Semantic Routing: Exploring Multi-Layer LLM Feature Weighting for Diffusion Transformers

Bozhou Li[1,2,*] **Yushuo Guan**[2,†] **Haolin Li**[3] **Bohan Zeng**[1] **Yiyan Ji**[4]
Yue Ding[5] **Pengfei Wan**[2] **Kun Gai**[2] **Yuanxing Zhang**[2,‡] **Wentao Zhang**[1]
[1]Peking University [2]KlingAI [3]Fudan University
[4]Nanjing University [5]School of Artificial Intelligence, University of Chinese Academy of Sciences
Code: `https://github.com/zooblastlbz/SemanticRouting`

Abstract

Recent DiT-based text-to-image models increasingly adopt LLMs as text encoders, yet text conditioning often remains static, relying on a fixed representation from a single LLM layer across all denoising timesteps and DiT blocks. This motivates a central question for DiT text conditioning: when multi-layer LLM features are available, how should hierarchical semantics be injected into the dynamic generation process? To investigate this question, we introduce Semantic Routing, a lightweight multi-layer conditioning interface that forms text features through normalized convex fusion of LLM hidden states and compares static, time-wise, depth-wise, and joint routing strategies under matched training and evaluation settings. Across text–image alignment and compositional benchmarks, depth-wise routing emerges as the most reliable strategy among the studied variants, with particularly strong gains on advanced prompt-following tasks (e.g., +9.97 on GenAI-Bench Counting over the penultimate-layer baseline). In contrast, purely time-wise routing can degrade visual fidelity; our analysis suggests a mismatch between nominal timesteps and the effective inference trajectory, making time-conditioned semantic injection poorly timed. These results indicate that DiT depth is an effective axis for improving text–image alignment through dynamic LLM feature injection, while robust time-dependent conditioning remains an open challenge.

1 Introduction

Recent text-to-image diffusion models increasingly adopt Diffusion Transformers [1] and stronger text encoders [2–8]. In particular, several systems have moved from encoder-only text encoders such as T5 [9] and CLIP [10] toward decoder-only LLMs, aiming to use their richer linguistic representations for visual generation [11–15].

The conditioning interface in predominant DiT systems follows a static design: a fixed text representation, commonly taken from a selected encoder layer or a fixed aggregation of layers, is reused across all denoising timesteps and all DiT blocks. This static interface contrasts with the hierarchical structure of LLM representations and the evolving nature of DiT denoising.

This contrast arises from two sources of variation. From the LLM side, hidden states across layers can encode different linguistic and semantic information [16–20]. From the DiT side, the text condition is consumed by a denoising process whose context changes across both denoising time and network depth: the denoising state evolves over timesteps, and different DiT blocks can contribute

*Work done during an internship at KlingAI (`libozhou@pku.edu.cn`).
†Project leader.
‡Corresponding author.

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

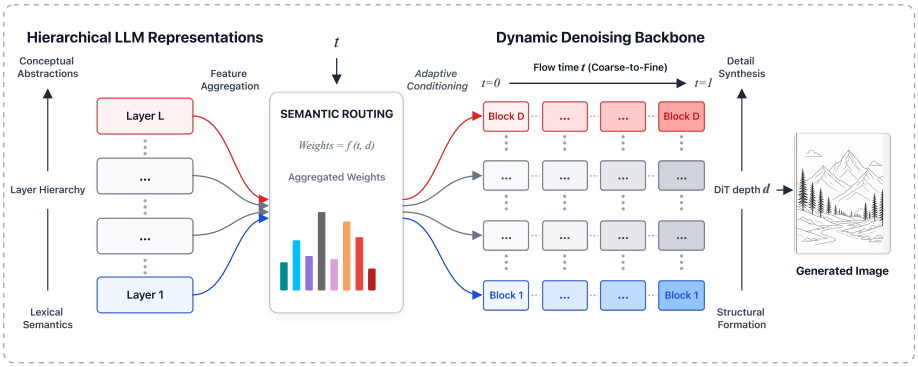


Figure 1: **Overview of Semantic Routing.** Multi-layer LLM hidden states provide a hierarchy of text features. Semantic Routing forms the conditioning signal by mixing these features with weights conditioned on flow time t and DiT block depth d, then injects the routed representation into the denoising backbone. This schematic summarizes the design space studied in this paper: static, time-wise, depth-wise, and joint multi-layer conditioning.

unevenly to structure formation and detail refinement [21]. Together, these two forms of variation motivate a routing view of DiT text conditioning: multi-layer LLM features provide a pool of semantic signals, while denoising time and DiT depth specify where those signals are consumed by the generation process. Figure 1 visualizes this view by routing multi-layer LLM features through weights conditioned over flow time t and DiT depth d, producing conditioning signals for different stages and blocks of the denoising backbone.

Guided by this routing view, we study **Semantic Routing**, a lightweight method for dynamic multi-layer LLM conditioning. The method represents the text condition as a normalized convex fusion of LLM hidden states, and compares fusion weights that vary along two orthogonal axes: (i) time-wise routing, where weights vary with the diffusion timestep, and (ii) depth-wise routing, where weights vary with the DiT block index. We further study their combination to characterize the interaction between temporal and depth adaptivity. By keeping the DiT backbone, text encoder, training data, and optimization protocol fixed, this interface lets us isolate the effect of the routing axis itself. Our contributions are as follows:

- **Semantic Routing for Multi-layer LLM Conditioning.** We formulate text conditioning as routing over LLM hidden-state layers, where the routing weights can depend on diffusion time, DiT depth, or both. This provides a lightweight and controlled interface for studying dynamic semantic injection in DiT generation.
- **Depth-wise Routing as an Effective Axis for Text–Image Alignment.** Under matched training and evaluation settings, depth-wise routing is the most reliable strategy among the studied variants for improving text–image alignment. It yields a **+9.97** improvement on GenAI-Bench Counting over the penultimate-layer baseline and **+5.45** over uniform averaging, with particularly pronounced gains on compositional reasoning tasks.
- **Trajectory Mismatch in Time-dependent Routing.** We analyze time-wise fusion through the lens of **train–inference trajectory mismatch**. Our analysis suggests that nominal timesteps can become misaligned with the effective inference trajectory, which is consistent with semantically mistimed feature injection and motivates trajectory-aware conditioning signals.

2 Related Work

2.1 Text Conditioning and Text Encoders

Text conditioning has evolved from U-Net cross-attention [22] to DiT's adaLN-Zero [1]. To enable fine-grained control beyond global adaLN, PixArt-α reintroduced cross-attention [23], while MMDiT further unified modalities via joint self-attention [2].

Regarding text encoders, early diffusion models predominantly employed encoder-only architectures such as CLIP [10] or T5 [9]. With the rapid evolution of LLMs, recent research has pivoted toward leveraging decoder-only LLMs to obtain stronger semantics for diffusion conditioning [11–15]. Due to the suboptimal performance of early iterations like LLaMA-2 [24] in this context, LiDiT introduced a refiner module to enhance extracted text features [12]. However, with the advent of more capable foundation models, the reliance on such auxiliary components has diminished [25, 4, 7, 3, 5, 6]. Consequently, the field has gravitated toward direct utilization of LLM representations, where standard practice typically adopts single-layer features, predominantly from the penultimate layer.

2.2 Semantic Heterogeneity and Multi-Layer Fusion

LLMs exhibit distinct representational characteristics across their hierarchy. Probing techniques [16] have elucidated that shallow layers primarily capture lexical semantics, whereas deeper layers are increasingly shaped by next-token prediction objectives. Complementarily, Jin et al. [17] observed that complex conceptual abstractions are typically acquired in deeper layers. Regarding task specificity, the utility of different layers varies across downstream applications [18, 26–28], while Gurnee and Tegmark [29] revealed that spatial and temporal information can be encoded in distinct layers. Notably, intermediate layers offer unique advantages, such as attenuated attention sinks [19] and high semantic compression [20], highlighting the potential of leveraging multi-layer semantics to enhance text conditioning.

Such heterogeneity highlights the potential of multi-layer conditioning, while recent evidence confirms its superiority over single-layer baselines [13]. For instance, Playground v3 [30] and Tang et al. [31] adopt a deep fusion strategy where internal attention K/V states from the LLM are directly reused in DiT's cross-attention, in pursuit of a deeper fusion between the text encoder and the DiT backbone. However, the former lacks rigorous controlled experiments to isolate the source of its improvements, while the latter primarily evaluates its approach against the final LLM layer rather than the typically used penultimate layer. While normalizing multi-layer features [13] or introducing learnable weights for adaptive fusion [14, 15] has shown promise, these conditioning mechanisms typically remain static, applying a uniform fusion strategy regardless of the diffusion timestep or the DiT block index.

2.3 Temporal and Depth-wise Dynamics in DiT

The diffusion generative process is characterized by distinct, non-uniform dynamics along both temporal and structural dimensions. From a temporal perspective, the denoising trajectory follows a coarse-to-fine paradigm: earlier timesteps prioritize the recovery of low-frequency global structures, whereas later stages transition toward the refinement of high-frequency textures and details [32–34]. This temporal non-stationarity suggests that the demand for textual guidance may evolve across different denoising stages. From a depth-wise perspective, DiTs exhibit functional stratification along the network hierarchy: shallower blocks are primarily responsible for structural formation, while deeper blocks contribute more to detail synthesis [21]. These variations across time and depth prompt us to consider whether multi-layer LLM semantics can be effectively fused via time-wise adaptivity, depth-wise adaptivity, or their combination, aimed at enhancing generative models from the perspective of text conditioning.

3 Method

3.1 Problem Setup

We study how to combine hidden states from multiple layers of a pretrained LLM for text conditioning in a DiT under the flow matching formulation [35]. Each point where text conditioning is applied is indexed by a flow time $t \in [0, 1]$ and a DiT block depth $d \in \{0, \ldots, D-1\}$. Our goal is not to prescribe a fixed semantic mapping between LLM layers and these coordinates, but to compare several constrained fusion-weight schemes—differing only in whether the weights are shared globally or adapted along t, d, or both—under an otherwise identical backbone, text encoder, data, and training protocol.

3.2 Preliminaries and Notation

Flow matching and timestep. Let $x(t)$ denote the sample state (or latent) at continuous time $t \in [0, 1]$ along the flow, with marginal distribution $x(t) \sim p_t$, where p_0 is a simple base distribution (e.g., Gaussian noise) and p_1 is the target data distribution. A DiT-based backbone parameterizes a text-conditioned vector field $v_\theta\big(x(t), t, c\big)$, where c denotes the text condition. Given c and an initial condition $x(0) \sim p_0$, sampling is performed by integrating the ODE

$$\frac{dx(t)}{dt} = v_\theta\big(x(t), t, c\big), \tag{1}$$

which transports the sample from p_0 toward p_1. For consistency, we refer to t as the timestep throughout this paper.

DiT depth and conditioning site. The DiT backbone consists of D Transformer blocks. We use $d \in \{0, \ldots, D-1\}$ to index the specific block where the conditioning signal is applied. In our experimental setup, the fused text representation $H_{\text{cond}}(t, d)$ provides the conditioning sequence of text hidden states that is fed to the cross-attention module in block d.

Multi-layer LLM features. Let the pretrained LLM provide hidden-state sequences from its entire hierarchy. Let $\mathcal{L}$ denote this set of layers, with $|\mathcal{L}| = L$. We represent the sequence output from layer $l \in \mathcal{L}$ as $H^{(l)} \in \mathbb{R}^{N \times C}$, where N is the text sequence length and C is the LLM hidden dimension.

3.3 A Constrained Interface for Multi-layer Conditioning

The central design question is whether the fusion weights for combining L LLM layers should adapt across D DiT blocks, timesteps, or both. We define the fused text representation as a convex combination of LayerNorm-normalized layer-wise features:

$$H_{\text{cond}}(t, d) = \sum_{l \in \mathcal{L}} \alpha_{t,d}^{(l)} \cdot \mathrm{LN}\Big(H^{(l)}\Big), \tag{2}$$

where the weight vector $\alpha_{t,d} \in \mathbb{R}^L$ satisfies $\sum_l \alpha_{t,d}^{(l)} = 1$ and $\alpha_{t,d}^{(l)} \geq 0$. LayerNorm [36] reduces scale discrepancies across layers [37, 38], and the simplex constraint keeps the fused output within the convex span of the normalized representations. We obtain $\alpha_{t,d}$ via a softmax over logits $z_{t,d} \in \mathbb{R}^L$:

$$\alpha_{t,d} = \mathrm{Softmax}(z_{t,d})\,. \tag{3}$$

The design space therefore reduces to how $z_{t,d}$ is conditioned on (t, d). Different choices impose different invariance assumptions—sharing $z_{t,d}$ across all (t, d), or adapting it along t, d, or both—and Section 3.4 instantiates each scheme. Appendix A.1 provides a conceptual interpretation of these constraints through the lens of semantic supply and demand, while Appendix A.7 analyzes how the softmax parameterization induces inter-layer suppression as an information-maximizing selection strategy.

3.4 Fusion Weight Routing Schemes as Inductive Biases

We instantiate Eq. (2) with a set of routing schemes that differ only in how the fusion weights are shared across flow time t and DiT depth d. The baselines keep the weights fixed globally or use a non-learned average, while the proposed variants allow the weights to vary along time, depth, or both axes. This design lets us compare which axis of adaptivity is useful under otherwise matched settings.

(B1) Penultimate-layer baseline. This baseline is fixed to a single LLM layer and does not fuse across layers:

$$H_{\text{cond}}(t, d) = \mathrm{LN}\Big(H^{(l^\star)}\Big), \quad l^\star = \text{penultimate}. \tag{4}$$

(B2) Uniform normalized averaging. This baseline uses all LLM layers but imposes a fixed, non-learned weighting after normalization:

$$H_{\text{cond}}(t,d) = \frac{1}{L}\sum_{l\in\mathcal{L}} \mathrm{LN}\Big(H^{(l)}\Big). \tag{5}$$

(B3) Static learnable fusion. This variant learns a global weight vector, while keeping it invariant to both flow time and DiT depth:

$$z_{t,d} = \beta, \quad \beta \in \mathbb{R}^L \text{ (learnable)}. \tag{6}$$

Time-conditioned fusion gate (TCFG). For variants that condition on flow time, we use a lightweight gating module that maps t to fusion logits. We first embed the continuous time using a sinusoidal encoding $\phi(t)$ and then compute the logits via a small MLP:

$$z_t = g_\psi\big(\phi(t)\big), \quad g_\psi = \mathrm{MLP}(\cdot), \tag{7}$$

where $z_t \in \mathbb{R}^L$ yields fusion weights via Eq. (3). We use this module in the time-wise and joint variants; implementation details are provided in Appendix B.

(S1) Time-wise fusion. This variant allows the weights to vary over flow time while remaining shared across DiT blocks:

$$z_{t,d} = z_t = g_\psi\big(\phi(t)\big). \tag{8}$$

(S2) Depth-wise fusion. This variant allows the weights to vary over DiT depth while remaining fixed over flow time:

$$z_{t,d} = z_d = \beta_d, \quad \beta_d \in \mathbb{R}^L \text{ (learnable for each } d). \tag{9}$$

(S3) Joint time-and-depth fusion. This variant allows the weights to vary over both time and depth by assigning each DiT block its own time-conditioned gate:

$$z_{t,d} = g_{\psi_d}\big(\phi(t)\big). \tag{10}$$

The weights $\alpha_{t,d}$ are then obtained via Eq. (3) and used in Eq. (2). This is the most flexible variant in our comparison, and its behavior is evaluated empirically under the matched setting described in Section 4.1.

4 Experiments

4.1 Experimental Setup

Models. We employ Qwen3-VL-4B-Instruct [39] as the text encoder and the pretrained VAE from Stable Diffusion 3 (SD3) [2]. The diffusion backbone is a cross-attention-based DiT comprising $D = 24$ Transformer blocks and approximately 2.24B parameters. The architectural design of the backbone follows the implementation of FuseDiT [31]: we apply 1D RoPE [40] to text prompts, 2D RoPE [41] to image latents, and use QK-Norm [42] in attention. The only architectural difference from FuseDiT is that we do not use Sandwich Norm [43]. Unless otherwise specified, all conditioning variants in Section 3 share this identical backbone and differ only in how multi-layer text features are fused. For the TCFG module, we utilize a 128-dimensional sinusoidal encoding for the timestep input.

Dataset. We train all models on a high-quality subset of LAION-400M [44], comprising approximately 30 million image-text pairs. We replace the original texts with dense synthetic captions [45, 15] generated by Qwen3-VL-32B-Instruct [39]. Images are resized to 256×256, and text prompts are tokenized with a maximum length of 512 tokens.

Training. We train all models using AdamW [46] with $(\beta_1, \beta_2) = (0.9, 0.999)$, a learning rate of 1×10^{-4}, weight decay of 1×10^{-4}, and a constant learning rate scheduler. The batch size is **512**, and all models are trained for **500k** steps. All training is conducted on 64 H200 GPUs. The prompt drop ratio is set to 0.1 to enable unconditional generation. For timestep sampling, we follow the logit-normal distribution used in SD3.

Baselines. We evaluate our proposed strategies against two categories of baselines: (i) the **Standard Baselines** (B1–B3) introduced in Section 3.4; and (ii) **FuseDiT** [31], a representative deep-fusion approach that reuses internal LLM attention K/V states directly within the DiT attention layers to facilitate a more intrinsic integration between the text encoder and the visual backbone.

Evaluation. Unless otherwise specified, images are generated using 50 sampling steps with a CFG [47] scale of 6.0 and the FlowMatch Euler scheduler from diffusers [48]. For each prompt, we generate 4 images with a fixed random seed of 42. We use GenAI-Bench [49] and GenEval [50] to assess text–image alignment in generated samples. For GenAI-Bench, evaluation is conducted using Qwen3-VL-235B-A22B-Instruct as the judge model. To quantify aesthetic appeal, we report the style dimension scores from UnifiedReward-2.0 [51] on samples generated using the DrawBench prompt set [52]. We additionally report FID [53] on MJHQ-30K [54] using 30,000 generated images as a reference perceptual metric.

4.2 Main Results

Table 1: Performance of different fusion strategies. Best in each column is in **bold**, and second best is underlined. FID is evaluated on MJHQ-30K with 30,000 generated images.

		Alignment		Quality	
Category	Method	GenEval ↑	GenAI ↑	Unif. Reward ↑	FID ↓
Baseline	B1: Penult.	64.54	74.96	3.02	25.52
	B2: Uniform	<u>66.51</u>	76.82	**3.06**	**24.77**
	B3: Static	64.77	76.31	<u>3.05</u>	25.25
Deep-fusion	FuseDiT	60.95	75.02	<u>3.05</u>	<u>24.86</u>
Semantic Routing	S1: Time	63.41	76.20	2.97	24.95
	S2: Depth	**67.07**	**79.07**	**3.06**	24.88
	S3: Joint	66.05	<u>77.44</u>	**3.06**	25.24

We report the overall performance in Table 1 and the granular capability breakdown on GenAI-Bench in Table 2. The performance differences across methods are concentrated in the alignment metrics (GenEval, GenAI-Bench), while quality-oriented metrics (UnifiedReward, FID) remain largely comparable. The empirical results further reveal several key insights:

Limits of Static Aggregation. First, aggregating multi-layer features (B2–S3) consistently outperforms the penultimate-layer baseline (B1). This trend suggests that LLM hierarchies contain complementary semantic signals that are largely underutilized by conventional single-layer conditioning. Furthermore, the learnable static fusion (B3) fails to surpass uniform normalized averaging (B2), indicating that without explicit adaptivity, a fixed set of learned weights is not robust enough to outperform a strong uniform prior. This is consistent with the supply–demand view in Appendix A.2: a static allocation must serve all timesteps and DiT blocks, and can therefore become a global compromise over heterogeneous conditioning demands.

Interplay in Deep Fusion Architectures. Regarding the deep-fusion baseline, FuseDiT, Table 1 reveals that its architecture struggles to effectively extract essential textual information for high-quality synthesis. We hypothesize that this performance gap stems from an inherent architectural constraint: by directly reusing internal LLM key/value states within cross-attention, FuseDiT imposes a restrictive coupling that deprives the DiT backbone of the flexibility to dynamically re-contextualize text features. These observations offer a useful perspective for future unified architectures, highlighting the importance of carefully considering the interplay between internal state-sharing mechanisms and the task-specific feature extraction capabilities required for high-fidelity generative modeling.

Superiority of Depth-wise Semantic Routing. Among our proposed strategies, **depth-wise fusion (S2)** delivers the most robust and significant overall gains. Notably, introducing learnable weights in S2 yields clear improvements over B2 (which can be viewed as a fixed-weight depth-wise scheme). This contrast with the static setting (where learnable B3 fails to surpass B2) high-

lights a critical divergence: a purely global allocation is ineffective; rather, the value of learnability is effectively unlocked only when aligned with the depth-wise structural hierarchy. In the formal view of Appendix A.3, S2 relaxes the global sharing constraint by allowing block-specific semantic allocations.

This advantage implies that hierarchical LLM semantics are indispensable for navigating intricate prompts. As shown in Table 2, performance gains are disproportionately pronounced within "Advanced" categories. Taking the "Counting" task as a representative case, S2 achieves a substantial improvement of **+9.97** over B1 and **+5.45** over B2. This disparity underscores a pivotal insight: naive aggregation of multiple layers is insufficient to unlock the full potential of text conditioning. Rather, the synergy created by allowing functional blocks at different DiT depths to selectively route and aggregate task-relevant LLM-layer semantics is the key to mastering compositional reasoning and fine-grained constraint following.

Instability of Time-Awareness and Joint Mitigation. In contrast, purely time-wise fusion (S1) does not provide consistent benefits and often leads to degraded generation quality, manifesting as noticeable blurriness and loss of fine details (see Appendix D). We provide a detailed mechanistic diagnosis of this phenomenon in Section 5.2, attributing it to optimization conflicts arising from fundamental train–inference inconsistencies along the temporal axis. Joint fusion (S3) remains competitive but is slightly less effective than S2. Notably, S3 avoids the blurriness characteristic of S1 by incorporating depth-specificity, a phenomenon further analyzed in Section 5.2.

Table 2: Fine-grained GenAI-Bench performance. **Basic skills** include Attribute, Scene, Spatial relations, Action relations, and Part relations. **Advanced skills** include Counting, Comparison, Differentiation, Negation, and Universal. Numbers in parentheses indicate the change relative to B1 (green/red). **Bold** and underlined denote best and second-best results, respectively.

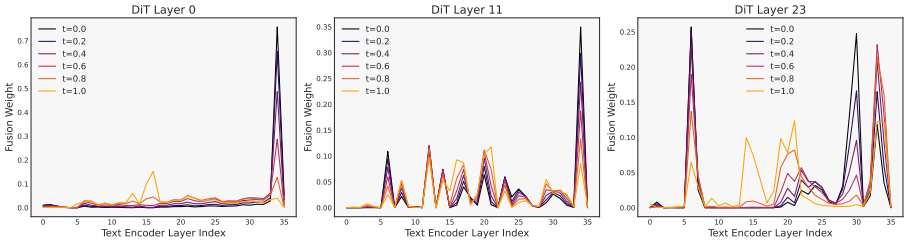
	Method						
Metric	B1: Penultimate	B2: Uniform	B3: Static	FuseDiT	S1: Time-wise	S2: Depth-wise	S3: Joint
Summary							
Avg	74.96 +0.00	76.82 +1.86	76.31 +1.35	75.02 +0.06	76.20 +1.24	**79.07 +4.11**	77.44 +2.48
Basic	80.05 +0.00	81.28 +1.23	80.36 +0.31	77.65 -2.40	79.69 -0.36	82.68 +2.63	**82.92 +2.87**
Adv.	70.55 +0.00	72.95 +2.40	72.88 +2.33	72.65 +2.10	73.16 +2.61	**76.03 +5.48**	72.71 +2.16
Basic skills							
Attribute	79.04 +0.00	81.10 +2.06	79.41 +0.37	77.45 -1.59	79.17 +0.13	81.67 +2.63	**82.16 +3.12**
Scene	85.38 +0.00	86.07 +0.69	84.93 -0.45	83.49 -1.89	83.50 -1.88	**88.33 +2.95**	87.90 +2.52
Spatial rel.	81.27 +0.00	81.98 +0.71	80.59 -0.68	78.69 -2.58	81.03 -0.24	83.08 +1.81	**85.13 +3.86**
Action rel.	83.28 +0.00	85.01 +1.73	84.13 +0.85	81.39 -1.89	83.18 -0.10	86.10 +2.82	**87.08 +3.80**
Part rel.	73.03 +0.00	73.13 +0.10	75.78 +2.75	70.52 -2.51	72.38 -0.65	**77.89 +4.86**	74.16 +1.13
Advanced skills							
Counting	64.60 +0.00	69.12 +4.52	68.32 +3.72	67.23 +2.63	66.37 +1.77	**74.57 +9.97**	67.55 +2.95
Comparison	66.70 +0.00	68.19 +1.49	70.52 +3.82	66.15 -0.55	71.74 +5.04	**72.29 +5.59**	69.44 +2.74
Differentiation	65.39 +0.00	70.49 +5.10	70.57 +5.18	68.48 +3.09	71.49 +6.10	**74.31 +8.92**	70.79 +5.40
Negation	71.21 +0.00	71.51 +0.30	74.73 +3.52	74.16 +2.95	**75.79 +4.58**	74.05 +2.84	72.54 +1.33
Universal	72.76 +0.00	71.10 -1.66	68.21 -4.55	73.31 +0.55	70.98 -1.78	**76.09 +3.33**	72.20 -0.56

5 Analysis

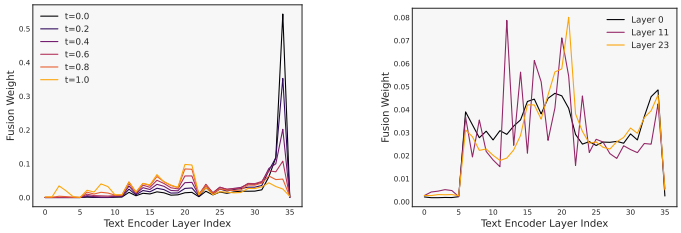
We analyze the proposed fusion strategies from two aspects: (i) the distribution patterns of fusion weights across flow time t and DiT depth d, validating the interpretability of the learned semantic routing; and (ii) a mechanistic diagnosis of the degradation observed in purely time-wise fusion, examining how the mismatch between training-time temporal representations and the inference denoising trajectory leads to semantic misalignment.

5.1 Weight Dynamics over Time and Depth

To validate that the learned fusion weights capture meaningful semantic preferences rather than arbitrary noise, we analyze their evolution across timestep t and DiT depth d in Figure 2. More details are provided in Appendix E.2.

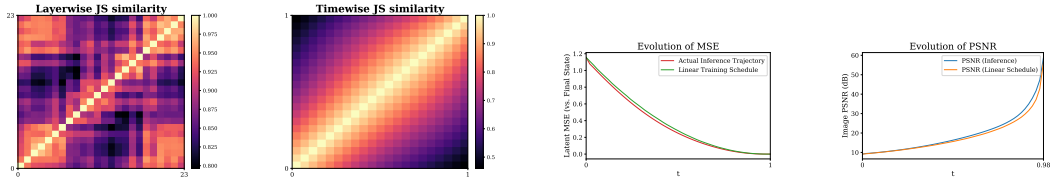


(a) Joint Fusion Weights



(b) Time-wise Fusion Weights (c) Depth-wise Fusion Weights (d) Static Fusion Weights

Figure 2: Weight distributions under different fusion-weight routing schemes. The x-axis denotes the text encoder layer index l, and the y-axis denotes the normalized fusion weight $\alpha_{t,d}$. For time-wise fusion, we sample $t \in [0, 1]$ with a step size of 0.2. For joint fusion and depth-wise fusion, we report representative DiT block indices $d \in \{0, 11, 23\}$. Additional visualizations are provided in Appendix E.1.



(a) Depth JS similarity. (b) Time JS similarity. (c) Latent MSE. (d) PSNR.

Figure 3: Grouped analysis visualizations. Panels (a,b) show local smoothness of learned fusion weights via pairwise JS similarity across DiT blocks and diffusion timesteps. Panels (c,d) show GenAI-Bench trajectory analysis via latent MSE and PSNR over nominal timesteps.

Text Encoder Layer Specificity. According to Figure 2, the learned weights exhibit clear selectivity: the initial and final text encoder layers consistently receive negligible attention, confirming that effective semantics reside within the model's internal depth. Notably, the penultimate text encoder layer dominates primarily during early timesteps but fades as generation progresses, as shown in Figure 2 (a-b). This suggests it serves as a high-level semantic anchor for initial structural layout, while lacking the fine-grained features required for later-stage texture refinement.

Neighbor Inhibition and Information Selection. For the intermediate layers of text encoder, as visualized in Figure 2, we observe an interesting pattern of weight fluctuation across the layer index. This is characterized by local peaks where specific layers receive high weights while their immediate neighbors are noticeably suppressed. We attribute this to an implicit redundancy reduction strategy learned by the model. Since LLM hidden states typically possess high inter-layer similarity due to residual connections, the gating mechanism tends to select the most representative layer within a local neighborhood to avoid redundant information infusion. This selective inhibition is particularly pronounced in the Joint strategy (Figure 2a), which exhibits much sharper peaks and higher local contrast compared to the smoother distributions observed in the decoupled Time-wise and Depth-wise settings. Appendix A.7 analyzes this pattern as an information-maximizing allocation under the softmax sum-to-one constraint: because adjacent layers carry highly similar representations, concentrating weight on one representative per neighborhood maximizes semantic diversity, while

per-block routing (S2, S3) preserves global layer coverage by letting different blocks select different representatives.

Spatiotemporal Dynamism. As shown in Figure 2b, the learned weights shift significantly across timesteps, reflecting the evolving semantic demands of the denoising process. Notably, a comparison between the depth-only setting (Figure 2c) and the joint setting (Figure 2a) reveals that S3 exhibits a stronger depth-dependent reallocation. This suggests that coupling time and depth enables a more nuanced orchestration of feature selection beyond what is possible with independent dimensions.

Emergent Local Smoothness. Similarity visualizations (Figure 3(a,b)) confirm that these variations are highly structured. We observe clear smoothness across neighboring timesteps and adjacent DiT blocks. In the depth-only setting, where no explicit cross-block constraints are enforced, this emergent locality provides robust evidence that the learned routing is driven by coherent semantic signals rather than stochastic optimization noise.

5.2 Trajectory Misalignment in Time-wise Fusion

Our experiments reveal that time-wise fusion leads to degraded generation quality, manifesting as blurriness and detail loss. We attribute this failure to a fundamental train–inference instability regarding the evolution of the SNR.

Empirical Evidence on GenAI-Bench. To quantify diffusion dynamics, we analyze the divergence between the actual inference trajectory and the theoretical forward process on GenAI-Bench. We generate images using a CFG scale of 6.0 and 50 steps. By constructing a reference trajectory from the final generated latent $\hat{x}_1$ via the training forward process, we evaluate the deviation of the intermediate latent x_t. As shown in Figure 3(c,d), the actual inference trajectory consistently outpaces the training schedule, exhibiting lower MSE and higher PSNR at identical nominal timesteps. This effectively decouples the real SNR from the nominal timestep t.

Mechanism: Iterative vs. Static Sampling. The root cause of this drift lies in the distinct data-construction mechanisms. During training, x_t is sampled from a pre-defined static interpolation, ensuring a rigid bijection between t and SNR. Conversely, inference is an iterative process where x_t is the recursive output of preceding steps, so cumulative denoising progress can deviate from the fixed training schedule. Appendix A.4 formalizes this as a mismatch between nominal timestep and effective semantic demand.

Consequence: Semantic Lag and Joint Stability. This misalignment renders the time-wise gating network $g_\psi(t)$ ineffective. Conditioned solely on the nominal t, the gate remains oblivious to the fact that the latent x_t has already reached a "cleaner" state (semantic lag), thus rigidly injecting coarse-grained training priors that hamper high-frequency detail formation.

In contrast, the joint strategy (S3) mitigates this by exhibiting higher temporal stability. As visualized in Figure 8, S3 weights undergo much smaller variations across timesteps than the S1 weights. This inherent stability, achieved by coupling time with depth, effectively dampens the synchronization errors caused by trajectory misalignment, explaining why S3 remains robust while S1 falters; Appendix A.6 gives the corresponding allocation-sensitivity interpretation.

Appendix A.5 reports a shifted-timestep counterfactual that provides additional support for this mismatch interpretation.

6 Conclusion

We introduced Semantic Routing, a lightweight interface for multi-layer LLM conditioning in DiT-based generation that forms text features through normalized convex fusion of LLM hidden states. Under matched training and evaluation settings, depth-wise routing emerges as the most reliable strategy, delivering consistent improvements on text–image alignment and compositional benchmarks. Purely time-wise routing can degrade visual fidelity; our analysis attributes this to a mismatch between nominal timesteps and the effective inference trajectory, highlighting the need for trajectory-aware conditioning signals for robust time-dependent routing.

References

[1] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[2] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first international conference on machine learning*, 2024.

[3] Y. Gao, H. Guo, T. Hoang, W. Huang, L. Jiang, F. Kong, H. Li, J. Li, L. Li, X. Li *et al.*, "Seedance 1.0: Exploring the boundaries of video generation models," *arXiv preprint arXiv:2506.09113*, 2025.

[4] C. Wu, J. Li, J. Zhou, J. Lin, K. Gao, K. Yan, S.-m. Yin, S. Bai, X. Xu, Y. Chen *et al.*, "Qwen-image technical report," *arXiv preprint arXiv:2508.02324*, 2025.

[5] W. Kong, Q. Tian, Z. Zhang, R. Min, Z. Dai, J. Zhou, J. Xiong, X. Li, B. Wu, J. Zhang *et al.*, "Hunyuanvideo: A systematic framework for large video generative models," *arXiv preprint arXiv:2412.03603*, 2024.

[6] G. Ma, H. Huang, K. Yan, L. Chen, N. Duan, S. Yin, C. Wan, R. Ming, X. Song, X. Chen *et al.*, "Step-video-t2v technical report: The practice, challenges, and future of video foundation model," *arXiv preprint arXiv:2502.10248*, 2025.

[7] H. Cai, S. Cao, R. Du, P. Gao, S. Hoi, S. Huang, Z. Hou, D. Jiang, X. Jin, L. Li *et al.*, "Z-image: An efficient image generation foundation model with single-stream diffusion transformer," *arXiv preprint arXiv:2511.22699*, 2025.

[8] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang *et al.*, "Wan: Open and advanced large-scale video generative models," *arXiv preprint arXiv:2503.20314*, 2025.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[11] P. BehnamGhader, V. Adlakha, M. Mosbach, D. Bahdanau, N. Chapados, and S. Reddy, "Llm2vec: Large language models are secretly powerful text encoders," *arXiv preprint arXiv:2404.05961*, 2024.

[12] B. Ma, Z. Zong, G. Song, H. Li, and Y. Liu, "Exploring the role of large language models in prompt encoding for diffusion models," *arXiv preprint arXiv:2406.11831*, 2024.

[13] A. Z. Wang, S. Ge, T. Karras, M.-Y. Liu, and Y. Balaji, "A comprehensive study of decoder-only llms for text-to-image generation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 28 575–28 585.

[14] B. Li, S. Yang, Y. Guan, R. An, X. Chen, Y. Shi, P. Wan, W. Zhang *et al.*, "Gran-ted: Generating robust, aligned, and nuanced text embedding for diffusion models," *arXiv preprint arXiv:2512.15560*, 2025.

[15] Kling Team, D. Chen, J. Chen, X. Chen, Y. Dai, H. Dong, B. Geng, Y. Guan, M. Guo, G. He *et al.*, "Behind video generation: Observations and insights on multimodal understanding from Kling," *arXiv preprint arXiv:2611.04916*, 2026.

[16] Z. Liu, C. Kong, Y. Liu, and M. Sun, "Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics," *arXiv preprint arXiv:2403.01509*, 2024.

[17] M. Jin, Q. Yu, J. Huang, Q. Zeng, Z. Wang, W. Hua, H. Zhao, K. Mei, Y. Meng, K. Ding *et al.*, "Exploring concept depth: How large language models acquire knowledge and concept at different layers?" in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 558–573.

[18] S. Fan, X. Jiang, X. Li, X. Meng, P. Han, S. Shang, A. Sun, Y. Wang, and Z. Wang, "Not all layers of llms are necessary during inference," *arXiv preprint arXiv:2403.02181*, 2024.

[19] F. Barbero, A. Arroyo, X. Gu, C. Perivolaropoulos, M. Bronstein, P. Veličković, and R. Pascanu, "Why do llms attend to the first token?" *arXiv preprint arXiv:2504.02732*, 2025.

[20] O. Skean, M. R. Arefin, D. Zhao, N. Patel, J. Naghiyev, Y. LeCun, and R. Shwartz-Ziv, "Layer by layer: Uncovering hidden representations in language models," *arXiv preprint arXiv:2502.02013*, 2025.

[21] P. Chen, M. Shen, P. Ye, J. Cao, C. Tu, C.-S. Bouganis, Y. Zhao, and T. Chen, "δ-dit: A training-free acceleration method tailored for diffusion transformers," *ArXiv*, vol. abs/2406.01125, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270215326

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[23] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li, "Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2310.00426

[24] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[25] T. Seedream, Y. Chen, Y. Gao, L. Gong, M. Guo, Q. Guo, Z. Guo, X. Hou, W. Huang, Y. Huang *et al.*, "Seedream 4.0: Toward next-generation multimodal image generation," *arXiv preprint arXiv:2509.20427*, 2025.

[26] Z. Yuan, Y. Wang, D. Zhang, K. Cui, R. Chen, J. Tang, L. Sun, H. Yu, C. Qian, X. Chu *et al.*, "What if agents could imagine? reinforcing open-vocabulary HOI comprehension through generation," *(NeurIPS'26) Advances in Neural Information Processing Systems*, 2026.

[27] Z. Yuan, X. Qu, C. Qian, R. Chen, J. Tang, L. Sun, X. Chu, D. Zhang, Y. Wang, Y. Cai *et al.*, "Video-STAR: Reinforcing open-vocabulary action recognition with tools," in *(ICLR'26) The Fourteenth International Conference on Learning Representations*, 2025.

[28] Z. Yuan, C. Qian, J. Tang, R. Chen, Z. Song, L. Sun, X. Chu, Y. Cai, D. Zhang, and S. Li, "AutoDrive-R^2: Incentivizing reasoning and self-reflection capacity for VLA model in autonomous driving," in *(ICLR'26) The Fourteenth International Conference on Learning Representations*, 2025.

[29] W. Gurnee and M. Tegmark, "Language models represent space and time," *arXiv preprint arXiv:2310.02207*, 2023.

[30] B. Liu, E. Akhgari, A. Visheratin, A. Kamko, L. Xu, S. Shrirao, C. Lambert, J. Souza, S. Doshi, and D. Li, "Playground v3: Improving text-to-image alignment with deep-fusion large language models," *arXiv preprint arXiv:2409.10695*, 2024.

[31] B. Tang, B. Zheng, S. Paul, and S. Xie, "Exploring the deep fusion of large language models and diffusion transformers for text-to-image synthesis," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 28 586–28 595.

[32] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," *arXiv preprint arXiv:2208.01626*, 2022.

[33] E. Liu, X. Ning, Z. Lin, H. Yang, and Y. Wang, "Oms-dpm: Optimizing the model schedule for diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 21 915–21 936.

[34] B. Wang and J. J. Vastola, "Diffusion models generate images like painters: an analytical theory of outline first, details later," *arXiv preprint arXiv:2303.02490*, 2023.

[35] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[36] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[37] J. Kim, B. Lee, C. Park, Y. Oh, B. Kim, T. Yoo, S. Shin, D. Han, J. Shin, and K. M. Yoo, "Peri-ln: Revisiting normalization layer in the transformer architecture," *arXiv preprint arXiv:2502.02732*, 2025.

[38] B. Li, X. Xue, S. Yang, Y. Shi, X. Chen, Y. Guan, Y. Zhang, and W. Zhang, "The unseen bias: How norm discrepancy in pre-norm mllms leads to visual information loss," *arXiv preprint arXiv:2512.08374*, 2025.

[39] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, W. Ge, Z. Guo, Q. Huang, J. Huang, F. Huang, B. Hui, S. Jiang, Z. Li, M. Li, M. Li, K. Li, Z. Lin, J. Lin, X. Liu, J. Liu, C. Liu, Y. Liu, D. Liu, S. Liu, D. Lu, R. Luo, C. Lv, R. Men, L. Meng, X. Ren, X. Ren, S. Song, Y. Sun, J. Tang, J. Tu, J. Wan, P. Wang, P. Wang, Q. Wang, Y. Wang, T. Xie, Y. Xu, H. Xu, J. Xu, Z. Yang, M. Yang, J. Yang, A. Yang, B. Yu, F. Zhang, H. Zhang, X. Zhang, B. Zheng, H. Zhong, J. Zhou, F. Zhou, J. Zhou, Y. Zhu, and K. Zhu, "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025.

[40] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[41] B. Heo, S. Park, D. Han, and S. Yun, "Rotary position embedding for vision transformer," in *European Conference on Computer Vision*. Springer, 2024, pp. 289–305.

[42] A. Henry, P. R. Dachapally, S. S. Pawar, and Y. Chen, "Query-key normalization for transformers," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 4246–4253.

[43] X. Gong, W. Chen, T. Chen, and Z. Wang, "Sandwich batch normalization: A drop-in replacement for feature distribution heterogeneity," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2494–2504.

[44] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," *arXiv preprint arXiv:2111.02114*, 2021.

[45] B. Li, J. Zhang, Y. Ding, Y. Guan, B. Zeng, Y. Ji, X. Chen, Y. Shi, Y. Dai, Y. Wang *et al.*, "Human-centric image captioning with subject-centered spatial understanding," *arXiv preprint arXiv:2609.08300*, 2026.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[47] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[48] P. von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, D. Nair, S. Paul, W. Berman, Y. Xu, S. Liu, and T. Wolf, "Diffusers: State-of-the-art diffusion models," https://github.com/huggingface/diffusers, 2022.

[49] B. Li, Z. Lin, D. Pathak, J. Li, Y. Fei, K. Wu, T. Ling, X. Xia, P. Zhang, G. Neubig *et al.*, "Genai-bench: Evaluating and improving compositional text-to-visual generation," *arXiv preprint arXiv:2406.13743*, 2024.

[50] D. Ghosh, H. Hajishirzi, and L. Schmidt, "Geneval: An object-focused framework for evaluating text-to-image alignment," *Advances in Neural Information Processing Systems*, vol. 36, pp. 52 132–52 152, 2023.

[51] Y. Wang, Y. Zang, H. Li, C. Jin, and J. Wang, "Unified reward model for multimodal understanding and generation," *arXiv preprint arXiv:2503.05236*, 2025.

[52] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[53] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[54] D. Li, A. Kamko, E. Akhgari, A. Sabet, L. Xu, and S. Doshi, "Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation," 2024.

[55] B. Zhang, F. Moiseev, J. Ainslie, P. Suganthan, M. Ma, S. Bhupatiraju, F. Lebron, O. Firat, A. Joulin, and Z. Dong, "Encoder-decoder gemma: Improving the quality-efficiency trade-off via adaptation," *arXiv preprint arXiv:2504.06225*, 2025.

[56] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.

A A Semantic Information Allocation View

This section provides an analytical framework for understanding Semantic Routing through the lens of semantic supply and demand. Under this view, different LLM layers supply distinct types of semantic information, different DiT conditioning sites demand different semantic granularities, and routing serves to match supply to demand. The framework yields concrete insights into why depth-dependent routing outperforms static and time-only alternatives, why static fusion can underperform even uniform averaging, why time-wise routing suffers from inference-time degradation, how joint routing mitigates temporal drift, and why inter-layer suppression emerges under softmax routing.

A.1 The Supply–Demand Mismatch View

The core premise is that different LLM layers encode different types of semantic information, and different conditioning sites—indexed by flow time t and DiT block depth d—have heterogeneous semantic needs. Routing can be understood as matching layer-wise semantic supply to site-wise semantic demand. This view is an interpretive abstraction rather than a formal derivation from the hidden-state geometry: it assumes that layer-wise semantic profiles can be summarized by additive latent factors and that convex hidden-state fusion induces a corresponding convex mixture of those semantic profiles.

Semantic supply. Let a prompt contain latent semantic factors $Z = \{Z_1, \ldots, Z_K\}$ (e.g., object identity, attributes, counting, spatial relations, global layout, fine-grained details). We associate each LLM layer l with a supply vector $s_l \in \mathbb{R}_+^K$, where $s_l^{(k)}$ denotes how strongly layer l exposes factor Z_k. These vectors capture what linear probes on each layer would recover about each semantic factor; they characterize the semantic profile of the LLM hierarchy without being directly parameterized. Stacking layer-wise supplies as columns yields the supply matrix $Q = [s_1, \ldots, s_L] \in \mathbb{R}_+^{K \times L}$. Given a routing weight $\alpha_{t,d} \in \Delta^{L-1}$, the semantic supply routed to site (t, d) is

$$S_{t,d} = Q\alpha_{t,d} = \sum_{l=1}^{L} \alpha_{t,d}^{(l)}\, s_l, \tag{11}$$

i.e., a convex combination of layer-wise semantic profiles. Semantic Routing does not create new semantic information; it changes which parts of the LLM hierarchy are exposed to each conditioning site.

Semantic demand. Each conditioning site (t, d) has a semantic demand vector $R_{t,d} \in \mathbb{R}_+^K$ reflecting the type of text guidance it requires. For instance, early denoising stages and shallow DiT blocks may demand object identity and global layout, while later stages and deeper blocks may require fine-grained attributes and local relations. The routing problem can then be viewed as minimizing a supply–demand mismatch:

$$\mathcal{E}(\alpha) = \sum_{t,d} D\big(R_{t,d},\, Q\alpha_{t,d}\big), \tag{12}$$

where $D(\cdot, \cdot)$ is a divergence (e.g., squared Euclidean distance). Under this objective, the routing problem has a clear structure: when $R_{t,d}$ varies across sites, a single global α cannot simultaneously match all demands, and adaptivity along t or d reduces the feasible mismatch. The actual model is trained end-to-end with the generative loss, which implicitly drives the weights toward solutions that align with this supply–demand logic.

A.2 Why Static Fusion Underperforms Uniform Averaging

Static fusion (B3) learns a single global weight vector α, forcing $S_{t,d} = Q\alpha$ for all (t, d). Gran-TED [14] identifies a fundamental challenge for global learnable layer weights in diffusion models: because the training objective shifts from coarse structure (early denoising) toward fine details (late denoising), the optimal layer weighting changes continuously over the course of training [15]. A single global weight vector must therefore chase a moving target, producing non-stationary conditioning. B3 inherits this difficulty—although it does not condition on t, its weights are updated throughout training under a shifting loss landscape.

This drift creates two problems. First, training instability: as the learned weights α evolve, the conditioning input to the DiT backbone shifts, making the denoising objective itself a moving target for the generator. Second, stage-specific convergence: the final learned weights may converge to a configuration that was optimal only for a particular training stage (e.g., late-stage fine-detail refinement) rather than being broadly effective across the full denoising trajectory. The resulting α^{static} can therefore be actively harmful for timesteps or blocks whose demands differ from the dominant training stage.

This provides a possible mechanism for why static fusion (B3) can underperform even uniform averaging (B2) in practice. Under the idealized supply–demand objective in Section A.1, the optimal static solution includes uniform averaging as a feasible point and therefore should not be worse than uniform averaging under exact optimization. The actual end-to-end training problem, however, is non-convex, and the router co-adapts with

the DiT backbone. As the global learned weights evolve, the conditioning distribution seen by the generator also changes, which can introduce optimization instability and bias the learned static weights toward dominant flow-time regions or block demands. In such cases, static fusion may converge to a poor global compromise, while uniform averaging remains a fixed and stable prior. Let $\bar{\alpha} = \mathbf{1}/L$ denote the uniform vector, and let $\alpha^*_{t,d}$ denote the per-site optimal weights. Since correlated neighboring layers can make raw weight-space distance a poor proxy for semantic mismatch, we compare the induced routed supplies. If $\alpha^*_{t,d}$ varies substantially across sites, it is possible that the learned static weights induce a larger routed-supply mismatch than the uniform prior:

$$\mathbb{E}_{t,d}\big[\|Q(\alpha^*_{t,d} - \bar{\alpha})\|_2^2\big] \;<\; \mathbb{E}_{t,d}\big[\|Q(\alpha^*_{t,d} - \alpha^{\text{static}})\|_2^2\big] \tag{13}$$

This can occur when α^{static} overfits to dominant conditioning demands and starves minority demands. Thus, uniform averaging can be safer than a learned global weight when the latter is affected by optimization drift and heterogeneous conditioning demands.

A.3 Depth-wise Routing and Per-Block Specialization

Different DiT blocks at different depths are responsible for different aspects of the generation process: prior work on DiT internals suggests that shallower blocks tend to handle coarse spatial layout while deeper blocks refine fine-grained details. This functional specialization has a direct consequence for optimization dynamics. Although all DiT blocks participate in the forward and backward passes for each sampled flow time, their effective contribution to the denoising objective is not uniform across flow time. Coarse-structure regions of the denoising trajectory tend to rely more on shallower blocks, whereas later, detail-refinement regions tend to rely more on deeper blocks.

Under depth-wise routing—which assigns each block d its own weight vector α_d via $S_{t,d} = Q\alpha_d$—each α_d is updated through the gradient pathway of its own block. Even though this update aggregates signals from all sampled flow times, the effective optimization pressure is filtered by the functional role of block d: flow-time regions where block d contributes more strongly to the loss provide stronger gradient signals for its router. In contrast, static fusion uses a single α that receives aggregate signals from all blocks and all flow-time regions, making it more likely to become a compromise across heterogeneous demands. The result is that each α_d can converge to a block-appropriate configuration, rather than being pulled toward a single global compromise.

This view also makes precise the sense in which depth-wise routing is a relaxation of static fusion. Under a squared supply–demand mismatch, forcing one α to serve all blocks yields:

$$\min_{\alpha \in \Delta^{L-1}} \sum_d \|R_d - Q\alpha\|_2^2 \;\geq\; \sum_d \min_{\alpha_d \in \Delta^{L-1}} \|R_d - Q\alpha_d\|_2^2, \tag{14}$$

where R_d denotes the time-averaged semantic demand of block d. The inequality follows from the fact that separate minimization per block is more flexible than minimization with a shared parameter. Combined with the weak per-block drift induced by functional specialization, this relaxed feasible set enables each α_d to stably converge to a block-appropriate configuration, providing a structural explanation for the empirical improvement of S2 over B3.

A.4 Time-wise Routing and the Inference Discrepancy

Time-wise routing (S1) assumes that the nominal timestep t is a reliable proxy for semantic demand: $\alpha_{t,d} = \alpha(t)$, $S_{t,d} = Q\alpha(t)$. We identify a fundamental failure mode of this assumption at inference time.

Effective timestep mismatch. During training, the latent x_t is sampled directly from the forward process: $x_t \sim p_t$, where p_t is the marginal at true progress t. The time gate g_ψ is therefore optimized to supply $\alpha(t)$ when the latent has the statistical properties of p_t. During inference, however, the latent $\hat{x}_t$ is produced recursively by an ODE solver. Due to discretization error and cumulative denoising drift, $\hat{x}_t$ does not follow the training marginal p_t. We define the effective timestep $\tau(t)$ as the nominal timestep at which the training distribution best matches the inference latent:

$$\tau(t) = \arg\min_{s \in [0,1]} \; D_{\text{latent}}\big(p_s,\; \text{dist}(\hat{x}_t)\big), \tag{15}$$

where D_{latent} measures distributional discrepancy. The empirical measurements in Figure 3(c,d) confirm that $\hat{x}_t$ is systematically "ahead" of p_t, i.e., $\tau(t) > t$ (the latent is cleaner than the nominal timestep would suggest).

Sensitivity to timing error. When $\tau(t) \neq t$, the gate supplies $\alpha(t)$ while the latent actually demands $\alpha(\tau(t))$. The resulting conditioning shift is bounded by the local variation of $\alpha(\cdot)$:

$$\|\alpha(t) - \alpha(\tau(t))\|_1 \;\leq\; \sup_{\xi \in [0,1]} \|\alpha'(\xi)\|_1 \cdot |t - \tau(t)|. \tag{16}$$

Two factors amplify the mismatch: (i) the magnitude of the timing error $|t - \tau(t)|$, which is determined by solver accuracy and the inherent difficulty of the denoising trajectory; and (ii) the rate of change $\|\alpha'(\xi)\|_1$,

which is a property of the learned time gate. When $\alpha(t)$ changes rapidly across timesteps—as observed in the 1-Wasserstein measurements in Figure 8—even a small effective-time mismatch can produce a large weight deviation, which in turn shifts the routed supply $Q\alpha$ and degrades conditioning. This mechanism explains the blur and detail loss observed in S1: the time gate injects semantic features calibrated for p_t into a latent that has already progressed to $\tau(t)$, creating a semantic lag that interferes with fine-grained generation.

A.5 Shifted-Timestep Counterfactual

The sensitivity view above motivates a counterfactual check: if S1 suffers from a mismatch between the nominal timestep and the effective inference trajectory, then advancing the timestep input to the TCFG should reduce the semantic lag and improve conditioning alignment. We therefore apply a heuristic shift function modulated by a cosine window active in $t \in (0.2, 1]$:

$$t' = \mathrm{clip}_{[0,1]}\big(t + \delta(t)\big), \quad \delta(t) = 0.005 \cdot \left(1 - \cos\left(\pi \cdot \frac{t - 0.2}{0.8}\right)\right). \tag{17}$$

Equation (16) explains why this intervention can help: advancing the nominal input from t to the clipped value t' reduces the mismatch $|t - \tau(t)|$ when the inference trajectory is ahead of the nominal schedule, bringing the gate output closer to what the latent actually needs while keeping the timestep within the valid range.

Table 3: Effect of manual timestep recalibration on S1 and S3. S1 shows consistent recovery, while S3 is largely unchanged.

Method	GenEval ↑	GenAI ↑	UnifiedReward ↑
S1	63.41	76.20	2.97
S1 + Shift	**63.65**	**76.46**	**2.98**
S3	66.05	77.44	3.06
S3 + Shift	66.00	77.65	3.06

As shown in Table 3, the shifted timestep yields consistent recovery for S1 across all metrics, including a $+0.24$ gain on GenEval. The systematic direction of this recovery—every metric improves—is consistent with the mismatch interpretation. In contrast, S3 shows no clear improvement under the same shift intervention: GenEval changes from 66.05 to 66.00, and the Style Score is unchanged. This asymmetry is consistent with the temporal stabilization view in Section A.6: S3's per-block time gates produce smoother weight trajectories with smaller $\|\alpha_d'(\xi)\|_1$, so the same shift $\delta(t)$ induces a smaller weight deviation and therefore less measurable change. The contrasting shift responses of S1 and S3 provide supporting evidence for the sensitivity interpretation in Equation (16).

Since the fixed heuristic shift can only linearly approximate the full nonlinear trajectory deviation, the recovery magnitude for S1 also points to a natural next step: learning the effective-time mapping, or designing trajectory-aware routing signals, could close the remaining gap.

A.6 Joint Routing as Temporal Stabilization

Joint routing (S3) assigns each DiT block d its own time-conditioned gate: $\alpha_{t,d} = \alpha_d(t)$. This separates two sources of variation: block-specific demand (captured by per-block parameterization) and time-dependent demand (captured by the temporal gate). The key empirical finding is that joint weights evolve more smoothly over time than purely time-wise weights (Figure 8). We attribute this to a division of labor: when per-block weighting already handles coarse block-specific preferences, the temporal gate only needs to model the residual time variation within each block, reducing $\|\alpha_d'(t)\|_1$ for each d.

By Equation (16), this directly reduces sensitivity to the effective-timestep mismatch. The same timing error $|t - \tau(t)|$ produces a smaller weight deviation, yielding more stable conditioning along the inference trajectory. A direct empirical test of this prediction is provided by the shifted-timestep counterfactual in Table 3: while the same shift $\delta(t)$ measurably improves S1, it leaves S3 essentially unchanged, confirming that joint routing's smoother temporal profiles translate to reduced sensitivity to timing perturbation.

We note that this argument does not imply joint routing must outperform depth-wise routing. If the dominant demand variation is depth-related (as suggested by S2's strong performance), and if the residual time signal is noisy or misaligned, the added temporal flexibility may introduce more fragility than benefit. The relative performance of S2 and S3 depends on whether the time-axis information is sufficiently reliable to overcome this overhead.

A.7 Inter-Layer Suppression as Information Maximization

The peak-and-valley weight pattern observed in Figure 2—where specific layers receive high weights while their immediate neighbors are suppressed—admits a natural interpretation through the lens of information maximization under a capacity constraint.

The key structural fact is that adjacent LLM layers carry highly correlated representations due to residual connections: the hidden state at layer $l+1$ is $H^{(l+1)} \approx H^{(l)} + F^{(l)}(H^{(l)})$, so neighboring layers share a large common component. When the router assigns weight to two highly similar layers, the marginal information gain from the second layer is small—it largely duplicates what the first layer already provides. However, because the fusion weights are constrained to sum to one ($\sum_l \alpha^{(l)} = 1$ via the softmax), every unit of weight allocated to a near-duplicate layer is a unit of weight taken away from a potentially more informative, semantically distinct layer.

Under this view, inter-layer suppression is an efficient selection strategy: the router concentrates weight on one representative layer within each local neighborhood of high inter-layer similarity, and suppresses the rest, thereby maximizing the semantic diversity of the fused representation under the fixed weight budget. This produces the characteristic peak-and-valley pattern, where each peak corresponds to a maximally informative representative for its local neighborhood. The pattern is more pronounced in joint routing (Figure 2a) because the joint scheme has greater capacity to tailor the selection to each DiT block's specific information needs.

B TCFG Implementation Details

The Time-Conditioned Fusion Gate (TCFG) serves as the core module for adaptively aggregating multi-layer LLM features based on the diffusion timestep. Our implementation follows a lightweight Multi-Layer Perceptron (MLP) design with specific initialization strategies to ensure training stability.

Sinusoidal Timestep Embedding. The continuous timestep t is first mapped to a high-dimensional feature vector using a sinusoidal embedding, similar to the positional encoding in Transformers. For a time embedding dimension D_t, the embedding $\phi(t) \in \mathbb{R}^{D_t}$ is computed as:

$$\phi(t) = [\dots, \cos(t \cdot \omega_i), \sin(t \cdot \omega_i), \dots], \quad \text{where } \omega_i = \frac{1}{10000^{2i/D_t}}. \tag{18}$$

In our experiments, we set $D_t = 128$.

Gating Network Architecture. The embedding $\phi(t)$ is processed by a two-layer MLP to generate the fusion logits $z_t \in \mathbb{R}^L$, where L is the number of LLM layers. The network structure consists of:

1. **Input Projection:** A linear layer mapping from D_t to a hidden dimension of $4 \times D_t$.
2. **Activation:** A Sigmoid Linear Unit (SiLU) activation function.
3. **Output Projection:** A linear layer mapping from $4 \times D_t$ to L.

Zero-Initialization Strategy. To facilitate a smooth starting point for optimization, we employ a zero-initialization strategy for the final output projection layer. Specifically, both the weight matrix and the bias vector of the second linear layer are initialized to zero. Consequently, at the beginning of training, the output logits z_t are zero vectors, which results in a uniform probability distribution after the Softmax operation (i.e., $\alpha_t^{(l)} = 1/L$ for all l). This ensures that the model initially utilizes an average of all layers before learning specific routing preferences.

Feature Normalization and Aggregation. Given a set of hidden states $\{H^{(l)}\}_{l=1}^{L}$ from the text encoder, we strictly apply Layer Normalization (LayerNorm) *before* fusion to handle scale discrepancies across layers. The final aggregated feature H_{cond} is computed as:

$$H_{\text{cond}} = \sum_{l=1}^{L} \text{Softmax}(z_t)^{(l)} \cdot \text{LayerNorm}(H^{(l)}). \tag{19}$$

This formulation ensures that the aggregated feature remains within the convex hull of the normalized representations, maintaining numerical stability throughout the training process.

C Latency Measurement Details

Latency is not the central contribution of this paper; we report it as a supplementary diagnostic of the runtime overhead introduced by routing. All measurements are conducted on the same single-GPU machine with an

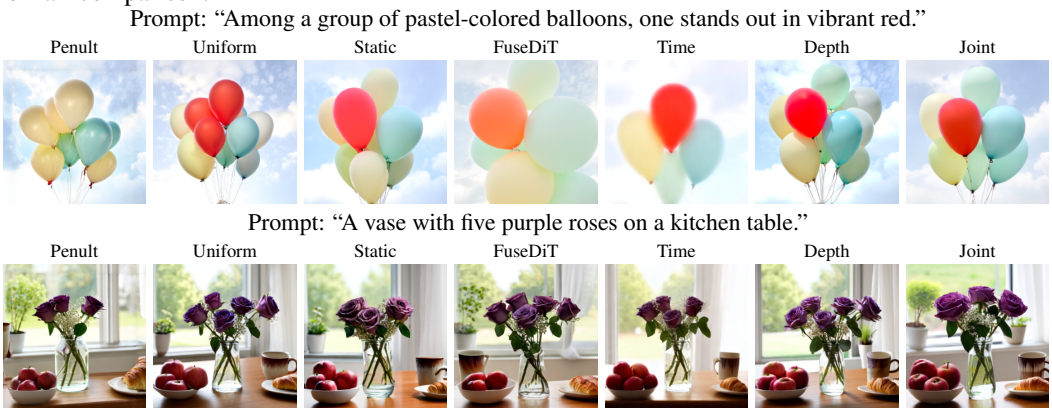
NVIDIA H800 GPU, 2 × Intel Xeon Platinum 8558 CPUs, and 2.0 TiB system memory. We use a fixed text length of 512 tokens, 50 sampling steps, batch size 1, bfloat16 inference precision, PyTorch 2.6.0+cu126, diffusers 0.32.0, and transformers 4.57.1. The reported numbers are based on 30 repeated generations and include standard deviations. They reflect the current Python-level implementation, without kernel-level optimization tailored to the routing module.

Table 4: Inference latency measured over 30 runs. We report mean ± standard deviation in milliseconds; "compile" denotes `torch.compile()`. Lower is better.

Method	Without compile	With compile
	256×256 *resolution*	
B1: Penult.	1425.26 ± 4.73	491.55 ± 2.41
B2: Uniform	1457.79 ± 4.41	513.74 ± 1.81
B3: Static	1445.67 ± 7.67	504.94 ± 2.27
FuseDiT	1362.09 ± 4.72	459.42 ± 1.92
S1: Time	1447.59 ± 2.59	508.58 ± 1.34
S2: Depth	1534.99 ± 4.02	547.72 ± 2.21
S3: Joint	1543.15 ± 5.39	551.47 ± 1.33
	512×512 *resolution*	
B1: Penult.	2246.12 ± 0.78	1176.20 ± 0.97
B2: Uniform	2290.28 ± 0.49	1208.93 ± 1.89
B3: Static	2276.01 ± 1.17	1201.05 ± 1.68
FuseDiT	2117.92 ± 2.30	1127.87 ± 5.50
S1: Time	2297.58 ± 0.93	1200.75 ± 2.18
S2: Depth	2380.99 ± 0.67	1241.34 ± 1.90
S3: Joint	2398.33 ± 1.46	1249.59 ± 1.83

D Image Examples

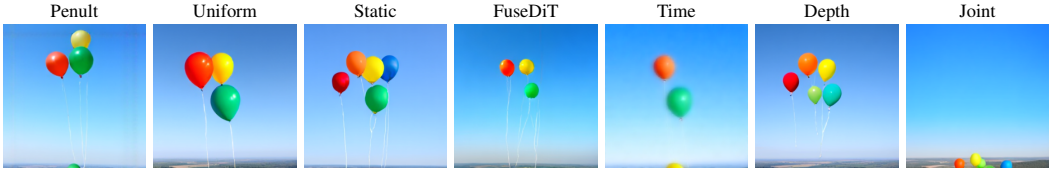
Figure 4: Qualitative comparisons across strategies under multiple prompts. Columns: B1/B2/B3, FuseDiT baseline, and three fusion strategies (S1/S2/S3). All images use identical sampling settings for fair comparison.



Prompt: “A pilot with aviator sunglasses.”



Prompt: “Five colorful balloons floating against a clear blue sky.”



Prompt: “A large pizza with pepperoni on the left half and mushrooms on the right half.”



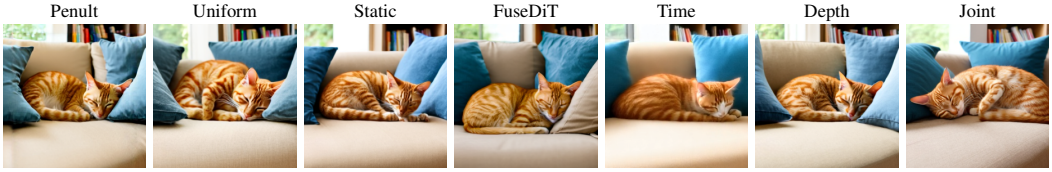
Prompt: “A large teddy bear wearing a bow tie next to a small teddy bear wearing a party hat.”



Prompt: “A small dog in a cozy orange sweater sitting beside a cat wearing a stylish blue bow tie.”



Prompt: “An orange cat lies on a couch surrounded by three pillows that are all blue.”



Prompt: “Three ceramic cups sit to the right of a wooden fork.”



Prompt: “Three puppies are standing by the pool and the one in the middle looks more excited than the other two puppies.”



E Additional Visualizations of Fusion Weight

In this section, we provide more detailed visualizations of the fusion weights (App. E.1) and an analysis of their variation trends (App. E.2).

E.1 Detailed Visualization of Fusion Weights

Figures 5 and 6 visualize the fusion-weight distributions of the text encoder across different layers and diffusion timesteps under the depth-wise and joint strategies, respectively.

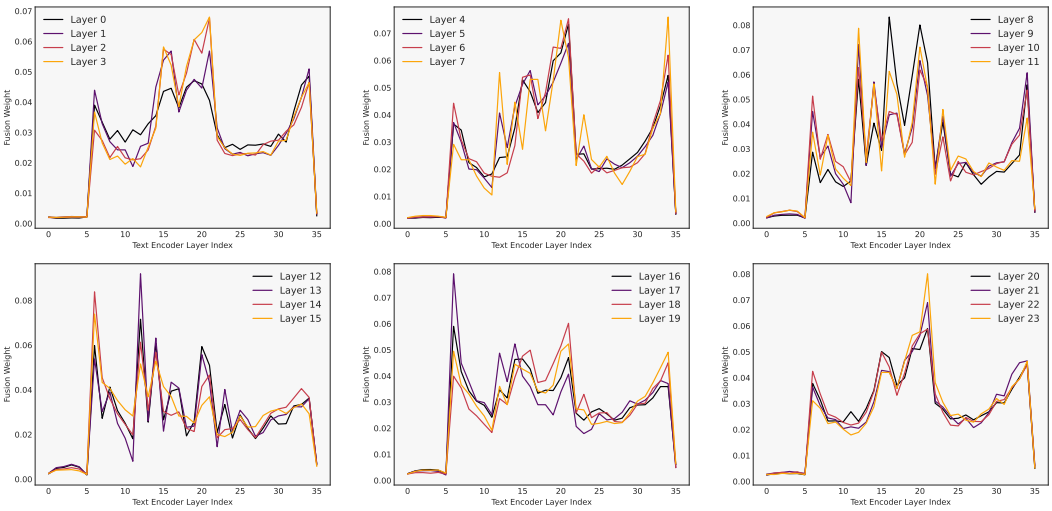


Figure 5: Depth-wise fusion weights across layers.

E.2 Weight Evolution and Trend Analysis

To quantify how the fusion weights over text-encoder layers evolve under different diffusion timesteps and depth settings, we treat each fusion-weight vector as a discrete probability distribution supported on uniformly spaced points over $[0, 1]$. We then compute its mean and variance as summary statistics, and plot their trends in Figure 7.

Given a fusion-weight vector $\mathbf{w} = \{w_i\}_{i=0}^{L-1}$ of length L, we first normalize it as

$$p_i = \frac{w_i}{\sum_{j=0}^{L-1} w_j + \epsilon}, \tag{20}$$

where ϵ is a small positive constant for numerical stability.

We define the uniformly spaced support locations by

$$l_i = \begin{cases} \frac{i}{L-1}, & L > 1, \\ 0, & L = 1, \end{cases} \qquad i \in \{0, 1, \ldots, L-1\}. \tag{21}$$

The mean (semantic center) is defined as

$$\hat{\mu} = \sum_{i=0}^{L-1} p_i \, l_i, \tag{22}$$

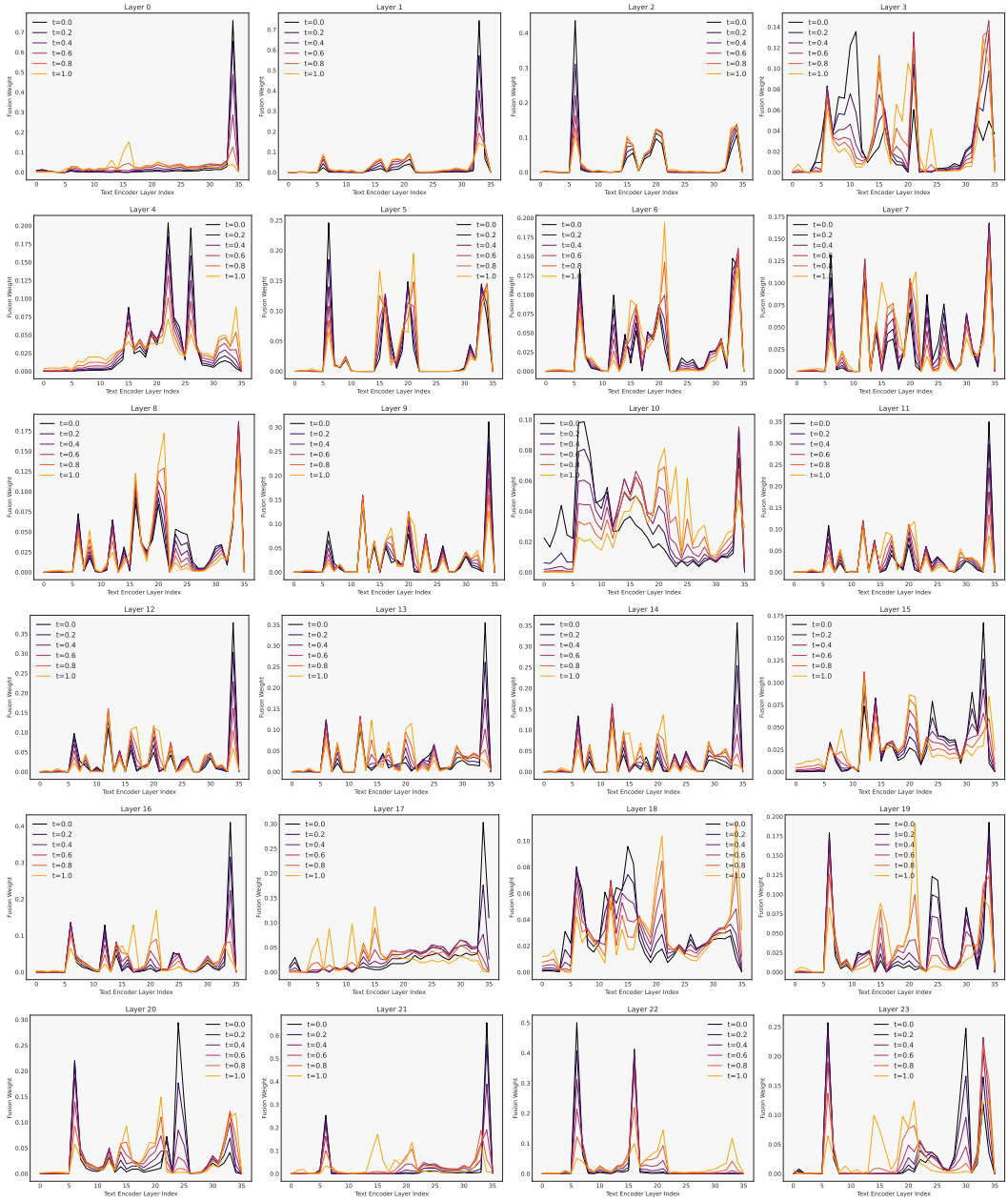


Figure 6: Joint fusion weights (depth $\times$ time) across layers.

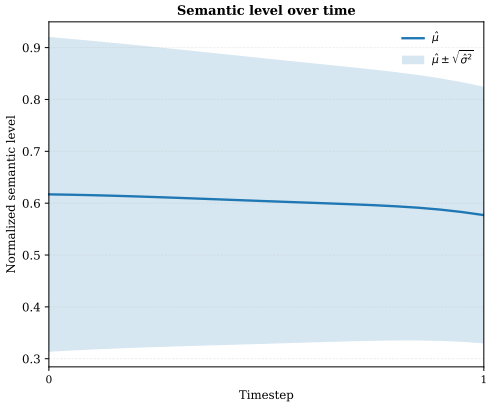
and the variance (semantic dispersion) is defined as

$$\hat{\sigma}^2 = \sum_{i=0}^{L-1} p_i \, (l_i - \hat{\mu})^2. \tag{23}$$

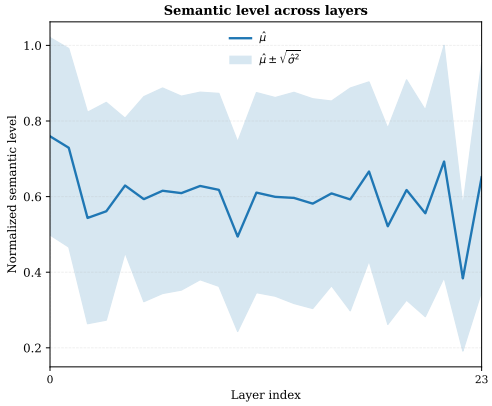
Intuitively, this interpretation views fusion weights as a distribution along a "semantic hierarchy" axis: $\hat{\mu}$ indicates where the mass concentrates on the axis (favoring earlier vs. later layers), while $\hat{\sigma}^2$ measures how dispersed the weights are (larger values imply a less concentrated, more spread-out allocation).

To better understand how fusion weights are allocated across diffusion timesteps and encoder depth, we visualize the statistics in Figure 7. Under the joint strategy, each sample yields a 2D weight map $w(t, l)$ over timestep t and text-encoder layer l. To obtain interpretable 1D trends, we form marginal distributions by normalizing along the complementary axis and compute the semantic center $\hat{\mu}$ and dispersion $\hat{\sigma}^2$ on each marginal.

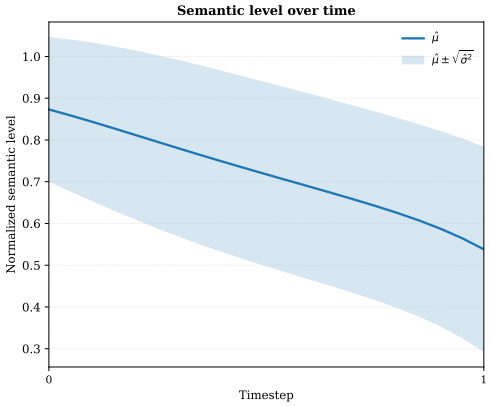
The results suggest that joint fusion maintains a largely stable semantic center over time: $\hat{\mu}$ stays at a moderately deep-layer regime and only shifts slightly toward shallower layers at later timesteps, while the consistently wide band indicates that the weights do not collapse onto a few layers but preserve broad multi-layer coverage throughout sampling. In contrast, time-wise fusion exhibits pronounced monotonic drift: $\hat{\mu}$ decreases substantially as timesteps progress, indicating a systematic shift from deeper-layer emphasis toward shallower-layer emphasis, which reflects a more time-dependent and less stable allocation of semantic levels. Depth-wise fusion yields smoother curves with smaller fluctuations, suggesting a more consistent and controllable allocation pattern across layers. Overall, joint fusion preserves broad layer coverage while substantially mitigating the deep-to-shallow drift observed in purely time-wise fusion, providing a statistical explanation for its more stable generation behavior.



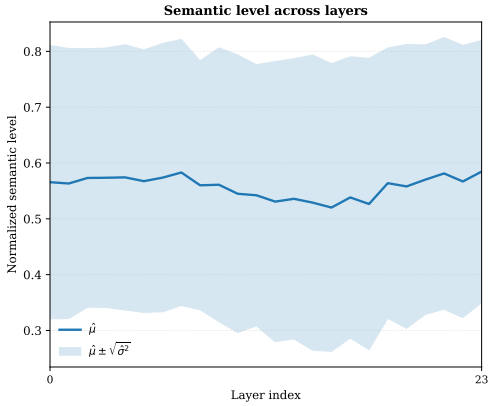
(a) Joint fusion: mean and variance computed on the time-marginal weight distribution.



(b) Joint fusion: mean and variance computed on the depth-marginal weight distribution.



(c) Time-wise fusion: mean and variance trends of the fusion weights across diffusion timesteps.

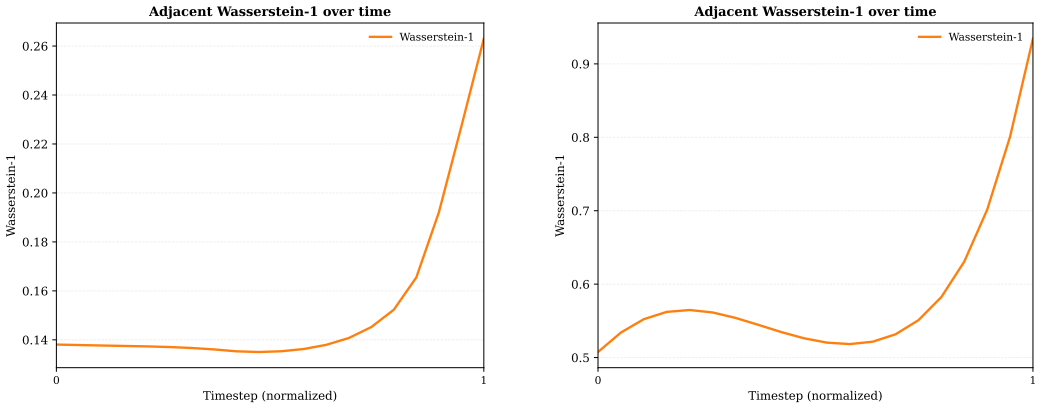


(d) Depth-wise fusion: mean and variance trends of the fusion weights across text-encoder layers.

Figure 7: Weight-trend statistics under different fusion strategies. Top row: joint fusion with marginalization over timesteps (left) and depths (right). Bottom row: the corresponding statistics under time-wise (left) and depth-wise (right) fusion.

To compare how fast the fusion weights evolve over time under the time-wise and joint strategies, and to further complement the analysis in Section 5.2, we quantify the discrepancy between weight distributions at consecutive timesteps. Concretely, for each sampled timestep t, we treat its corresponding weight vector as a distribution and compute the 1-Wasserstein distance to the distribution at the previous timestep $t-1$. For the joint strategy, since the weights are defined over both timesteps and layers, we first marginalize over the layer dimension to obtain a 1D marginal distribution for each timestep, and then compute the 1-Wasserstein distance between consecutive timesteps. We sample 21 timesteps in total, and report the results in Figure 8.

As shown in Figure 8, the joint strategy yields consistently smaller 1-Wasserstein distances between consecutive timesteps than the time-wise strategy, indicating a substantially smoother temporal evolution of the weight allocation. In light of the train–inference timestep-trajectory mismatch discussed in Section 5.2, a simple interpretation is as follows: under a time-wise-only scheme, rapid weight changes across adjacent timesteps



(a) Joint fusion (time-marginal): 1-Wasserstein distance between consecutive timesteps.

(b) Time-wise fusion: 1-Wasserstein distance between consecutive timesteps.

Figure 8: Temporal variation speed of fusion weights, measured by the 1-Wasserstein distance between weights at timesteps spaced by 0.05. For joint fusion, we compute the distance on the time-marginal distributions.

can amplify the mismatch-induced deviation, making the semantic conditioning less stable during inference and thus more prone to blur; joint fusion markedly slows down such temporal drift, leading to more coherent conditioning along the inference trajectory and mitigating the resulting degradation.

F Supplementary Experiments

This section reports additional experiments evaluating cross-encoder generalization.

F.1 Cross-Encoder Validation with T5Gemma

To test whether the conclusions generalize beyond the Qwen3-VL encoder family, we conduct a stress-test experiment with a substantially different text encoder. We replace Qwen3-VL-4B with T5Gemma-2B [55], an encoder-decoder model built on Gemma2-2B [56] and further trained on 8T tokens. The DiT backbone is scaled up to 3.95B parameters. Due to resource constraints, models are trained for 250k steps (vs. 500k in the main setting). We compare Baseline (penultimate layer), Norm-Avg (uniform averaging), and Depth (depth-wise routing). Evaluation follows the same protocol as the main experiments (4 images per prompt, seed 42).

Table 5: Cross-encoder validation with T5Gemma-2B encoder and 3.95B DiT backbone (250k steps).

Method	GenEval ↑	GenAI ↑	Style Score ↑	FID ↓
Baseline	36.58	42.74	2.89	31.76
Norm-Avg	34.68	41.64	2.83	30.23
Depth	**41.12**	**50.05**	2.89	**29.40**

The absolute scores are lower than in the main setting because models are trained for only 250k steps with a smaller 2B text encoder. Two observations are notable. First, Norm-Avg no longer exceeds Baseline in this setting; a plausible explanation is that T5Gemma, as an encoder-decoder model trained at very large scale, organizes layer-wise features differently from the decoder-only LLM in our main setup, making uniform averaging less suitable as a default. Second, Depth-wise routing remains the strongest variant, with gains on GenEval and GenAI-Bench over Baseline, supporting the robustness of the design conclusion across encoder architectures and model scales.

G Limitations and Broader Impact

Limitations. The routing module introduces a modest increase in inference latency (Table 4), which may be a consideration for latency-sensitive deployments. As with most T2I evaluation protocols, the alignment

benchmarks we use (GenEval, GenAI-Bench) rely on VLM-based judges whose assessments may not capture all dimensions of human preference. Finally, our analysis focuses on a specific set of fusion-weight parameterizations.

Broader Impact. This work aims to improve text-conditioned image generation, which has positive applications in creative tools, accessibility, and content editing. The proposed routing mechanism is lightweight and operates on frozen LLM features, making it compatible with existing safety and compliance practices. As with other advances in T2I generation, improved text–image alignment could be misused to produce misleading or manipulated imagery. However, our method does not introduce fundamentally new capabilities or risks beyond those already present in the underlying generative models. Standard mitigation strategies—including content safety filtering, sensitive concept blocking, and provenance mechanisms—remain applicable. The computational resources required for training (64 H200 GPUs, 500k steps) carry an associated energy cost, though the routing module itself adds negligible overhead relative to the base DiT training budget.

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 state that we study multi-layer LLM feature fusion for DiT conditioning, compare routing strategies under matched settings, and find depth-wise routing to be the most reliable. These claims are directly supported by the experimental results in Section 4.2 and the analysis in Section 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: Appendix G includes a dedicated Limitations paragraph discussing evaluation scale, inference latency overhead, VLM-based judge limitations, and the scope of routing parameterizations studied.

   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. The appendix provides an analytical framework (supply–demand view, sensitivity bounds, softmax gradient analysis) to interpret the empirical findings, but these are not presented as theorems with formal proofs.

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: Section 4.1 provides full training details (optimizer, learning rate, batch size, training steps, prompt drop ratio, timestep sampling distribution). The DiT backbone architecture, TCFG module design, and evaluation protocol are specified in Sections 3 and 4.1, with additional implementation details in Appendix B. The dataset is a publicly available subset of LAION-400M.

   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: [Yes]

   Justification: Our code and instructions needed to reproduce the main experiments are publicly available at `https://github.com/zooblastlbz/SemanticRouting`. Pretrained models (Qwen3-VL-4B, SD3 VAE) are publicly available. The training data (a processed subset of LAION-400M with synthetic captions) is not publicly released.

   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: Section 4.1 specifies the optimizer (AdamW), learning rate (1×10^{-4}), weight decay (1×10^{-4}), batch size (512), training steps (500k), prompt drop ratio (0.1), timestep sampling (logit-normal), image resolution (256×256), and text token length (512). Evaluation details (CFG scale, sampling steps, scheduler, number of images per prompt, random seed) are also provided.

   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: [N/A]

   Justification: In our benchmark evaluation, we employ standardized configurations and deterministic evaluation protocols. Since the primary metrics are alignment scores on fixed benchmark sets under a fixed random seed, reporting statistical dispersion measures such as standard deviation, variance, or error bars is not applicable.

   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: Section 4.1 states that all training is conducted on 64 H200 GPUs. Appendix C reports inference latency on a single H800 GPU with full hardware specifications. Each model variant requires approximately 500k training steps with batch size 512.

   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: This research uses publicly available datasets (LAION-400M) and pretrained models (Qwen3-VL, SD3 VAE). No human subjects were involved. We have reviewed the NeurIPS Code of Ethics and believe our work conforms to it in all respects.

   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: Appendix G discusses both positive impacts (improved controllability in creative tools, accessibility, content editing) and negative impacts (potential misuse for misleading imagery, amplification of training data biases). Standard mitigation strategies are noted.

    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: [Yes]

    Justification: Consistent with Question 5, our public repository (`https://github.com/zooblastlbz/SemanticRouting`) includes the code, reproduction instructions, and released model weights. We do not release the training data. The release documents the intended research use of the weights and the standard safety practices applicable to the underlying text-to-image generative models, including use of content safety filtering and compliance with the terms of the base models.

    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 existing assets used in this work are properly cited, including LAION-400M [44], Qwen3-VL [39], the Stable Diffusion 3 VAE [2], the diffusers library [48], GenAI-Bench [49], GenEval [50], and DrawBench [52]. We use these assets in accordance with their respective licenses and terms of use.

    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: [Yes]

    Justification: Our public repository (`https://github.com/zooblastlbz/SemanticRouting`) provides the training code, released model weights, and documentation for reproducing the main experiments.

    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: This paper does not involve crowdsourcing or research with human subjects.

    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: This paper does not involve crowdsourcing or research with human subjects.

    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: LLMs are not used as a core, original, or non-standard component of our methodology; any LLM usage is limited to writing assistance and does not impact the scientific rigor or originality of the research.

    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.