Discovering Adaptive Task Dependencies for Efficient Multi-Task Representation Compression

Supplementary Material

A. Derivations of the Proxy Loss

A.1 Conditional Cross-Entropy

We provide a compact derivation of the conditional cross-entropy used in Eq. (4) of the main paper. For any true conditional distribution $p(Z_i|Z_j)$ and its parametric approximation $q_\phi(Z_i|Z_j)$, the conditional cross-entropy is defined as

$$\mathcal{H}(p, q_\phi) = -\mathbb{E}_{p(Z_i,Z_j)}[\log q_\phi(Z_i|Z_j)]\,. \tag{10}$$

Adding and subtracting $\log p(Z_i|Z_j)$ inside the expectation gives

$$\mathcal{H}(p, q_\phi) = -\mathbb{E}[\log p(Z_i|Z_j)] - \mathbb{E}\left[\log \frac{q_\phi(Z_i|Z_j)}{p(Z_i|Z_j)}\right]. \tag{11}$$

The first term is the true conditional entropy

$$H_p(Z_i|Z_j) = -\mathbb{E}[\log p(Z_i|Z_j)]\,, \tag{12}$$

and the second term is the KL divergence

$$\mathrm{KL}(p(Z_i|Z_j) \,\|\, q_\phi(Z_i|Z_j)) = \mathbb{E}\left[\log \frac{p(Z_i|Z_j)}{q_\phi(Z_i|Z_j)}\right]. \tag{13}$$

Therefore,

$$\mathcal{H}(p, q_\phi) = H_p(Z_i|Z_j) + \mathrm{KL}(p(Z_i|Z_j) \,\|\, q_\phi(Z_i|Z_j))\,, \tag{14}$$

which immediately yields the upper bound

$$\mathcal{H}(p, q_\phi) \geq H_p(Z_i|Z_j), \tag{15}$$

with equality if and only if $q_\phi(Z_i|Z_j) = p(Z_i|Z_j)$.

Since the true conditional entropy $H_p(Z_i|Z_j)$ is independent of q_ϕ, minimizing $\mathcal{H}(p, q_\phi)$ with respect to ϕ does not minimize $H_p(Z_i|Z_j)$. Instead, it minimizes the KL divergence in Eq. (14), driving $q_\phi(Z_i|Z_j)$ closer to the true conditional distribution. When the approximation is sufficiently accurate, the cross-entropy $\mathcal{H}(p, q_\phi)$ becomes a tight, differentiable upper bound of $H_p(Z_i|Z_j)$, and the per-sample negative log-likelihood $-\log q_\phi(Z_i|Z_j)$ provides a practical proxy for estimating conditional rates and task predictability.

A.2 Proxy Loss

To obtain a tractable approximation of the conditional distribution $p(Z_i|Z_j)$, the proxy head models $q_\phi(Z_i|Z_j)$ as a Gaussian whose parameters are predicted from Z_j. For a normalized feature vector $z'_i \in \mathbb{R}^d$, the negative log-likelihood under this model is

$$-\log q_\phi(z'_i|Z_j) = \frac{1}{2}\sum_{k=1}^{d}\left[\log(2\pi\sigma_{i,k}^2) + \frac{(z'_{i,k} - \mu_{i,k})^2}{\sigma_{i,k}^2}\right]. \tag{16}$$

Since the constant term $\frac{1}{2}\log(2\pi)$ does not affect optimization, we use the simplified expression

$$-\log q_\phi(z'_i|Z_j) \simeq \frac{1}{2}\sum_{k=1}^{d}\left[\log \sigma_{i,k}^2 + \frac{(z'_{i,k} - \mu_{i,k})^2}{\sigma_{i,k}^2}\right]. \tag{17}$$

Taking expectation over the empirical data distribution yields the proxy loss used in the main paper:

$$\mathcal{L}_{\text{proxy}} = \frac{1}{2}\,\mathbb{E}\left[\log \sigma_i^2 + \frac{\|z'_i - \mu_i\|_2^2}{\sigma_i^2}\right], \tag{18}$$

where all operations are applied elementwise over channel dimensions.

The Gaussian form does not assume that the true conditional distribution $p(Z_i|Z_j)$ is Gaussian; instead, it provides a tractable, differentiable parameterization for q_ϕ. Minimizing $-\log q_\phi(z'_i|Z_j)$ reduces the KL divergence term in Eq. (14), making $q_\phi(Z_i|Z_j)$ a closer approximation to the true conditional distribution. Consequently, the per-sample negative log-likelihood serves as a practical proxy for conditional entropy and enables estimating inter-task predictability in a differentiable manner.

B. Acyclicity Enforcement of DAG

To guarantee that the inferred structure remains a directed acyclic graph, we employ a greedy construction procedure that incrementally includes candidate edges without forming cycles.

- Initialize each parent set $S_i = \emptyset$.
- Enumerate all directed edges $(t_p \to t_i)$ and sort them in descending order of gain $g_{p\to i}(x)$.
- Iterate through the sorted list; an edge is added only if
  1. inserting it does not introduce a directed cycle, and
  2. the child node t_i has not exceeded the maximum in-degree $K_{\max}$.
- Cycle detection is performed using a depth-first search (DFS) on the current partial graph; the overall complexity remains $O(N^2)$ for N tasks.

This greedy procedure favors strong conditional dependencies while ensuring that the final graph remains acyclic for every input instance.

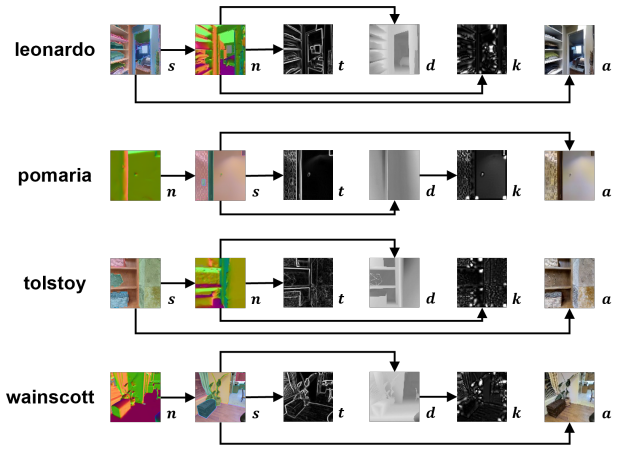

Figure 7. Adaptive DAGs inferred from different scene types. While specific structures vary with content, the high-level task relationships remain stable across images.

C. DAG Visualization and Stability

This section provides additional qualitative and quantitative evidence for the stability and semantic coherence of the adaptive DAGs inferred by our method. We present several scene-specific visualizations, followed by edge-frequency statistics aggregated over the validation set.

C.1 Scene-wise DAG Visualizations

Fig. 7 shows DAGs inferred from several representative indoor scenes in Taskonomy (e.g., leonardo, pomaria, tolstoy, wainscott). Each DAG is the most frequently inferred structure for a specific scene. In structure-dominated scenes (e.g., leonardo, tolstoy), semantic segmentation and surface normals consistently emerge as early nodes in the DAG. In more cluttered scenes, we occasionally see texture or edge maps move upward in the hierarchy, which aligns with their stronger local evidence in these cases. The overall patterns remain consistent across images, confirming the content-adaptive but stable nature of the inferred graphs.

C.2 Edge Frequency Statistics

To quantify the stability of the inferred dependency structures, we compute the occurrence frequency of each directed edge across the validation set. Let $A_{p\to i}$ denote the number of images for which the edge $t_p \to t_i$ appears in the final DAG. The normalized frequency is defined as

$$F_{p\to i} = \frac{A_{p\to i}}{N_{\text{images}}}. \tag{19}$$

Table 3 reports the most frequent edges. Consistent and interpretable patterns emerge: geometry-related tasks (e.g., normals) often serve as strong parents for other geometry or structure-related tasks, while semantic segmentation provides complementary cues for keypoint predictions. These high-frequency edges indicate that the learned dependencies are not random artifacts but reflect stable conditional relationships shared across diverse scenes.

Table 3. Top directed edges ranked by occurrence frequency on the validation set.

Edge	Frequency $F_{p\to i}$
normal → edge_texture	0.58
normal → keypoint2d	0.57
normal → depth_zbuffer	0.55
semseg → keypoint2d	0.52

Interestingly, we observe that all possible directed edges occur at least once across the dataset. The rarest edge is *edge_texture* → *semseg*, which still appears with a non-zero frequency of 0.02, illustrating that the adaptive graphs fully explore the space of potential inter-task relations while maintaining clear dominant patterns.

D. Ablation Studies

This section provides additional ablations that analyze the sensitivity of the adaptive DAG mechanism to key hyperparameters. We begin with the gain threshold τ, which determines whether a candidate edge is included in the graph.

D.1 Effect of the Gain Threshold τ

The threshold τ controls the minimum conditional gain required for adding an edge $t_p \to t_i$ into the DAG. A lower threshold leads to denser graphs with potentially noisy dependencies, whereas a higher threshold yields sparser structures that may discard useful parents.

We evaluate $\tau \in \{0.0, 0.1, 0.2, 0.5\}$. Table 4 summarizes the average number of edges per graph and the corresponding rate–accuracy performance. TNL denotes the total normalized loss aggregated over five analysis tasks. The results show that the adaptive DAG remains stable across a broad range of thresholds, with $\tau = 0.2$ achieving the best trade-off between graph sparsity and predictive performance. A moderate threshold such as $\tau = 0.2$ effectively removes weak or unstable edges without over-pruning, producing compact yet reliable DAGs for downstream compression. Note that a larger number of edges generally indicates higher encoding complexity, so choosing τ also involves balancing predictive performance against computational cost.

D.2 Effect of the Maximum In-Degree $K_{\max}$

The hyperparameter $K_{\max}$ limits how many parent tasks each node is allowed to depend on. Increasing $K_{\max}$ enhances the predictive capacity of the adaptive DAG, since each task can aggregate richer contextual cues from more

Table 4. Ablation on the gain threshold τ. Lower values produce denser DAGs, while higher values yield sparser ones.

τ	Avg. Edges	Rate (bpp)	TNL↑
0.0	8.3	0.0132	4.8405
0.1	6.1	0.0129	4.8358
0.2	5.4	0.0130	**4.8464**
0.5	3.8	0.0148	4.8143

Table 5. Ablation on the maximum in-degree $K_{\max}$. Larger values increase predictive capacity but also raise computational and training complexity.

$K_{\max}$	Rate (bpp)	TNL↑
1	0.0127	4.8213
2	0.0130	4.8464
3	0.0137	**4.8531**
5	0.0145	4.8426

parents. However, a larger in-degree also raises both inference and training complexity. During inference, more parent features must be decoded before each child task, resulting in deeper dependency chains and additional predictive steps.

More importantly, increasing $K_{\max}$ substantially enlarges the training burden. For a task with N potential parents, constructing training samples for every possible parent subset requires $\binom{N}{K_{\max}}$ combinations. Thus, moving from $K_{\max} = 2$ to $K_{\max} = 3$ already requires covering a significantly larger set of parent configurations, which increases the number of training pairs and the overall optimization cost. This combinatorial explosion makes very large $K_{\max}$ values impractical despite their higher expressive capacity.

We evaluate $K_{\max} \in \{1, 2, 3, 5\}$. Table 5 reports the rate (bpp), and total normalized loss (TNL) across five analysis tasks. As expected, predictive performance improves when increasing $K_{\max}$, peaking at $K_{\max} = 3$, which provides the richest dependency structures. However, this comes at the cost of noticeably denser graphs, higher bitrate, and significantly increased training cost due to the $\binom{N}{K_{\max}}$ growth in parent-set combinations.

Interestingly, $K_{\max} = 2$ achieves nearly the same predictive performance as $K_{\max} = 3$ but with substantially fewer edges, better rate efficiency, and a much smaller combination space for training. In contrast, restricting to a single parent ($K_{\max} = 1$) imposes excessive sparsity and degrades accuracy.

Overall, while $K_{\max} = 3$ offers the strongest predictive accuracy, the additional computational cost, higher bitrate, and the combinatorial increase in training data requirements make it considerably less practical. $K_{\max} = 2$ therefore provides the best balance between accuracy, efficiency, and training feasibility.

E. Additional Implementation Details

This section provides implementation details omitted from the main paper due to space constraints. We describe the task-specific losses used to train the multi-task backbone and present the full computation of the Total Normalized Loss (TNL) reported in Sec. 4.

E.1 Tasks and Training Losses

We evaluate six representative analysis tasks from Taskonomy [57], each equipped with a task-specific decoder and loss function. Below we provide the full definitions and evaluation criteria used in our experiments.

Semantic Segmentation. The dataset contains 18 semantic categories, including 16 object classes, one background class, and one uncertain label. We supervise the segmentation decoder using the standard pixel-wise cross-entropy loss:

$$\mathcal{L}_s = -\sum_{c=1}^{18} y_c \log p_c.$$

Compression performance is evaluated at multiple bitrates using the same loss.

Keypoint2D. This task predicts 2D keypoint heatmaps. Following Taskonomy, we train the decoder with an L_1 heatmap regression loss:

$$\mathcal{L}_k = \|H - \hat{H}\|_1.$$

Performance is reported across bitrates according to this loss.

Edge Texture. Edge texture maps capture high-frequency structural boundaries. We adopt a per-pixel L_1 loss:

$$\mathcal{L}_t = \|t - \hat{t}\|_1.$$

Evaluation follows the same metric across different compression rates.

Surface Normal. Surface normals are encoded as 3-channel vectors centered at 127. To ensure robustness against viewpoint and orientation variations, we adopt the *rotate loss*, commonly used in image processing and neural rendering:

$$\mathcal{L}_n = \min_{\theta \in \Theta} \|R_\theta(n) - R_\theta(\hat{n})\|_1,$$

where R_θ rotates the prediction and target using one of $|\Theta| = 9$ predefined orientations. This encourages consistent normal predictions under small spatial transformations.

Depth (Z-buffer). Depth values are encoded in units of $1/512$ meters, with a maximum range of 128 meters. Similar to surface normals, we use rotate loss:

$$\mathcal{L}_d = \min_{\theta \in \Theta} \left\| R_\theta(d) - R_\theta(\hat{d}) \right\|_1,$$

which stabilizes depth estimation under viewpoint perturbations, reducing sensitivity to noise and local misalignment.

Autoencoder/Image Compression. The autoencoder reconstructs 512×512 RGB images. For fidelity and perceptual assessment, we additionally compute PSNR, SSIM, LPIPS, and FID across bitrates, though these metrics are not used for training.

E.2 Total Normalized Loss

To report a unified metric across heterogeneous analysis tasks, we compute the Total Normalized Loss (TNL). For each task m, we compute a normalized score relative to a task-independent baseline under the same bitrate:

$$\mathrm{NormLoss}_m = 1 - \frac{\mathcal{L}_m - \mathcal{L}_m^{\mathrm{ref}}}{\mathcal{L}_m^{\mathrm{ref}}}.$$

The TNL score is defined as the sum over the five analysis tasks (excluding autoencoding):

$$\mathrm{TNL} = \sum_{m \in \{n,d,s,k,t\}} \mathrm{NormLoss}_m.$$

Higher values indicate better multi-task prediction accuracy. This normalization eliminates scale differences among tasks and enables aggregation into a single performance metric.

E.3 Additional Task-Level Results

Sec. 3.4 of the main paper shows representative curves for two tasks. For completeness, Fig. 8 reports the remaining three task-level results. The adaptive DAG achieves consistent improvements over both *random* and *static* baselines across all bitrates, verifying that the benefits of content-adaptive task ordering generalize across all analysis tasks.

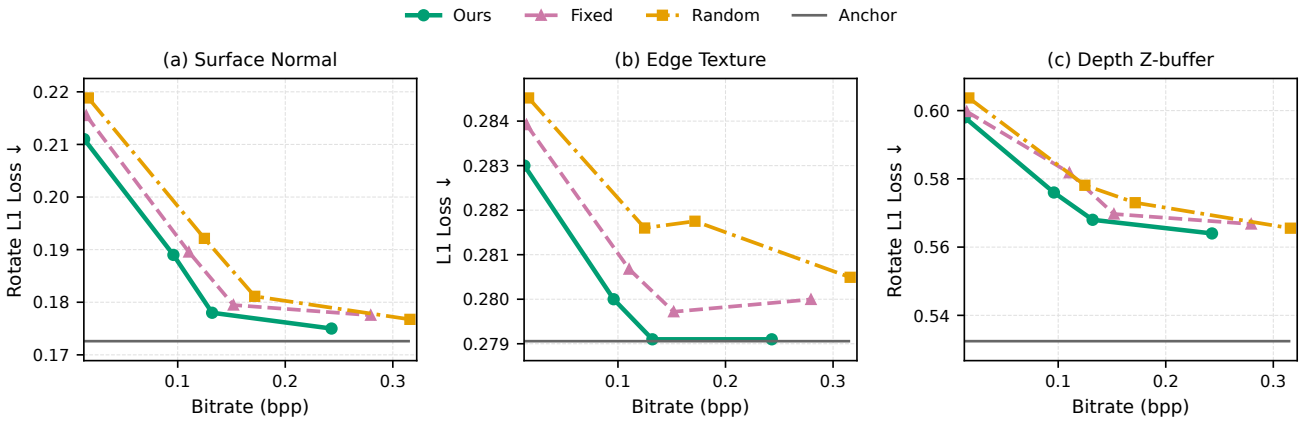

Figure 8. Supplementary task-level performance curves for the remaining three analysis tasks. Across all bitrates, the proposed Adaptive DAG consistently improves over both *random* and *static* compression orders.