conclusion and Discussion

Conclusion. We introduce Virtual Task Prompting (VTP), a prompting-based multi-task model that eliminates the need for modifications to the vision encoder. By preserving a clean backbone, VTP integrates well with off-the-shelf VFMs, further pushing the limits on Pascal-Context and NYUD-v2. Stress testing on Taskonomy with 13 diverse tasks also shows VTP's robustness and scalability.

Limitations and Future Work. Due to limited resources, we have not tried VTP at larger multi-modal foundation models like Qwen3-VL [3]. Exploring expressive prompting methods and scaling up modalities would be beneficial for applications like autonomous driving and embodied AI.

Acknowledgments and Disclosure of Funding

The work is supported in part by the Early Career Scheme of the Research Grants Council (RGC) of the Hong Kong SAR under grant No. 26202321, ITF PRP/046/24FX, SAIL Research Project, and HKUST-Zeekr Collaborative Research Fund.

References

[1] A. Agiza, M. Neseem, and S. Reda. MTLoRA: Low-rank adaptation approach for efficient multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[2] S. Baek, S. Lee, H. Jo, H. Choi, and D. Min. TADFormer: Task-adaptive dynamic transformer for efficient multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[3] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] H. Ban and K. Ji. Fair resource allocation in multi-task learning. In *International Conference on Machine Learning (ICML)*, 2024.

[5] D. Bhattacharjee, S. Süsstrunk, and M. Salzmann. Vision transformer adapters for generalizable multitask learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[6] D. Brüggemann, M. Kanakis, A. Obukhov, S. Georgoulis, and L. Van Gool. Exploring relational context for multi-task dense prediction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[7] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[8] D. S. Chaplot, L. Lee, R. Salakhutdinov, D. Parikh, and D. Batra. Embodied multimodal multitask learning. *arXiv preprint arXiv:1902.01385*, 2019.

[9] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[10] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, 2018.

[11] Z. Chen, Y. Shen, M. Ding, Z. Chen, H. Zhao, E. G. Learned-Miller, and C. Gan. Mod-Squad: Designing mixtures of experts as modular multi-task learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[12] S. Chennupati, G. Sistu, S. Yogamani, and S. A Rawashdeh. MultiNet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[13] S. Chowdhuri, T. Pankaj, and K. Zipser. MultiNet: Multi-modal multi-task learning for autonomous driving. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[14] J.-A. Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5):313–318, 2012.

[15] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[17] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. NDDR-CNN: Layerwise feature fusing in multi-task CNNs by neural discriminative dimensionality reduction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] Y. He, S. Zheng, Y. Tay, J. Gupta, Y. Du, V. Aribandi, Z. Zhao, Y. Li, Z. Chen, D. Metzler, et al. HyperPrompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning (ICML)*, 2022.

[19] R. Herzig, O. Abramovich, E. Ben Avraham, A. Arbelle, L. Karlinsky, A. Shamir, T. Darrell, and A. Globerson. PromptonomyViT: Multi-task prompt learning improves video transformers using synthetic scene data. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[21] R. Hu and A. Singh. UniT: Multimodal multitask learning with a unified transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[22] M. Huh, B. Cheung, T. Wang, and P. Isola. Position: The platonic representation hypothesis. In *International Conference on Machine Learning (ICML)*, 2024.

[23] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki. Multi-task learning with attention for end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[24] A. Javaloy and I. Valera. RotoGrad: Gradient homogenization in multitask learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[25] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022.

[26] F. Jiang, S. Wang, and X. Gong. Task-conditional adapter for multi-task dense prediction. In *ACM International Conference on Multimedia (ACM MM)*, 2024.

[27] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[29] B. Li, B. Wu, B. Ma, B. Xiao, C. Zhang, C. Li, C. Wang, C. Xu, C. Zhang, C. Hu, et al. Virtual width networks. *arXiv preprint arXiv:2511.11238*, 2025.

[30] X. Liang, M. Niu, J. Han, H. Xu, C. Xu, and X. Liang. Visual exemplar driven task-prompting for unified perception in autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[31] B. Lin, W. Jiang, P. Chen, Y. Zhang, S. Liu, and Y.-C. Chen. MTMamba: Enhancing multi-task dense scene understanding by mamba-based decoders. In *European Conference on Computer Vision (ECCV)*, 2024.

[32] B. Lin, F. Ye, Y. Zhang, and I. Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research (TMLR)*, 2022.

[33] B. Liu, Y. Feng, P. Stone, and Q. Liu. FAMO: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[35] L. Liu, Y. Li, Z. Kuang, J.-H. Xue, Y. Chen, W. Yang, Q. Liao, and W. Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[36] S. Liu, S. James, A. Davison, and E. Johns. Auto-Lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research (TMLR)*, 2022.

[37] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] Y.-C. Liu, C.-Y. Ma, J. Tian, Z. He, and Z. Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[39] Y. Lu, S. Sirejiding, Y. Ding, C. Wang, and H. Lu. Prompt guided transformer for multi-task dense prediction. *IEEE Transactions on Multimedia (TMM)*, 26:6375–6385, 2024.

[40] K.-K. Maninis, I. Radosavovic, and I. Kokkinos. Attentive single-tasking of multiple tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[41] K. S. I. Mantri, C.-B. Schönlieb, B. Ribeiro, C. Baskin, and M. Eliasof. DiTask: Multi-task fine-tuning with diffeomorphic transformations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[42] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[43] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[44] A. Navon, A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning (ICML)*, 2022.

[45] M. Oquab, T. Darcet, T. Moutakanni, H. Q. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. B. Huang, S.-W. Li, I. Misra, M. G. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

[46] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Latent multi-task architecture learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[47] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[48] D. Senushkin, N. Patakin, A. Kuznetsov, and A. Konushin. Independent component alignment for multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[49] J. Shen, C. Wang, Z. Xiao, N. Van Noord, and M. Worring. GO4Align: Group optimization for multi-task alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[50] S. Shen, S. Yang, T. Zhang, B. Zhai, J. E. Gonzalez, K. Keutzer, and T. Darrell. Multitask vision-language prompt tuning. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[51] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision (ECCV)*, 2012.

[52] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

[53] X. Sun, R. Panda, R. Feris, and K. Saenko. AdaShare: Learning what to share for efficient deep multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[54] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.

[55] S. Vandenhende, S. Georgoulis, and L. Van Gool. MTI-Net: Multi-scale task interaction networks for multi-task learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[57] F. Wang, S. Ren, T. Zhang, P. Neskovic, A. Bhattad, C. Xie, and A. Yuille. ViT-5: Vision transformers for the mid-2020s. *arXiv preprint arXiv:2602.08071*, 2026.

[58] S. Wang, J. Li, Z. Zhao, D. Lian, B. Huang, X. Wang, Z. Li, and S. Gao. TSP-Transformer: Task-specific prompts boosted transformer for holistic scene understanding. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[59] Y. Wang, Z. Wang, Y. Wu, Y. Ma, and D. Xu. Care-edit: Condition-aware routing of experts for contextual image editing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[60] Z. Wang, S. Li, and D. Xu. Rep-MTL: Unleashing the power of representation-level task saliency for multi-task learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[61] Z. Xie, Y. Wei, H. Cao, C. Zhao, C. Deng, J. Li, D. Dai, H. Gao, J. Chang, K. Yu, et al. mHC: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

[62] Y. Xin, J. Du, Q. Wang, Z. Lin, and K. Yan. VMT-Adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[63] D. Xu, W. Ouyang, X. Wang, and N. Sebe. PAD-Net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[64] X. Xu, H. Zhao, V. Vineet, S.-N. Lim, and A. Torralba. MTFormer: Multi-task learning via transformer and cross-task reasoning. In *European Conference on Computer Vision (ECCV)*, 2022.

[65] Y. Xu, X. Li, H. Yuan, Y. Yang, and L. Zhang. Multi-task learning with multi-query transformer for dense prediction. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 34(2):1228–1240, 2023.

[66] Y. Xu, Y. Yang, and L. Zhang. DeMT: Deformable mixer transformer for multi-task learning of dense prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[67] Y. Yang, P.-T. Jiang, Q. Hou, H. Zhang, J. Chen, and B. Li. Multi-task dense prediction via mixture of low-rank experts. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[68] Y. Yang, P.-T. Jiang, Q. Hou, H. Zhang, J. Chen, and B. Li. Multi-task dense predictions via unleashing the power of diffusion. In *International Conference on Learning Representations (ICLR)*, 2025.

[69] H. Ye and D. Xu. Inverted pyramid multi-task transformer for dense scene understanding. In *European Conference on Computer Vision (ECCV)*, 2022.

[70] H. Ye and D. Xu. TaskExpert: Dynamically assembling multi-task representations with memorial mixture-of-experts. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[71] H. Ye and D. Xu. TaskPrompter: Spatial-channel multi-task prompting for dense scene understanding. In *International Conference on Learning Representations (ICLR)*, 2023.

[72] H. Ye and D. Xu. InvPT++: Inverted pyramid multi-task transformer for visual scene understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 46(12):7493–7508, 2024.

[73] S. Yoo, E. Kim, D. Jung, J. Lee, and S. Yoon. Improving visual prompt tuning for self-supervised vision transformers. In *International Conference on Machine Learning (ICML)*, 2023.

[74] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[75] X. Yu, C. Zhou, Y. Fang, and X. Zhang. MultiGPrompt for multi-task pre-training and prompting on graphs. In *ACM Web Conference (WWW)*, 2024.

[76] X. Yu, Y. Zhuge, S. Gong, L. Zhang, P. Zhang, and H. Lu. Parameter-aware mamba model for multitask dense prediction. *IEEE Transactions on Cybernetics (TCYB)*, 2026.

[77] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[78] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 34(12):5586–5609, 2022.

[79] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[80] H. Zhong, J. Chen, Y. Zhang, D. Huang, and Y. Wang. Transforming vision transformer: Towards efficient multi-task asynchronous learner. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[81] L. Zhou, Z. Cui, C. Xu, Z. Zhang, C. Wang, T. Zhang, and J. Yang. Pattern-structure diffusion for multi-task learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[82] D. Zhu, H. Huang, Z. Huang, Y. Zeng, Y. Mao, B. Wu, Q. Min, and X. Zhou. Hyper-connections. In *International Conference on Learning Representations (ICLR)*, 2025.

A Appendix

This appendix provides additional implementation details, quantitative results, qualitative visualizations, and further discussions that complement the main text. It is organized as follows:

In Appendix B, we provide the dataset and implementation specifications for Pascal-Context [9], NYUD-v2 [51], and Taskonomy [77], covering task definitions, evaluation metrics, data augmentation, optimization schedules, loss functions, task balancing weights, and hyperparameter settings.

In Appendix C, we present the complete results on Taskonomy [77], reporting the full suite of depth estimation metrics (RMSE, absolute error, relative error, and threshold accuracies δ_1, δ_2, δ_3) for both Euclidean and z-buffer depth, which were omitted from the main text due to the space limitations.

In Appendix D, we provide extended qualitative comparisons beyond Figures 2, highlighting challenging scenarios such as thin structures, articulated poses, and crowded multi-person layouts, as well as one set of Taskonomy [77] predictions spanning nine typical dense prediction tasks.

In Appendix E, we present extended activation correlation analysis and UMAP [42] visualizations of the learned task embeddings. We show that the prompts in TaskPrompter exhibit a mode collapse at deep layers, while VTP's prompts form clearly separated, task-discriminative clusters in UMAP subspace and demonstrate progressive specialization across different network layers.

In Appendix F, we formalize VTP's flexibility and drop-in compatibility with off-the-shelf VFMs, and compare these properties against previous model architectures for multi-task scene understanding.

B Dataset and Implementation Details

In this section, we discuss the task definitions, evaluation metrics, and training protocols for our three evaluation benchmarks, Pascal-Context, NYUD-v2, and Taskonomy. These implementation details complement the overview in Sec. 4.1 of the main text and are intended to facilitate reproducibility.

B.1 Pascal-Context

Pascal-Context [9] extends the PASCAL VOC 2010 dataset with dense per-pixel annotations for the entire scene, covering both indoor and outdoor imagery. Following the standard protocol from previous studies [69, 71, 72], we use 4,998 images for training and 5,105 images for testing. The five tasks jointly trained on this benchmark are: **Semantic Segmentation (Semseg):** 21-class per-pixel labeling (20 object categories plus background). Evaluated with mean Intersection over Union (mIoU, %). **Human Parsing (Parsing):** 7-class dense segmentation of human body parts, including head, torso, upper arms, lower arms, upper legs, and lower legs. Evaluated with mIoU (%). **Saliency Detection (Saliency):** Binary per-pixel saliency estimation using the pseudo-labels generated by [40]. Evaluated with the maximal F-measure (maxF). **Surface Normal Estimation (Normal):** Per-pixel prediction of 3D surface orientation vectors. Following [40], we use the pseudo-labels produced by a pretrained normal estimator. Evaluated with mean angular error (mErr, in degrees; lower is better). **Boundary Detection (Edge or Boundary):** Binary edge detection using the DistillBDCN [40] annotations. Evaluated with the optimal-dataset-scale F-measure (odsF).

All images are resized to 512×512 during training and testing, consistent with prior work [69, 71]. Data augmentation includes random horizontal flipping, random scaling (scale factor sampled uniformly from $[0.5, 2.0]$), and random cropping. No test-time augmentation is applied. For the training configuration on Pascal-Context, we use the Adam optimizer with an initial learning rate of 2×10^{-5} and weight decay of 1×10^{-6}. A polynomial learning rate schedule with power 0.9 is applied over 40,000 iterations with a batch size of 4. The task-balancing weights λ_t in Eq. 9 are set the same as previous studies [71, 60]. For VTP-specific modules, we set the default number of virtual tasks $V = 2$ (see Table 3 in the main text for sensitivity analysis). All learnable task vector embeddings, including the virtual ones, are randomly initialized from a normal distribution with $\mathcal{N}(\mu{=}1, \sigma{=}1)$. Φ^l and Γ^l. The VTP read-write routing matrices Φ^l and Γ^l are initialized as block identity matrices to ensure pass-through reading and full carry-forward, while Ψ^l is with a cyclic pattern ($\Psi^l_{i,j}{=}1$ if $i{=}j \bmod T$, else 0) so that each task writes back to its natural state, reducing to standard prompting for stability, with routing structure emerging during training.

B.2 NYUD-v2

NYUD-v2 [51] is a widely used indoor scene understanding benchmark consisting of RGB-D images captured with a Microsoft Kinect sensor. Following the standard protocol [69, 71], we use 1,449 images for training and 795 images for testing. The four tasks are: **Semantic Segmentation (Semseg):** 40-class per-pixel labeling of indoor scene categories. Evaluated with mIoU (%). **Monocular Depth Estimation (Depth):** Per-pixel prediction of absolute depth from a single RGB input. Evaluated with root mean squared error (RMSE; lower is better). **Surface Normal Estimation (Normal):** Per-pixel prediction of 3D surface normals. Evaluated with mean angular error (mErr, in degrees; lower is better). **Boundary Detection (Edge or Boundary):** Object boundary detection. Evaluated with the optimal-dataset-scale F-measure (odsF).

Images are resized to 512×512 during both training and testing. The data augmentation strategy is identical to Pascal-Context: random horizontal flipping, random scaling with a scale factor in $[0.5, 2.0]$, and random cropping. No test-time augmentation is used. The training setup on NYUD-v2 mirrors Pascal-Context in most respects: Adam optimizer with learning rate 2×10^{-5}, weight decay 1×10^{-6}, polynomial schedule with power 0.9, 40,000 iterations, and batch size 4. The task-balancing weights λ_t in Eq. 9 are set the same as previous studies [71, 60]. All VTP hyperparameters ($V = 2$, VTP read-write matrices initialization, and multi-scale extraction at $K = 4$ blocks) remain identical to Pascal-Context. The prompts are randomly initialized with normal distribution. All experiments are implemented with PyTorch and run on NVIDIA L40-48G GPUs.

B.3 Taskonomy

Taskonomy [77] is a large-scale indoor dataset originally comprising approximately 4.5 million images from over 500 buildings, annotated for 26 diverse tasks. To make experimentation tractable under our computational constraints, we adopt the official *tiny* split, which contains 381.8K images sampled from 35 buildings (25 for training and 5 for testing, with the remaining 5 reserved for validation). From the full set of 26 tasks, we select 13 representative ones that span a broad spectrum of visual understanding: **Classification tasks (sparse supervision):** *Object Classification (Obj. Cls.):* Image-level prediction of the dominant object category. Evaluated with top-1 accuracy (Acc, %). *Scene Classification (Scn. Cls.):* Image-level prediction of the scene type (*e.g.*, bedroom, kitchen, office). Evaluated with top-1 accuracy (Acc, %). **Regression tasks (dense supervision):** *Euclidean Depth (Dep. Euc.):* Per-pixel Euclidean distance from the camera. Evaluated with RMSE (lower is better). *Z-buffer Depth (Dep. Zbu.):* Per-pixel depth along the camera's optical axis. Evaluated with RMSE (lower is better). *Surface Normals (Normal):* Per-pixel 3D surface orientation. Evaluated with mean angular error (mErr; lower is better). *Curvature (Curv.):* Per-pixel principal curvature estimation. Evaluated with L2 distance (lower is better). *Reshading (Reshad.):* Per-pixel re-rendering under novel illumination. Evaluated with L1 distance (lower is better). *2D Edge Detection (Edge 2D):* Binary edge maps from RGB cues. Evaluated with L1 distance (lower is better). *3D Edge Detection (Edge 3D):* Occluding contour detection from geometric cues. Evaluated with L1 distance (lower is better). *2D Keypoint Detection (Keyp. 2D):* Per-pixel keypoint heatmap estimation. Evaluated with L1 distance (lower is better). *3D Keypoint Detection (Keyp. 3D):* Geometric keypoint detection in 3D. Evaluated with L1 distance (lower is better). *2D Segmentation (Seg. 2D):* Unsupervised 2D region segmentation. Evaluated with L1 distance (lower is better). *2.5D Segmentation (Seg. 2.5D):* Unsupervised segmentation incorporating depth cues. Evaluated with L1 distance (lower is better).

This selection is deliberately diverse: it includes both image-level classification and pixel-level dense prediction, spans semantic, geometric, and low-level structural understanding, and encompasses tasks with markedly different output distributions and gradient magnitudes. This diversity makes Taskonomy an ideal stress test for multi-task architectures, as the risk of negative transfer grows substantially with the number and heterogeneity of tasks.

All images are resized to 224×224 during training and testing, following the resolution convention of the original Taskonomy benchmark and previous studies [77, 11]. Data augmentation consists of random horizontal flipping and random color jittering (brightness, contrast, saturation, and hue), without random scaling or cropping due to the already moderate resolution. Our training uses the Adam optimizer with an initial learning rate of 1×10^{-5} (lower than the other benchmarks to accommodate the larger dataset and greater task diversity), weight decay of 1×10^{-6}, and a polynomial learning rate schedule with power 0.9 over 80,000 iterations (doubled relative to Pascal-Context and NYUD-v2 to allow sufficient convergence across 13 tasks) with a batch size of 8 due

to the limited computational resource. Classification tasks use cross-entropy loss, while all dense regression tasks use L1 loss. The task-balancing weights λ_t are set uniformly to 1.0 for all 13 tasks for impartial comparisons. All VTP hyperparameters and VTP read-write matrices initialization remain unchanged, with multi-scale features extracted at $K = 4$ evenly spaced blocks. All experiments are implemented with PyTorch and run on NVIDIA L40-48G GPUs.

C Full Quantitative Results on Taskonomy

As discussed in Section 4.1, depth estimation is evaluated with a richer set of metrics beyond RMSE, including absolute error (Abs), relative error (Rel), and threshold accuracy ($\delta_1 = 1.25$, $\delta_2 = 1.25^2$, $\delta_3 = 1.25^3$). While Table 2 of the main text reports only RMSE for conciseness, we present the full depth evaluation for both Euclidean depth and z-buffer depth on Taskonomy in Table 4.

Table 4: **Full depth estimation metrics on Taskonomy.** We report RMSE, absolute error (Abs), relative error (Rel), and threshold accuracies (δ_1, δ_2, δ_3) for both Euclidean depth and z-buffer depth. $\downarrow$ indicates lower is better; $\uparrow$ indicates higher is better. Best and second-best results are marked in **bold** and underline, respectively.

	Euclidean Depth (Dep. Euc.)						**Z-buffer Depth (Dep. Zbu.)**					
Method	RMSE↓	Abs↓	Rel↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	RMSE↓	Abs↓	Rel↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
InvPT [69]	6.251	4.873	0.412	34.52	60.18	75.24	6.384	4.965	0.420	33.87	59.43	74.61
TaskPrompter [71]	6.150	4.784	0.398	36.10	61.85	76.43	6.268	4.871	0.406	35.42	61.06	75.78
VTP_{timm}	5.757	4.402	0.353	39.85	65.23	79.16	5.823	4.456	0.374	39.17	64.52	78.48
$\text{VTP}_{\text{vit-5}}$	5.460	4.118	0.358	42.73	67.95	81.30	5.620	4.262	0.352	41.55	66.88	80.42
$\text{VTP}_{\text{dinov3}}$	**5.440**	**4.093**	**0.337**	**43.15**	**68.40**	**81.72**	**5.630**	**4.240**	**0.348**	**41.92**	**67.25**	**80.85**

Several observations are worth noting. First, VTP consistently outperforms both InvPT and TaskPrompter across all depth metrics on both depth variants. The improvements are not confined to a single error measure: VTP reduces RMSE, absolute error, and relative error simultaneously, while substantially boosting all three threshold accuracies. This indicates that the gains are not artifacts of a particular evaluation criterion but reflect genuinely improved depth predictions across the full error distribution.

Second, VFM variants ($\text{VTP}_{\text{vit-5}}$ and $\text{VTP}_{\text{dinov3}}$) offer large improvements on the threshold accuracy metrics. For instance, on Euclidean depth, $\text{VTP}_{\text{dinov3}}$ improves δ_1 from 36.10% (TaskPrompter) to 43.15%, a gain of 7.05 absolute points. This suggests that VFM features are especially beneficial for reducing large outlier errors, as the threshold metrics are sensitive to the fraction of pixels with predictions within a fixed multiplicative factor of the ground truth. Third, the two VFMs exhibit a similar pattern to what we observed in the main text: $\text{VTP}_{\text{dinov3}}$ achieves the best overall depth performance on most metrics, while $\text{VTP}_{\text{vit-5}}$ is competitive and occasionally matches or exceeds DINOv3 on specific measures (*e.g.*, z-buffer Rel). This supports the finding that different VFMs bring complementary strengths, and VTP can leverage each of them as a plug-and-play backbone.

D Additional Qualitative Comparisons

To complement the qualitative comparison in Figure 2 of the main manuscript, we present extra qualitative results that further illustrate VTP's advantages over the previous multi-task prompting method TaskPrompter [71] across different scenes, tasks, and difficulty levels. We first provide the full version of Figure 2 for apple-to-apple comparison in Figure 5. Set 1 in Figure 6 focuses on multi-person scenes where inter-individual occlusion and proximity create severe ambiguity for parsing and segmentation. Set 2 in Figure 6 targets a scene with extremely small and thin semantic objects alongside dense, complicated edge structures. The Taskonomy visualization in Figure 8 demonstrates VTP's effectiveness in the extreme MTL setting, emphasizing the progressive quality improvement across VTP variants requiring no additional method integration efforts.

D.1 Additional Pascal-Context Results (Set 1)

Figure 6 shows predictions on two additional Pascal-Context scenes, both featuring multiple human subjects—a scenario that is particularly demanding for multi-task scene understanding models

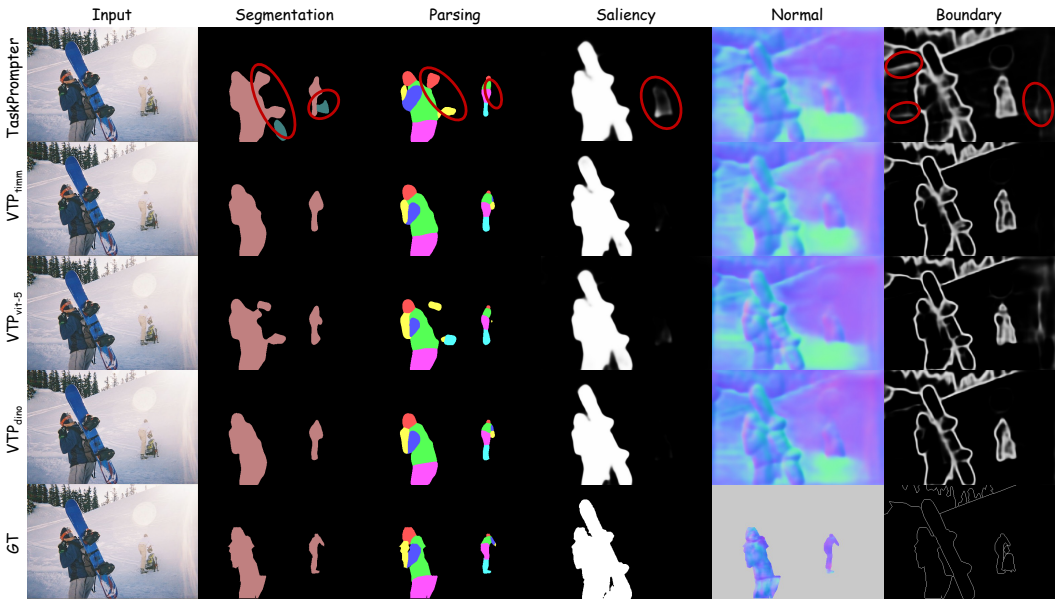


Figure 5: **Qualitative results on Pascal-Context [9].** VTP$_{timm}$ produces precise fine-grained object segmentation (bags), body-part parsing (human arms) with tighter saliency (suppress the spurious background). VTP$_{vit-5}$ and VTP$_{dinov3}$ further sharpen the details. Key regions are with red circles.

because it requires simultaneously resolving per-person semantic segmentation, fine-grained body-part parsing, instance-aware saliency, geometrically consistent surface normals, and precise boundary delineation across overlapping and mutually occluding individuals.

The left scene depicts a close-up outdoor portrait with a man holding an infant against a complex natural background. As highlighted by the red circles, TaskPrompter produces segmentation errors at the boundary between the person and the background, where semantic categories bleed across the contour. Its parsing output also exhibits fragmented body-part assignments with ambiguous region boundaries. In contrast, VTP$_{timm}$ already delivers noticeably cleaner segmentation and saliency masks with tighter adherence to the true object silhouette. TaskPrompter over-highlights background clutter near the person, while VTP concentrates its saliency response more precisely on the foreground subject. Moving to VTP$_{vit-5}$ and VTP$_{dinov3}$, the predictions progressively sharpen, with DINOv3 [52]'s strong semantic representations yielding parsing maps that faithfully delineate individual body parts with minimal inter-part confusion.

The right scene also presents a hard case: a group of multiple people walking together in a crowded outdoor setting, where inter-person occlusion and similar appearance create substantial ambiguity. TaskPrompter struggles most visibly here in the parsing task—the body-part labels for adjacent and partially overlapping individuals become entangled, with limb assignments frequently bleeding across person boundaries (see red circles). The segmentation output also shows confusion at the interfaces between closely spaced people. As shown in Figure 6, VTP handles this multi-person scenario markedly better: even VTP$_{timm}$ produces more coherent per-person region assignments, and the VFM-enhanced variants further resolve the fine-grained inter-person boundaries. The normal estimation is another notable differentiator: VTP generates spatially smooth and geometrically plausible normal fields over each person's body surface, whereas TaskPrompter's normals exhibit noisy, blurred, locally inconsistent orientations at precisely the regions where multiple people overlap.

These two multi-person scenes confirm that VTP provides the representational capacity needed to jointly resolve instance-level ambiguities across tasks, where the single-prompt-per-task baseline is fundamentally capacity-limited.

D.2 Additional Pascal-Context Results (Set 2)

Figure 6 presents a particularly challenging scene on Pascal-Context [9] that stress-tests multi-task models along two orthogonal axes simultaneously: (i) extremely small and thin semantic objects that

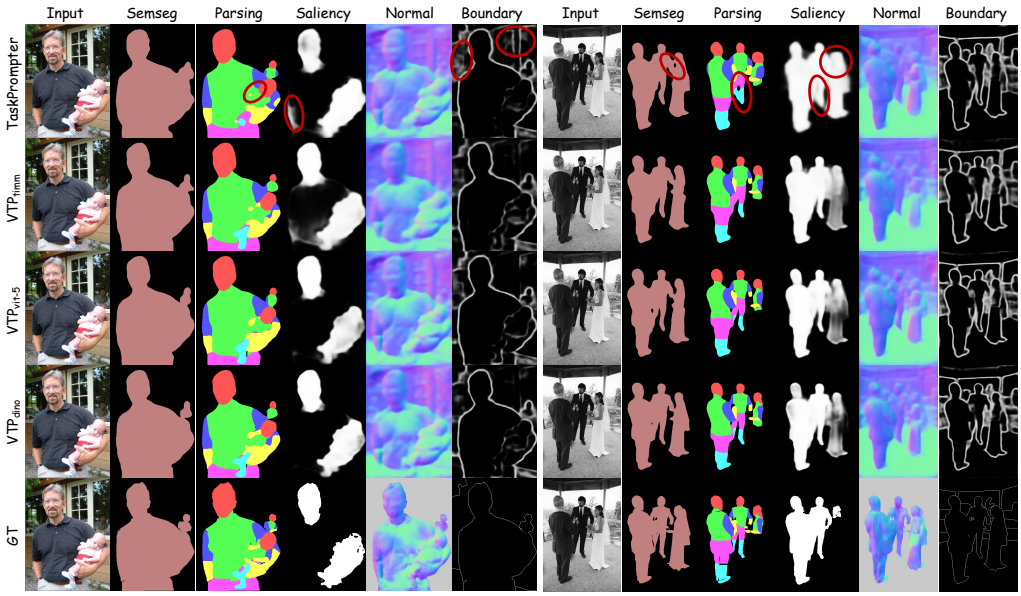


Figure 6: **Additional qualitative results on Pascal-Context (Set 1).** Two multi-person outdoor scenes that stress-test inter-individual disambiguation. **Left:** a man holding an infant against a cluttered natural background. TaskPrompter [71] (red circles) lets semantic labels bleed across the person with background contour in segmentation, fragments body-part assignments at the overlapping arms in parsing, and over-highlights background clutter in boundary. **Right:** a crowded group where closely spaced individuals share similar appearance. TaskPrompter's parsing labels entangle across adjacent persons, with leg assignments bleeding from one individual into the next. In both scenes, VTP$_{timm}$ recovers coherent per-person segmentation masks, disentangled body-part labels, and tighter saliency focus. VTP$_{vit-5}$ and VTP$_{dinov3}$ [57, 52] further resolve fine-grained inter-person boundaries, saliency, and produce geometrically smoother surface normals, approaching the ground-truth.

occupy only a handful of pixels, and (ii) dense, complicated edge structures that arise from the rich spatial layout of the scene. This combination is especially demanding as the model must preserve fine-grained spatial detail for accurate segmentation and parsing of tiny objects while also producing dense, high-fidelity boundary maps across the entire image.

The scene depicts a road with distant structures, lampposts, poles, a small human object, and other slender objects set against a backdrop of buildings and sky. These thin structures pose a severe challenge for segmentation: a single misclassified pixel can cause an entire pole or post to vanish from the prediction, or merge with the surrounding background class. As highlighted by the red circles, TaskPrompter drops these small objects altogether—tiny human objects are either entirely absent from its segmentation and parsing maps. VTP$_{timm}$, even without VFM features, already begins to recover these thin objects. This improvement suggests that the richer prompt representations afforded by the virtual tasks provide the discriminative power necessary to preserve spatial detail that a single static prompt vector simply cannot encode. The boundary detection task reveals an equally striking contrast. The ground truth for this scene contains a dense web of edges arising from road markings, building contours, poles, and background textures—a setting where both precision and recall are critical. VTP generates thin and accurate edge maps that capture both the macro-scale contours of large objects and the micro-scale boundaries of thin structures. The saliency predictions further corroborate this: VTP's maps exhibit tighter, more precisely localized responses on the salient foreground objects, whereas TaskPrompter's saliency bleeds into irrelevant background regions.

Across all five tasks, the progressive improvement from VTP$_{timm}$ through VTP$_{vit-5}$ to VTP$_{dinov3}$ is clearly visible, with VFM features amplifying the gains on the most challenging thin structures and tiny objects. This shows that VTP is not merely beneficial for common-case scenarios but is especially advantageous in the long tail of difficult real-world scenes where spatial precision is paramount.

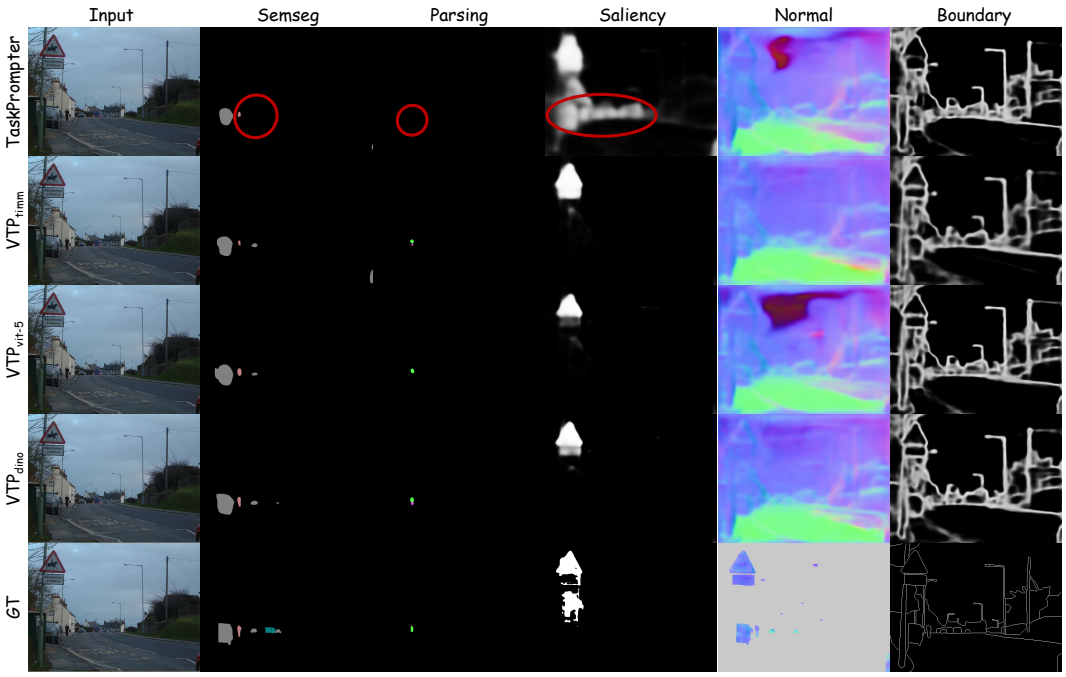


Figure 7: **Additional qualitative results (Set 2).** A challenging street scene that stress-tests two orthogonal challenges simultaneously: extremely small and thin semantic objects (distant poles, a tiny person occupying only a handful of pixels) and dense edge structures from road markings, building contours, and lampposts. TaskPrompter [71] (red circles) drops these tiny objects entirely from its segmentation and parsing maps, and produces a large spurious saliency blob on the background sky. VTP$_{\text{timm}}$ begins to recover the thin structures in segmentation and parsing while suppressing the false saliency response. VTP$_{\text{vit-5}}$ and VTP$_{\text{dinov3}}$ further sharpen the objects (the blue and green masks in Semseg) and produce boundaries that capture both macro building contours and micro pole edges.

D.3 Taskonomy Qualitative Results

Figure 8 provides qualitative comparisons on Taskonomy [77], which represents one of the most extreme multi-task learning settings in the literature: 13 heterogeneous tasks trained simultaneously, spanning image-level classification, pixel-level geometric estimation, and low-level structural detection. This extreme task diversity creates intense competition for model capacity, making it exceptionally difficult for any single architecture to produce high-quality predictions across all tasks without severe negative transfer. We display 9 of the 13 dense prediction tasks: Euclidean depth (Dep. Euc.), z-buffer depth (Dep. Zbu.), surface normals (Normal), curvature (Curv.), reshading (Reshad.), 2D edges (Edge 2D), 3D edges (Edge 3D), 2D segmentation (Seg. 2D), and 2.5D segmentation (Seg. 2.5D). These tasks collectively span the full spectrum from geometric depth and surface estimation to low-level boundary detection and unsupervised region grouping.

The indoor scene in Figure 8 features a kitchen workspace with furniture, appliances, and objects at varying depths—a setting that demands accurate geometric reasoning across multiple spatial scales. Reading the figure from top to bottom, a clear trend of progressive improvement is immediately visible: TaskPrompter (top row) produces the coarsest predictions, while VTP$_{\text{timm}}$, VTP$_{\text{vit-5}}$, and VTP$_{\text{dinov3}}$ each successively refine the outputs, converging toward the ground truth (bottom row). Crucially, this progressive improvement comes at no extra computational cost or model integration effort as shown in Table 3, all VTP variants share nearly identical parameters and FLOPs, and the VFM integration requires only swapping the backbone pretrained weights.

For the two depth tasks, TaskPrompter tends to produce over-smoothed depth maps that lose the sharp transitions at furniture edges and wall–floor boundaries. The depth discontinuities between the foreground objects and the background are visibly blurred, making it difficult to recover the true 3D layout of the scene. VTP$_{\text{timm}}$ already preserves these depth boundaries more faithfully. VTP$_{\text{dinov3}}$ further sharpens the depth predictions, producing maps where the relative ordering and absolute

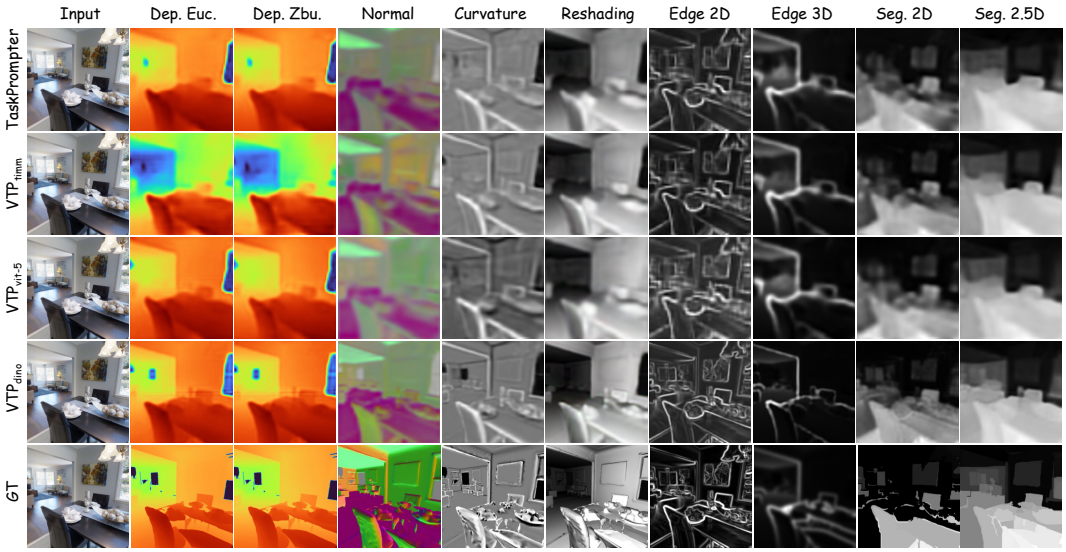


Figure 8: **Qualitative results on Taskonomy [77].** An indoor scene with 9 dense prediction tasks under the extreme 13-task joint training regime. From top to bottom, predictions become progressively finer and more accurate. TaskPrompter [71] produces over-smoothed depth maps, flat curvature, and imprecise edges, while VTP$_{timm}$ through VTP$_{dinov3}$ successively recover depth discontinuities, surface details, and region coherence, converging toward the ground truth. This progressive improvement requires no additional computational cost. Only the backbone weights are swapped among VFMs.

distances of objects are in close agreement with the ground truth for both Euclidean and z-buffer formulations.

The curvature and reshading tasks are revealing, as they require the model to capture subtle surface geometry that is easily overwhelmed in a 13-task setting where gradient signals from diverse tasks compete for shared representation space. TaskPrompter's curvature maps appear flat and lack the fine surface detail visible in the ground truth, while its reshading output misses the nuanced illumination variations on curved surfaces (*e.g.*, the contours of appliances and cabinet edges). VTP progressively recovers these details: VTP$_{timm}$ reintroduces coarse surface variation, and VTP$_{dinov3}$ further captures the fine-grained curvature and shading gradients that bring the predictions closer to ground-truth.

For edge detection (both 2D and 3D), the contrast between methods is stark. TaskPrompter's edges are often over-smoothed in heavily textured regions while simultaneously missing true geometric boundaries, resulting in noisy and spatially imprecise edge maps. VTP produces substantially cleaner outputs: the edges are thinner, better localized, and more faithfully aligned with the ground-truth contours. This improvement is particularly pronounced for 3D edges, which rely on geometric rather than photometric cues and thus benefit directly from the richer task-specific representations that VTP's virtual prompts provide.

The 2D and 2.5D segmentation tasks complete the picture. TaskPrompter's segmentation maps tend to fragment large planar surfaces into incoherent sub-regions and merge distinct objects that happen to share similar appearance. VTP generates more spatially coherent regions with boundaries that follow the actual object contours, and VTP$_{dinov3}$ further improves region purity by leveraging the strong semantic features from DINOv3. The 2.5D segmentation, which incorporates depth cues, shows a consistent advantage for VTP across all variants, confirming that the improved depth representations propagate benefits to downstream tasks that depend on geometric understanding.

Taken together, these results demonstrate that VTP's adaptive prompting mechanism scales gracefully to the extreme 13-task regime, consistently improving prediction quality across tasks of vastly different natures—from geometric depth to photometric reshading to structural edge detection—without any further task-specific architectural tuning or additional computational overhead.

Table 5: **Comparisons to further ablating VTP's effectiveness and contribution on Pascal-Context [9].** We compare VTP against (1) standard PEFT (LoRA) on a frozen ViT, (2) HPS with DINOv3 and task-specific heads, (3) TaskPrompter forcibly adapted to DINOv3, (4) multi-task PEFT methods (Polyhistor, VMT-Adapter, MTLoRA) with their original frozen-backbone setting, (5) the same multi-task PEFT methods extended with a fully tunable ViT-L backbone, and (6) our VTP_{timm}. Δ_p is computed relative to TaskPrompter [71]. The results demonstrate that the effectiveness of VTP is attributed more to its prompting design, rather than the VFMs or the added capacity. Although it shares similar technical formation with those PEFT methods, their goals are different. VTP as a multi-task model architecture jointly trains the base model and the VTP modules, surpassing PEFT methods on both fully and partially fine-tune scenarios.

Method	**Backbone Setup**	**Semseg** mIoU↑	**Parsing** mIoU↑	**Saliency** maxF↑	**Normal** mErr↓	**Edge** odsF↑	Δ_p (%)↑
(1) Standard PEFT on Frozen ViT							
LoRA [20] (r=16)	ViT-L, frozen + LoRA	74.62	63.18	82.45	14.85	71.20	−6.84
LoRA [20] (r=64)	ViT-L, frozen + LoRA	76.18	64.92	83.10	14.52	71.85	−5.21
(2) Hard Parameter Sharing with VFM							
DINOv3 [52] + HPS heads	ViT-L, fully tunable + per-task heads	81.15	70.35	84.80	13.92	73.65	+0.23
(3) TaskPrompter Forcibly Adapted to DINOv3							
TaskPrompter [71] (DINOv3-init)	ViT-L, weight-mismatched init	80.45	69.74	84.78	13.82	73.65	+0.27
(4) Multi-Task PEFT with Frozen Backbone (original setting)							
Polyhistor [38]	Swin-T, frozen	70.87	59.54	65.47	17.47	71.85	−8.93
VMT-Adapter [62]	Swin-T, frozen	71.60	60.67	64.02	16.41	72.10	−8.32
MTLoRA [1] (r=64)	Swin-T, frozen + LoRA	67.90	59.84	65.40	16.60	71.95	−9.21
(5) Multi-Task PEFT Extended to Fully Tunable ViT-L Backbone							
Polyhistor [38]	ViT-L, fully tunable	77.50	65.80	83.20	14.50	72.00	−3.66
VMT-Adapter [62]	ViT-L, fully tunable	78.60	67.00	83.80	14.30	72.40	−2.50
MTLoRA [1]	ViT-L, fully tunable	78.10	66.40	83.50	14.40	72.20	−3.07
(6) VTP_{timm} (Ours)	ViT-L, fully tunable	**82.88**	**73.85**	**85.52**	**13.56**	**75.12**	**+2.77**

E Visualization of Learned Task Vectors

To gain insights into how task vectors learn and evolve across the model, we analyze the prompt embeddings from two complementary perspectives: (i) activation correlation across tasks and layers, which reveals the pairwise similarity structure of the learned prompts, and (ii) UMAP [42] visualization, which reveals the organization of the prompt embedding space. We conduct this analysis on Pascal-Context with $T = 5$ tasks (semseg, parsing, saliency, surface normals, and boundary detection) and compare TaskPrompter [71] against our VTP.

E.1 Visualization setup.

For the activation analysis, we compute the pairwise cosine similarity between all task prompt vectors at each ViT block and visualize the resulting $T \times T$ correlation matrices as heatmaps, where red indicates high correlation and white indicates low correlation. This is performed across all $L = 24$ ViT blocks to reveal how inter-task vector relationships develop with network depth. For the UMAP analysis, we extract VTP's $T = 5$ compressed prompt vectors (after the width connection in Eq. 6) at four evenly spaced ViT blocks (blocks 6, 12, 18, and 24), randomly sample 500 images from the Pascal-Context test set, and project the resulting embeddings into a 3D space using UMAP with n_neighbors $= 15$, min_dist $= 0.1$, and cosine distance. All the points are color-coded by their task identities and are annotated by layer.

E.2 TaskPrompter: anomalous activation patterns.

As shown in Figure 4 (a), TaskPrompter's task vectors exhibit progressive over-alignment with network depth. At shallow layers, the correlation matrix shows moderate inter-task similarity with discernible pair-wise structure, indicating that the task vectors initially encode differentiated information. However, as depth increases, this structure is rapidly eroded: by layer 18, the mean off-diagonal correlation reaches 0.87 with a min–max range of only 0.04, and by layer 24 it further rises to 0.94 with a comparably narrow range. This pattern reveals that the issue is not merely increasing similarity, but the loss of task-pair specificity—all pairs converge toward the same high-correlation value regardless of their semantic relationship. Such uniform saturation indicates representational

collapse, where the task vectors no longer encode meaningful task-specific distinctions and instead degenerate into near-identical directions in the embedding space.

E.3 VTP: well-structured UMAP clusters.

Figure 4 (b) presents the 3D UMAP projection of VTP's compressed prompt embeddings across four representative layers. It shows clearly separated clusters, with each task forming a distinct, compact group in the embedding space. The five task clusters are well-separated from one another, indicating that VTP's read-write cycle successfully distills task-discriminative information from the richer $T + V$ virtual prompt bank into the T prompts that interact with image tokens. This structure confirms that the read-write routing matrices learn to organize the prompt space in a way that preserves both task specificity and cross-task sharing, which is precisely the balance that effective multi-task learning requires. Furthermore, the layer-wise information encoded in the UMAP reveals that task vectors within each layer form an internally cohesive latent subspace tailored to that layer's specific representational role, rather than collapsing into a single shared representation.

E.4 VTP: progressive activation specialization.

Figure 4 (c) shows VTP's activation correlation matrices across the same 24 ViT blocks, providing a direct counterpart to the TaskPrompter analysis in (a). Holistically, VTP preserves a layer-dependent task-affinity geometry. In early layers, the strongest correlations appear between tasks that share fine-grained geometric and boundary cues, particularly surface normals and edges. As depth increases, semantic pairs such as semantic segmentation and saliency become increasingly aligned, reflecting the transition from local boundary processing to object-level reasoning. Importantly, this evolution does not collapse into uniform high correlation: weakly related task pairs remain clearly separated from strongly related ones throughout all layers. This structured evolution contrasts sharply with TaskPrompter, whose deep-layer task vectors become nearly homogeneous across all task pairs (off-diagonal range of 0.05 at layer 24, compared to 0.29 for VTP).

F Discussion on Flexibility and VFM Compatibility

In the main text, we describe VTP as a flexible, drop-in prompting architecture that preserves full compatibility with off-the-shelf VFMs. Below, we discuss these properties and compare them with existing multi-task scene understanding architectures, and outline the integration procedure of VFMs.

F.1 Drop-in Compatibility

The aforementioned drop-in compatibility indicates that VTP can be attached to existing standard pretrained ViTs without modifying its internal block topology, weight shapes, or forward-pass logic.

Unmodified ViT block internals. Multi-head self-attention, feed-forward networks (FFNs), layer normalization, and residual connections within each standard ViT block remain untouched. Rather than inserting specifically tailored attention heads or altering the residual streams, VTP simply extends the input image sequence by T task tokens along the sequence dimension, which is an operation natively supported by standard Transformer implementations [45, 52].

Exact weight-shape preservation. Since VTP preserves the ViT block's internal structure, all pretrained weight tensors retain their original pretrained shapes and can be loaded without any remapping, pruning, or interpolation. This is in direct contrast to previous multi-task prompting methods like TaskPrompter [71], which modifies the inner self-attention module to incorporate spatial-channel dual attention for enhanced representation capacity. Such modifications inevitably alter the weight shapes of the attention projection matrices, making it difficult to directly load off-the-shelf VFM weights without re-engineering the model architectures. VTP avoids this issue entirely: the pretrained backbone weights are loaded as-is, and only the VTP-specific parameters (prompt embeddings, read-write routing matrices, prediction heads) are initialized from scratch.

Preserved macro architecture. VTP does not alter the number of ViT blocks, the embedding dimension, the number of attention heads, or the feed-forward expansion ratio. Operating entirely as an external bypass that interfaces with the encoder at block boundaries, VTP supports different ViT scales (e.g., ViT-B, ViT-L, ViT-H) without further structural adaptation.

Compatibility across VFM families. We validate this property empirically by deploying VTP on three distinct weight sources: (i) standard ImageNet-supervised pretraining (ViT-L from `timm` [16]), (ii) DINOv3 [52], a self-supervised VFM trained with a distillation-based objective, and (iii) ViT-5 [57], a recent VFM trained with a hybrid supervised–self-supervised pipeline. These three represent qualitatively different pretraining paradigms, yet VTP integrates with each by simply swapping the backbone weights—no code changes, no hyperparameter re-tuning for the VTP itself. The results in Tables 1 and 2 confirm that VTP benefits from each VFM's strengths without compatibility issues.

F.2 Architectural Flexibility

Beyond VFM compatibility, VTP offers flexibility across three practical dimensions:

Flexibility in model choice. As aforementioned, VTP is agnostic to the specific ViT variant and pretrained weights used as the backbone. This allows practitioners to adopt their available VFM for their domain, *e.g.*, a medical imaging VFM for clinical applications, a remote sensing VFM for satellite imagery, without any modifications to the VTP module. The only need is that the backbone follows the standard ViT block structure: multi-head self-attention followed by a feed-forward network, with layer normalization and residual connections.

Task scalability. VTP accommodates varying numbers of tasks without structural changes to the architecture. Adding a new task requires only: (a) adding one new learnable prompt embedding (increasing T by 1), (b) expanding the read-write routing matrices by one row/column, and (c) adding a new lightweight prediction head. The encoder backbone and the VTP module's core logic remain unchanged. This is validated empirically by our experiments spanning 4 tasks (NYUD-v2), 5 tasks (Pascal-Context), and 13 tasks (Taskonomy), all using the same VTP architecture with only the task count adjusted. The virtual task number V can remain fixed across these settings as shown in Table 3.

Flexibility in deployment. Since VTP's external routing operates solely on the $T+V$ task stream, it introduces minimal parameter and FLOP overhead when $T, V \ll N$. As shown in Table 3 (right), VTP_{timm} reduces parameter count (342M vs. 401M) and increases inference throughput (16.3 vs. 14.4 imgs/s) compared to TaskPrompter [71], making it well-suited for resource-constrained deployments.

F.3 Compatibility Comparison with Prior MTL Architectures

Existing multi-task scene understanding architectures face distinct barriers when adapting off-the-shelf VFMs: **(i) TaskPrompter [71]** replaces standard self-attention with spatial-channel dual attention and channel-wise sliding windows, creating weight-shape mismatches that preclude direct checkpoint loading. **(ii) InvPT [69] and InvPT++ [72]** interleave cross-attention modules that tightly couple multi-scale decoder projections to backbone feature distributions, requiring careful per-VFM recalibration. **(iii) TaskExpert [70]** inserts dynamic mixture-of-experts gating between encoder layers, similarly coupling routing calibration to specific backbone hidden states. **(iv) MTMamba [31] and PAME [76]** rely on state-space decoders with specialized scanning patterns tailored to specific feature hierarchies, which complicates backbone substitution.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Sec. 1) clearly state three contributions: (i) VTP as a prompting-based architecture that eliminates inner encoder modifications (validated in Sec. 3.2), (ii) seamless VFM integration advancing state-of-the-art on Pascal-Context and NYUD-v2 (validated in Table 1, Sec. 4.2), and (iii) stress-testing on Taskonomy with 13 tasks (validated in Table 2, Sec. 4.3). The claims about gains stemming from the VTP method more than added capacity are supported by ablations in Table 3 and the mode-collapse analysis in Figure 4. Limitations are acknowledged in Sec. 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5 explicitly discusses limitations: (i) VTP has not been tested with larger multimodal foundation models such as Qwen3-VL due to limited computational resources, and (ii) exploring more expressive prompting mechanisms and scaling to additional modalities are identified as future work. Additionally, Sec. 4.3 acknowledges that not all tasks benefit uniformly (*e.g.*, VTP_{timm} regresses on 3D edges, $\text{VTP}_{\text{vit-5}}$ degrades on normals in Taskonomy), and Sec. 4.2 notes that boundary detection remains below MTMamba.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results such as theorems, lemmas, or proofs. It is primarily an empirical model architecture contribution. The mathematical formulations in Sec. 3 (Eqs. 1–9) define the multi-task model architecture and overall training objectives rather than stating theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides comprehensive implementation details in Sec. 4.1 and Appendix B, including: optimizer (Adam), learning rate (2×10^{-5}), weight decay (1×10^{-6}), batch size (4), training iterations (40,000), polynomial schedule with power 0.9, data augmentation strategies, input resolutions, task-balancing weights, initialization schemes for routing matrices (Φ^l, Ψ^l, Γ^l), prompt initialization ($\mathcal{N}(\mu{=}1, \sigma{=}1)$), multi-scale extraction points ($K{=}4$), virtual task count ($V{=}2$), loss functions per task type, and GPU type (NVIDIA L40-48G). The included datasets are all publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code and trained checkpoints will be released upon acceptance to preserve double-blind anonymity. The training corpus is built from publicly available datasets (Pascal-Context, NYUD-v2, Taskonomy), and the architectural details and hyperparameters are described in sufficient detail (Sec. 4.1 and Appendix B) to enable re-implementation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Sec. 4.1 and Appendix B provide all essential training and evaluation details: data splits, optimizer, learning rate, weight decay, batch size, training iterations, learning rate schedule, input resolution, data augmentation, loss functions, task-balancing weights, initialization protocols for all VTP-specific components, multi-scale feature extraction layers, and evaluation metrics with precise definitions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars, confidence intervals, or results over multiple runs. This is consistent with the prevailing practice in the multi-task dense prediction literature, where the high computational cost of training large ViT-L models over 40,000 iterations makes multi-seed evaluation prohibitively expensive. The paper partially compensates by evaluating across three diverse benchmarks (Pascal-Context, NYUD-v2, Taskonomy) with up to 13 tasks and three different backbone initializations (timm, ViT-5, DINOv3), providing evidence of robustness across settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper specifies the GPU type used for all experiments (NVIDIA L40-48G, stated in Appendix B). Throughput measurements (images/second) on L40 GPUs are reported in Table 3 (right). The number of training iterations, batch sizes, and model parameters/FLOPs are all reported, enabling compute estimation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. It uses publicly available, well-established benchmarks (Pascal-Context, NYUD-v2, Taskonomy) with proper citations to the original dataset creators. The work does not involve human subjects, does not raise privacy concerns, and the submission preserves anonymity. The method is a general-purpose multi-task scene understanding architecture without direct harmful applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The work is foundational research on multi-task model architectures for scene understanding. The method could benefit applications in autonomous driving and embodied AI, and these domains carry inherent societal implications. As with all the deep learning models in these area, this could amplify dual-use risks. But our method is a pure research on multi-task prompting, inheriting any safeguards or limitations of the base model.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pretrained models, new datasets, or any artifacts that pose a high risk for misuse. VTP is a multi-task scene understanding architecture that operates on existing public benchmarks and does not generate content (e.g., images or text) that could be misused.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets are publicly available under their respective open licenses and are properly cited in this paper (Pascal-Context, NYUD-v2, and Taskonomy). All compared methods and pretrained model sources are cited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not introduce new assets. The contribution is a multi-task model architecture described in full detail in the paper, but no downloadable assets are provided with the submission.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments use existing public benchmarks with pre-existing annotations, and evaluation is fully automated using standard metrics.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects. All experiments are conducted on publicly available benchmark datasets using automated evaluation protocols.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology of this paper (Virtual Task Prompting) does not involve LLMs as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.