Where to Connect? Boosting MLLMs via Dynamic Gated Pathways across ALL ViT and LLM Layers

Yingying Yan[1,*] **Jiaqi Tang**[2,*] **Wei Wei**[1,†] **Qianzhou Wang**[1] **Lei Zhang**[1]
Jianmin Chen[1] **Yuyang Xia**[1] **Botong Geng**[1] **Jinjian Wu**[1] **Qifeng Chen**[2]

[1]Northwestern Polytechnical University, Xi'an, China
[2]Hong Kong University of Science and Technology, Hong Kong, China

{yanyyy, wangqianzhou, chenjianmin, wujinjian}@mail.nwpu.edu.cn
{weiweinwpu, nwpuzhanglei}@nwpu.edu.cn
jtang092@connect.ust.hk, cqf@ust.hk

Abstract

A central yet under-explored question in Multimodal Large Language Models (MLLMs) is *where to connect*—how visual information from a hierarchical vision encoder should be wired into the layer-wise semantics of a large language model. Mainstream MLLMs inject a fixed single-layer visual representation into the LLM and treat all LLM layers as a single consumer, ignoring that different layers demand different visual granularities. Recent multi-layer fusion methods either compress hierarchical features on the encoder side or rely on predefined sparse or hierarchical layer-to-layer connections, and therefore stop short of modeling *adaptive* cross-layer interactions between the ViT and LLM hierarchies. We address this gap with DGP (Dynamic Gated Pathways), a routing mechanism that establishes *all-to-all* connections between every ViT layer and every LLM layer, with each LLM layer dynamically gating its preference over visual layers conditioned on its own semantic state. Built on top of LLaVA-1.5-7B and 13B, DGP delivers consistent gains over the LLaVA-1.5-7B baseline (e.g., +4.2 on SQA^I, +3.3 on MM-Vet, +3.1 on VizWiz and $LLaVA^W$) and surpasses prior multi-layer visual fusion methods on most benchmarks. Layer-wise routing analyses further show that different LLM layers do select different visual granularities on demand, supporting both the effectiveness and interpretability of the proposed dynamic gated pathways. The code is available: https://github.com/YingyingYan1/DGP.

1 Introduction

Multimodal Large Language Models (MLLMs) [3, 16, 19, 21] encode an image with a pretrained vision backbone [31, 32, 42], project the resulting features through an adapter, and inject the visual tokens into a language model. In this paradigm, the effectiveness of visual information depends not only on the representational power of the encoder [7, 34, 38], but—more critically—on *where to connect*: how the visual tokens are wired into the internal layers of the LLM.

As illustrated in Figure 1, attention patterns across ViT layers differ markedly: shallow layers attend to edges, textures, and local details, whereas deeper layers progressively concentrate on a small number of semantic anchors. Yet, most MLLMs [2, 10, 26] simply forward the output of one fixed ViT layer (typically the last or penultimate layer) to the LLM, as illustrated in Figure 2(A), effectively

[*]Equal Contribution.
[†]Corresponding author: Wei Wei.

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

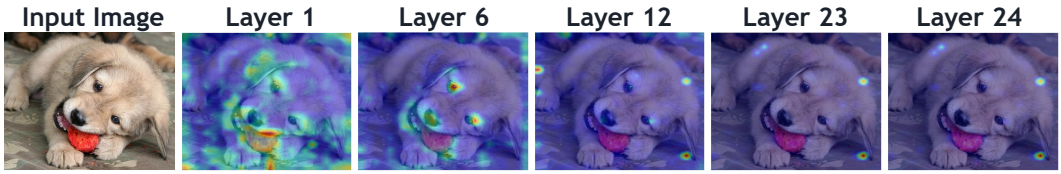


Figure 1: **Attention distributions across ViT layers.** Shallow layers show dispersed attention over edges and textures; deeper layers progressively concentrate on a few semantic anchors.

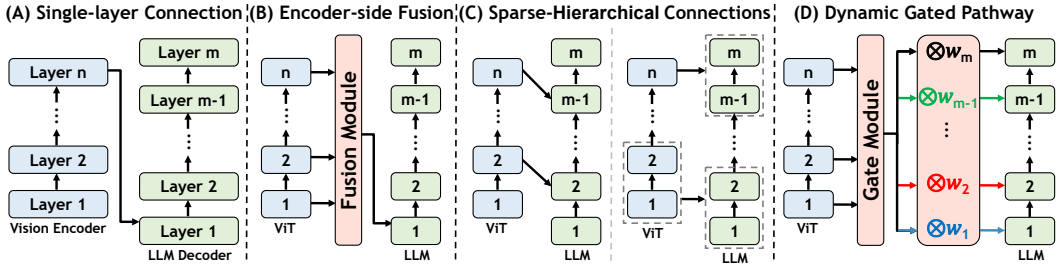


Figure 2: **Strategies for visual token insertion.** (A) Fixed single-layer connection. (B) Encoder-side fusion of multiple ViT layers into one representation. (C) Predefined sparse or hierarchical connections between ViT and LLM layers. (D) DGP (Ours): dynamic gated pathways between *all* ViT and *all* LLM layers. w_m denotes the routing weights that LLM layer m assigns across *all* ViT layers.

collapsing the image into a single-granularity semantic representation. Connecting the LLM only to the terminal deep layer therefore discards shallow- and mid-level cues and prevents the LLM's layer-wise semantics from interacting with the full visual hierarchy.

Two lines of work have started to expose multi-layer visual features to the LLM, but each falls short on the *where to connect* question. **Encoder-side fusion** [4, 20, 23, 36, 39] merges multi-layer features into one representation before the LLM (Figure 2(B)); any dynamism is consumed inside the encoder and cannot adapt to the differentiated demands of individual LLM layers. **Predefined sparse or hierarchical connections** [8, 30, 37, 43] wire selected ViT layers to specific LLM layers (Figure 2(C)), typically using hand-crafted connection patterns or fixed hierarchical mappings. CLI [5] introduces adaptive gating conditioned on the current LLM state, but operates over sampled ViT layers and designated LLM injection points. Neither category explicitly models the *adaptive*, *dynamic* cross-layer interactions between *all* ViT layers and *all* LLM layers that the question of *where to connect* demands.

Predictive coding in visual neuroscience suggests that perception is not a unidirectional feed-forward process but a hierarchical feedback loop that continuously modulates activations across layers [33]. We take this as inspiration to revisit visual token injection: visual evidence should not be delivered in one shot through a fixed pathway, but should keep *participating* in the LLM's representation updates through dynamic cross-layer pathways. Building on this view, we introduce DGP (Dynamic Gated Pathways), which connects *all* ViT layers to *all* LLM layers via learned, dynamically gated routes (Figure 2(D)).

Concretely, the outputs of different ViT layers act as candidate representations at varying granularities. At every LLM layer, DGP produces a routing distribution over all ViT layers from the current hidden state, aggregates the candidates via gated weighted fusion, and injects the result back into the layer's visual hidden states through a residual connection. The hierarchical visual evidence keeps supplying cues for language reasoning, while the LLM's layer-wise semantic state drives the preference over visual hierarchies, yielding a cross-layer, progressively evolving vision–language interaction that integrates low-level details and high-level semantics *on demand* at different decoding depths.

We instantiate DGP on top of LLaVA-1.5 [26] and further validate it on Qwen3-VL-2B [3] and LLaVA-NeXT [27]. Across a wide range of multimodal benchmarks, DGP yields consistent improvements over baselines and outperforms prior multi-layer visual feature utilization methods. The learned routing patterns confirm that different LLM layers indeed prefer different visual granularities, alleviating the information bottleneck of fixed single-layer fusion. Our contributions are:

- We reframe visual information injection in MLLMs around the question of *where to connect*, and show that fixed single-layer inputs and predefined sparse or hierarchical connections cannot match the differentiated demands of hierarchical ViT representations and layer-wise LLM semantics.
- We propose DGP (Dynamic Gated Pathways), an all-to-all cross-layer connection mechanism in which every LLM layer dynamically gates its preference over every ViT layer conditioned on its current semantic state.
- We demonstrate the effectiveness and interpretability of DGP on multiple multimodal benchmarks, where layer-wise routing analyses reveal the adaptive, layer-dependent selection behavior of the proposed dynamic gated pathways.

2 Related Work

2.1 Multimodal Large Language Model

A typical Multimodal Large Language Model (MLLM) couples a vision encoder, a vision–language adapter, and a large language model: the encoder turns the image into visual tokens, the adapter projects them into the language space, and the LLM consumes the joint visual–textual sequence to generate responses. Existing MLLMs mainly fall into two architectural families: projector-based decoder-only designs (e.g., LLaVA [26], MiniGPT-4 [44], Qwen-VL [2], InternVL [10]) and cross-attention-based designs (e.g., Flamingo [1]).

Current MLLMs use pretrained vision backbones such as CLIP-ViT [32] and SigLIP [42], all of which expose representations of varying granularities across their layers. Yet most pipelines project only a single fixed layer into the LLM, so the language model sees only one level of visual abstraction. The visual connection mechanism is not merely a cross-modal interface: it determines whether the rich, hierarchical visual representations can be *accessed and reused* inside the LLM.

2.2 Connection Strategies for Visual Token Insertion

Different ViT layers carry different semantic granularities: shallow layers retain local textures and spatial details, while middle and deep layers progressively encode object structures and high-level semantics [6]. A single deep layer is not always the best choice [6], and exploiting the visual hierarchy depends as much on *how* layers are connected to the LLM as on *which* layers are selected [24]. The crux, in other words, is to establish effective connections between vision-encoder layers and LLM layers—i.e., the *where to connect* problem. Prior efforts can be grouped into two categories.

Unified Encoder-side Connections. Methods in this category fuse multi-layer visual features on the encoder side and pass a single unified representation to the LLM. DenseConnector [39] concatenates features from different ViT layers along the token or channel dimension. MMFuser [4] uses deep-layer features as queries to pull fine-grained information from shallow layers. IGVA [20] and TGIF [23] adjust ViT-layer weights conditioned on the instruction. CCRA [36] combines several attention mechanisms for text-conditioned region–semantic alignment. While these designs improve multi-layer feature utilization, the entire fusion happens *before* the LLM; the LLM still receives one homogenized input and cannot reach back into the visual hierarchy as decoding progresses.

Predefined Sparse or Hierarchical Connections. Methods in this category route hierarchical visual information to different LLM layers. DeepStack [30] distributes visual tokens across decoder depths according to a fixed hierarchical schedule. Sparse Shortcuts [43] wires ViT and LLM layers through predefined connection patterns. EVLM [8] feeds multi-layer ViT features into different gated cross-attention layers. CLI [5] applies adaptive gating at sampled vision layers and designated LLM injection points, rather than treating every ViT–LLM pair as a candidate connection. A more recent line of work [37] selects and fuses hierarchical visual features conditioned on each LLM layer's representation. Although these methods begin to acknowledge the hierarchical correspondence between vision and language, most still rely on hand-crafted connection patterns or fixed layer-to-layer mappings, and few of them simultaneously realize (i) *all-to-all* cross-layer connectivity, (ii) routing weights driven by the *current* LLM-layer state, and (iii) end-to-end training with a stable routing distribution—the design point that DGP targets.

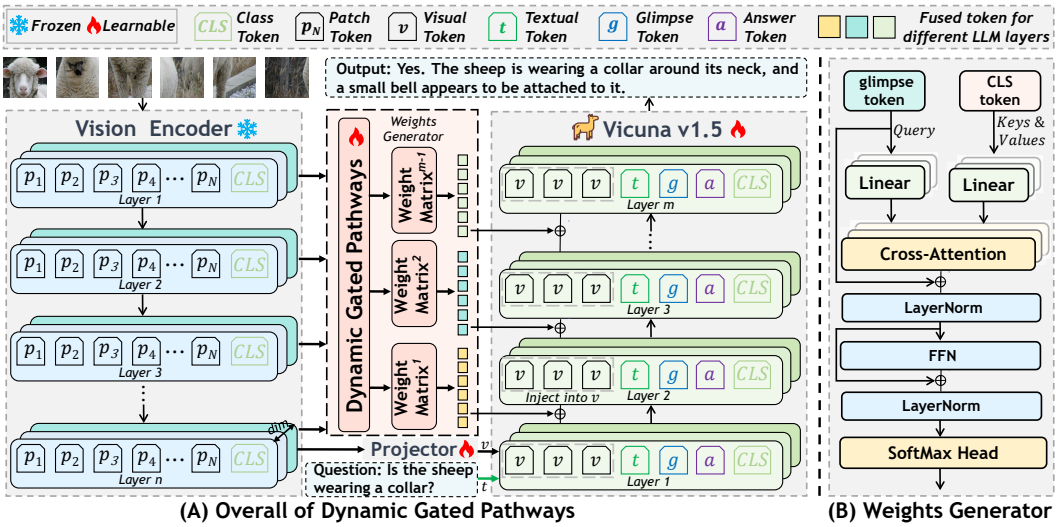


Figure 3: **Overview of** DGP**.** (A) At every LLM layer, the Dynamic Gated Pathways module routes over *all* ViT layers conditioned on the current language-layer state, aggregates the selected visual features, and injects them back into the LLM hidden states through a gated residual connection—establishing all-to-all cross-layer connections between ViT and LLM. (B) Inside the module, ViT layer summaries serve as keys/values and the language-layer state as the query; a shared cross-attention followed by a layer-specific routing MLP produces the per-layer routing weights. Weight Matrix$^{(m)}$ denotes the routing weights of LLM layer m over all ViT layers.

In summary, existing methods either compress the visual hierarchy into a single pre-LLM representation, or wire it into the LLM through predefined sparse or hierarchical connections; neither answers *where to connect* via dynamic cross-layer interactions between *all* ViT layers and *all* LLM layers. We close this gap with DGP, which establishes adaptive, all-to-all connections between ViT and LLM layers through dynamic gated pathways.

3 Methodology

Problem Formulation. In existing MLLMs, a single fixed visual representation is injected once and then propagates implicitly through the LLM [2, 10, 26], leaving no direct cross-layer pathway between the visual hierarchy and the language hierarchy. As a result, different decoding stages cannot reach back to the visual granularity they actually need. DGP (Dynamic Gated Pathways) removes this bottleneck by opening dynamic gated pathways between every ViT layer and every LLM layer, so that each LLM layer can adaptively pick the visual hierarchy that best fits its current semantic state.

Let the vision encoder have L_v layers, with the output of the ℓ-th layer ($\ell = 1, \ldots, L_v$) denoted as $\mathbf{X}_v^{(\ell)}$. The LLM has L layers in total, and the hidden state of the l-th layer ($l = 1, \ldots, L$) is $\mathbf{H}^{(l)} \in \mathbb{R}^{(N_v+N_t)\times d}$, where $\mathbf{H}_v^{(l)} \in \mathbb{R}^{N_v \times d}$ corresponds to the visual token segment. For each LLM layer l, we aim to learn a layer-specific fusion function:

$$\widehat{\mathbf{Z}}^{(l)} = g^{(l)}\left(\mathbf{H}^{(l)}, \{\mathbf{X}_v^{(\ell)}\}_{\ell=1}^{L_v}\right), \tag{1}$$

where $\widehat{\mathbf{Z}}^{(l)} \in \mathbb{R}^{N_v \times d}$ is the fused visual representation tailored to the l-th LLM layer. The objective of $g^{(l)}(\cdot)$ is to allocate fusion weights over visual layers conditioned on the current language-layer state, so that visual evidence at multiple depths is integrated *on demand* during decoding.

3.1 All-to-All Cross-Layer Connections

Our structural departure from typical MLLMs is to wire *every* LLM layer ($l = 1, \ldots, L$) to *every* ViT layer ($\ell = 1, \ldots, L_v$): each decoding depth retains direct access to the full visual hierarchy, rather than being restricted to one fixed depth or a hand-crafted sparse routing pattern. As illustrated in

Figure 2(D) and Figure 3(A), we replace the conventional fixed single-layer visual injection with dynamic gated cross-layer pathways between the ViT and LLM stacks. Given an input image $\mathbf{I}$, the vision encoder $F(\cdot)$ produces a hierarchy of visual representations:

$$\{\mathbf{X}_v^{(\ell)}\}_{\ell=1}^{L_v} = F(\mathbf{I}), \qquad \mathbf{X}_v^{(\ell)} = [\mathbf{c}^{(\ell)}; \mathbf{P}^{(\ell)}], \tag{2}$$

where $\mathbf{c}^{(\ell)}$ is the `[CLS]` representation of the ℓ-th ViT layer and $\mathbf{P}^{(\ell)} \in \mathbb{R}^{N_v \times d_v}$ is its patch representation. Instead of relying on implicit propagation of the initial visual tokens, we connect every LLM layer to every ViT layer explicitly. Concretely, for any LLM layer $l = 1, \ldots, L$, the model derives a layer-specific visual representation $\widehat{\mathbf{Z}}^{(l)}$ from $\{\mathbf{X}_v^{(\ell)}\}_{\ell=1}^{L_v}$ via Eq. (1). This realizes an all-to-all connection set between visual and language hierarchies:

$$\mathcal{E} = \{(\ell, l) \mid \ell = 1, \ldots, L_v,\ l = 1, \ldots, L\}. \tag{3}$$

Each LLM layer can therefore reach low-level details, mid-level structures, and high-level semantics directly, rather than being locked to a single visual layer or a hand-crafted sparse routing pattern.

The layer-specific representation $\widehat{\mathbf{Z}}^{(l)}$ is written back into the visual token segment of the same LLM layer via a gated residual connection (formalized in the Token Fusion paragraph below). Importantly, $g^{(l)}(\cdot)$ is *not* a uniform average over visual layers: it is realized by the Dynamic Gated Pathways module described next, which adaptively controls the strength of each ViT–LLM connection.

3.2 Dynamic Gated Pathways Module

Weight Generation. Inspired by existing glimpse-token designs [41], we prepend a learnable glimpse token $\mathbf{g} \in \mathbb{R}^d$ immediately before the first answer token of every multimodal training sample. Because of the causal autoregressive structure, this token can attend to all preceding visual and textual context, and therefore acts as a compact query signal that summarises the current decoding depth. Let $\mathbf{h}_g^{(l)} \in \mathbb{R}^d$ denote the hidden state at the glimpse position in the l-th LLM layer; we use it as the routing query, $\mathbf{Q}^{(l)} = \mathbf{h}_g^{(l)}$.

For each ViT layer ℓ, we take its global `[CLS]` representation $\mathbf{c}^{(\ell)}$ as the layer summary: since `[CLS]` already aggregates the whole image inside that layer, it provides a compact layer-level descriptor of the corresponding visual granularity [12]. We map all L_v summaries into the LLM hidden space and stack them into shared keys and values, $\mathbf{K} = \mathbf{V} = \mathrm{Stack}\big(\widetilde{\mathbf{c}}^{(1)}, \ldots, \widetilde{\mathbf{c}}^{(L_v)}\big) \in \mathbb{R}^{L_v \times d}$. A multi-head cross-attention [35] module—shared across all LLM layers—then models the interaction between the current language-layer state and the visual-layer summaries, yielding a routing feature:

$$\mathbf{r}^{(l)} = \mathrm{MHA}_{\mathrm{shared}}\Big(\mathbf{Q}^{(l)}, \mathbf{K}, \mathbf{V}\Big) + \mathbf{Q}^{(l)}, \tag{4}$$

where $\mathbf{Q}^{(l)}$ encodes the multimodal context of the current LLM layer and $\mathbf{K}, \mathbf{V}$ summarize all candidate visual layers; together they determine how the l-th LLM layer should allocate connection strength across the ViT hierarchy.

Conditioned on this routing feature, the l-th LLM layer produces a routing distribution over all ViT layers through a layer-specific—i.e., *not* shared—routing MLP:

$$\boldsymbol{\omega}^{(l)} = \mathrm{Softmax}\Big(\mathbf{W}_2^{(l)}\,\mathrm{Dropout}\Big(\mathrm{GELU}\Big(\mathbf{W}_1^{(l)}\,\mathrm{LN}(\mathbf{r}^{(l)}) + \mathbf{b}_1^{(l)}\Big)\Big) + \mathbf{b}_2^{(l)}\Big) \in \mathbb{R}^{L_v}, \tag{5}$$

where $\boldsymbol{\omega}^{(l)} = (\omega_1^{(l)}, \ldots, \omega_{L_v}^{(l)})^\top$ denotes the routing weights of LLM layer l over the L_v ViT layers and satisfies $\sum_{k=1}^{L_v} \omega_k^{(l)} = 1$. $\mathbf{W}_1^{(l)} \in \mathbb{R}^{h \times d}$ and $\mathbf{b}_1^{(l)} \in \mathbb{R}^h$ map the routing feature from the LLM hidden space to the routing hidden space; $\mathbf{W}_2^{(l)} \in \mathbb{R}^{L_v \times h}$ and $\mathbf{b}_2^{(l)} \in \mathbb{R}^{L_v}$ project it to an L_v-dimensional logit that softmax converts into the per-layer routing distribution. In short, $\mathrm{MHA}_{\mathrm{shared}}$ exposes the same query–summary interface to every LLM layer for fair comparison across visual layers, whereas the layer-specific routing MLP $\{\mathbf{W}_1^{(l)}, \mathbf{b}_1^{(l)}, \mathbf{W}_2^{(l)}, \mathbf{b}_2^{(l)}\}$ lets each LLM layer voice its own preference over the visual hierarchy.

Token Fusion. Given the routing weights, we fuse patch features from different visual layers position by position:

$$\widetilde{\mathbf{p}}_i^{(l)} = \sum_{k=1}^{L_v} \omega_k^{(l)} \mathbf{p}_i^{(k)}, \quad i = 1, \ldots, N_v, \tag{6}$$

where $\mathbf{p}_i^{(k)} \in \mathbb{R}^{d_v}$ is the patch at position i from the k-th visual layer and $\widetilde{\mathbf{p}}_i^{(l)} \in \mathbb{R}^{d_v}$ is the aggregated visual feature at that position for the l-th LLM layer. Stacking these vectors row-wise gives $\widetilde{\mathbf{P}}^{(l)} = [\widetilde{\mathbf{p}}_1^{(l)}; \ldots; \widetilde{\mathbf{p}}_{N_v}^{(l)}] \in \mathbb{R}^{N_v \times d_v}$. A layer-specific projector $\rho_l : \mathbb{R}^{d_v} \to \mathbb{R}^d$ is then applied row-wise, $\widehat{\mathbf{Z}}^{(l)} = \rho_l(\widetilde{\mathbf{P}}^{(l)}) \in \mathbb{R}^{N_v \times d}$, and the result is injected back into the visual hidden states through a gated residual connection:

$$\widetilde{\mathbf{H}}_v^{(l)} = \mathbf{H}_v^{(l)} + \gamma_l \widehat{\mathbf{Z}}^{(l)}, \quad \gamma_l = \sigma(a_l), \tag{7}$$

where a_l is a learnable scalar and σ is the sigmoid function; the gate γ_l controls how strongly visual information is injected at the current layer.

Through these dynamic gated pathways, the visual hierarchy interacts with the LLM at every layer: the visual representation evolves with decoding depth rather than being committed up front by a one-shot, fixed injection.

3.3 Training Objective

To prevent the routing distribution from collapsing onto a small subset of visual layers, we augment the standard autoregressive loss with a lightweight *entropy regularizer* that maximizes the entropy of the batch-averaged routing distribution at every LLM layer:

$$\mathcal{L}_{\text{total}} = -\frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} \log p_\theta\Big(x_t \mid x_{<t}, \{\widehat{\mathbf{Z}}^{(l)}\}_{l=1}^{L}\Big) - \lambda \frac{1}{L} \sum_{l=1}^{L} H\Big(\bar{\boldsymbol{\omega}}^{(l)}\Big), \quad H(\boldsymbol{p}) = -\sum_{k=1}^{L_v} p_k \log p_k. \tag{8}$$

Here $\mathcal{M}$ denotes the set of valid supervision positions, $\{\widehat{\mathbf{Z}}^{(l)}\}_{l=1}^{L}$ are the per-layer fused visual representations injected into the LLM, $\bar{\boldsymbol{\omega}}^{(l)} = (\bar{\omega}_1^{(l)}, \ldots, \bar{\omega}_{L_v}^{(l)})^\top$ is the batch-averaged routing distribution at LLM layer l, $H(\cdot)$ is the Shannon entropy, and $\lambda \geq 0$ controls the regularization strength. The second term rewards higher entropy and therefore discourages collapse onto a few visual layers. Following TGIF [23], we set $\lambda = 0.01$ during pre-training and $\lambda = 0$ during instruction fine-tuning so that routing can specialize to task-relevant visual granularities.

4 Experiments

4.1 Experimental Settings

Implementation Details and Training Recipe. We follow the LLaVA-1.5 [26] training recipe, using CLIP-ViT-L/14-336 [32] as the vision encoder and Vicuna as the LLM backbone. Training proceeds in two stages: pre-training (PT) on LLaVA-558K [26] and supervised fine-tuning (SFT) on LLaVA-mixed-665K [26]. Because DGP interacts with the hidden states at every LLM layer, both stages jointly optimize the LLM together with the newly introduced DGP modules. The vision encoder is kept frozen throughout, and each stage is trained for one epoch.

All hyperparameters follow LLaVA-1.5: AdamW with cosine scheduling (warmup ratio 0.03), batch sizes of $256/128$ and learning rates of $1 \times 10^{-3}/2 \times 10^{-5}$ for PT/SFT, respectively. The routing MLP uses a hidden dimension of 1024, and the residual gate is initialized to 0.3. Following TGIF [23], the entropy regularizer uses $\lambda = 0.01$ during PT and is disabled ($\lambda = 0$) during SFT so routing can specialize. More hyperparameter sensitivity results are provided in Appendix C.

Benchmarks. We evaluate on two complementary suites of multimodal benchmarks. *Academic VQA* covers GQA [15], SciQA-IMG [29], VQAv2 [13], and VizWiz [14]. *Instruction-following / LMM benchmarks* cover SEED-Bench (SEED/SEED^I) [18], LLaVA-Wild [25], MM-Vet [40], MMBench [28], and POPE [22], jointly probing multimodal comprehension, fine-grained perception, and hallucination resistance.

Baselines. We compare DGP against two groups of baselines. The first group consists of general-purpose MLLMs: BLIP-2 [19], InstructBLIP [11], Shikra [9], IDEFICS [17], Qwen-VL [2], and LLaVA-1.5 [26]. The second group focuses on *multi-layer visual feature utilization*: MMFuser [4], Dense Connector [39], DeepStack-L [30], Sparse Shortcuts [43], IGVA [20], and TGIF [23]. For a fair comparison, we focus on methods that share the Vicuna backbone and the LLaVA-1.5 configuration.

Table 1: **Comparison with state-of-the-art MLLMs.** We evaluate on Academic-oriented VQA tasks and recent LMM benchmarks. Best results are in **bold**; second-best are underlined. "-" denotes results not reported in the original papers. Rows below DGP report the absolute point gain (↑) or loss (↓) relative to LLaVA-1.5.

Method	LLM	*Academic-oriented VQA*				*LMM benchmarks*						
		GQA	SQA^I	VQA^{v2}	VizWiz	SEED	$SEED^I$	$LLaVA^W$	MM-Vet	MMB	MMB^{CN}	POPE
BLIP-2 [19]	Vicuna-13B	41.0	61.0	65.0	19.6	46.4	49.7	38.1	22.4	–	–	85.3
InstructBLIP [11]	Vicuna-7B	49.2	60.5	–	34.5	53.4	58.8	60.9	26.2	36.0	23.7	–
InstructBLIP [11]	Vicuna-13B	49.5	63.1	–	33.4	78.9	–	58.2	25.6	–	–	78.9
Shikra [9]	Vicuna-13B	–	–	77.4	–	–	–	–	–	58.8	–	–
IDEFICS-9B [17]	LLaMA-7B	38.4	–	50.9	35.5	–	44.5	–	–	48.2	25.2	–
IDEFICS-80B [17]	LLaMA-65B	45.2	–	60.0	36.0	–	53.2	–	–	54.5	38.1	–
Qwen-VL [2]	Qwen-7B	59.3	67.1	78.8	35.2	56.3	62.3	–	–	38.2	7.4	–
Qwen-VL-Chat [2]	Qwen-7B	57.5	68.2	78.2	38.9	58.2	65.4	–	–	60.6	56.7	–
LLaVA-1.5 [26]	Vicuna-7B	62.0	66.8	78.5	50.0	58.6	66.1	63.4	30.5	64.3	58.3	85.9
MMFuser [4]	Vicuna-7B	62.8	68.7	79.1	53.4	60.8	–	65.5	32.6	**67.5**	60.1	86.3
Dense Connector [39]	Vicuna-7B	63.8	69.5	79.5	–	–	–	–	32.7	66.8	–	86.6
DeepStack-L [30]	Vicuna-7B	63.1	–	79.5	–	60.6	–	–	29.9	–	–	86.7
SparseCut [43]	Vicuna-7B	63.6	70.0	**80.0**	54.2	–	67.3	–	–	67.3	60.1	87.7
IGVA [20]	Vicuna-7B	63.1	70.2	–	**54.3**	–	**68.3**	–	33.5	66.9	–	**87.8**
TGIF [23]	Vicuna-7B	62.6	70.1	–	–	–	–	–	–	66.4	–	86.2
DGP	Vicuna-7B	**64.6**	**71.0**	79.7	53.1	**61.5**	67.5	**66.5**	**33.8**	66.7	**60.5**	87.0
		↑2.6	↑4.2	↑1.2	↑3.1	↑2.9	↑1.4	↑3.1	↑3.3	↑2.4	↑2.2	↑1.1
LLaVA-1.5 [26]	Vicuna-13B	63.3	71.6	80.0	53.6	61.6	68.2	70.7	35.4	67.7	63.6	85.9
MMFuser [4]	Vicuna-13B	63.4	71.2	80.1	**57.4**	62.0	–	**71.8**	36.6	69.9	63.8	87.5
DeepStack-L [30]	Vicuna-13B	64.2	–	80.9	–	**63.5**	–	–	35.9	–	–	87.7
SparseCut [43]	Vicuna-13B	64.0	71.8	**81.8**	56.1	–	**69.0**	–	–	69.7	64.1	**87.8**
DGP	Vicuna-13B	**65.5**	**73.0**	80.7	**57.4**	63.0	**69.0**	70.0	**37.5**	**70.1**	**65.4**	87.6
		↑2.2	↑1.4	↑0.7	↑3.8	↑1.4	↑0.8	↓0.7	↑2.1	↑2.4	↑1.8	↑1.7

4.2 Main Results

Comparison with SOTA Methods. Table 1 compares DGP with mainstream MLLMs and multi-layer visual feature utilization methods under Vicuna-7B and 13B settings. Taking Vicuna-7B as an illustrative example, DGP consistently improves over the LLaVA-1.5 baseline—most prominently on VizWiz (+3.1), SQA^I (+4.2), $LLaVA^W$ (+3.1), and MM-Vet (+3.3)—with similar trends on the remaining benchmarks. Against the strongest multi-layer baselines, DGP attains the best score on GQA, SQA^I, SEED, $LLaVA^W$, MM-Vet, and MMB^{CN}, suggesting that compressing the visual hierarchy on the encoder side or using predefined sparse or hierarchical connections is insufficient to capture the full interaction between hierarchical visual evidence and the layer-wise LLM state. Overall, the gains support our central claim: dynamically gating *where to connect* between ViT and LLM layers is a more effective use of the visual hierarchy than any fixed wiring.

Generalization Across MLLM Frameworks. To verify that DGP captures a general architectural principle, we evaluate it on two additional MLLM configurations that differ substantially in vision encoder, LLM backbone, and resolution strategy. For fair comparison, all models are trained on the same 558K (PT) + 665K (SFT) data splits following DenseConnector [39], with one epoch per stage. As shown in Table 2, DGP delivers consistent improvements on both Qwen3-VL-2B (SigLIP2 + Qwen3-1.7B) and LLaVA-NeXT (AnyRes tiling), confirming that the *where to connect* principle is architecture-agnostic.

Table 2: **Generalization across MLLM frameworks.** We evaluate DGP on different vision encoders (SigLIP2 vs. CLIP), LLM backbones (Qwen3 vs. Vicuna), and high-resolution tiling.

Method	ViT	LLM	GQA	SQA^I	MMB	MM-Vet
Qwen3-VL-2B [3]	SigLIP2-L	Qwen3-1.7B	63.9	85.3	76.4	49.8
+ DGP	SigLIP2-L	Qwen3-1.7B	**65.4**	**86.9**	**77.4**	**50.9**
LLaVA-NeXT [27]	CLIP-L	Vicuna-7B	64.0	69.5	66.5	33.1
+ DGP	CLIP-L	Vicuna-7B	**64.8**	**72.3**	**68.7**	**34.6**

Case Study. To make the effect of DGP more concrete, Figure 4 contrasts DGP with LLaVA-1.5 on math reasoning, OCR understanding, and fine-grained visual reasoning. LLaVA-1.5 frequently fails to read critical visual evidence or forms wrong semantic associations—for example, misreading

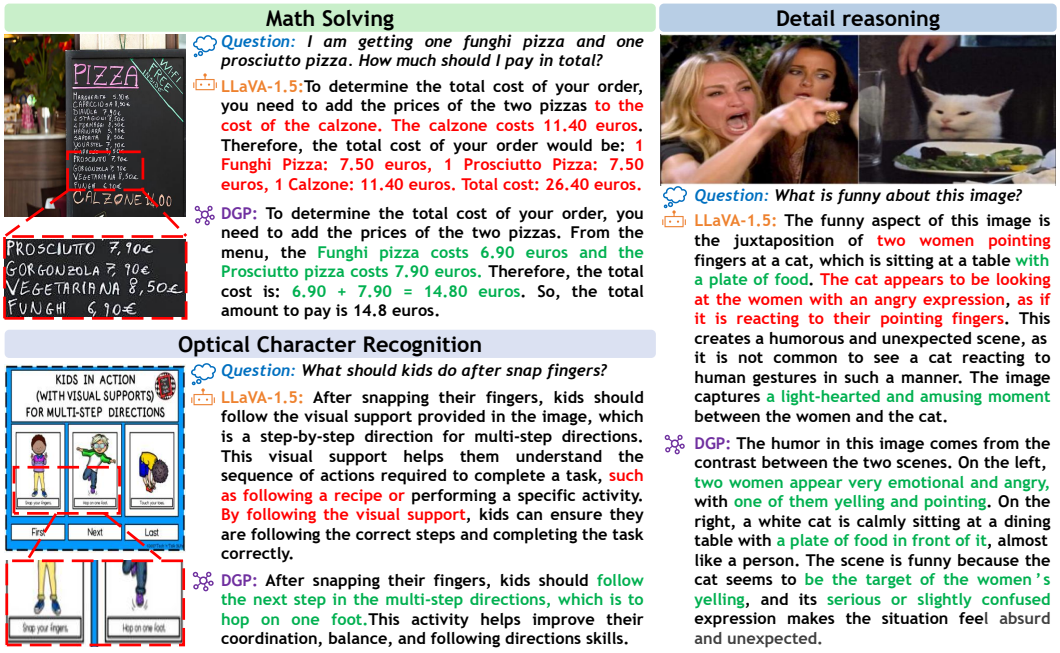


Figure 4: **Qualitative comparison between** DGP **and LLaVA-1.5.** Representative examples from math solving, OCR understanding, and detailed visual reasoning. LLaVA-1.5 often misses critical visual evidence or hallucinates content; DGP produces more accurate, evidence-grounded responses by adaptively accessing multi-granularity visual features.

menu prices or hallucinating non-existent items such as a "calzone"—and this propagates into clearly hallucinated reasoning. DGP, in contrast, uses its dynamic gated pathways to let different LLM layers access multi-granularity visual cues on demand, accurately recovering textual and numerical details, facial expressions, and other fine-grained evidence, and grounds the final answer in those cues.

Efficiency Analysis. Table 3 reports the computational overhead of DGP. The DGP module introduces only 5.29% additional parameters for the 7B model and 4.27% for the 13B model. The FLOPs overhead is modest for 7B (+4.07%) and negligible for 13B (−0.07%), while peak memory is nearly unchanged for 7B (−0.13%) and increases by 4.13% for 13B. Taken together with the performance gains, these modest overheads indicate a favorable trade-off between multimodal performance and computational cost.

Table 3: Efficiency analysis of DGP at batch size 1. FLOPs and peak memory are measured during end-to-end autoregressive generation.

Method	Params (M)	FLOPs (G)	Mem (GB)
LLaVA-1.5-7B	6,759.4	9,168.3	15.51
+ DGP	7,117.3 +5.29%	9,541.2 +4.07%	15.49 −0.13%
LLaVA-1.5-13B	13,047.3	17,731.2	27.34
+ DGP	13,604.9 +4.27%	17,718.8 −0.07%	28.47 +4.13%

4.3 Ablation Studies and Analysis

Ablation of DGP **Components.** Table 4 ablates the two key ingredients of DGP—gated routing and entropy regularization—and keeps a uniform average over the 24 ViT layers as a control, so as to disentangle gains coming from "using more visual layers" from those coming from *where to connect*. Adding ***gated routing alone*** already yields consistent improvements over the LLaVA-1.5 baseline on most benchmarks. By contrast, ***uniform average fusion*** without gated routing hurts performance on most benchmarks relative to the baseline. Equal-weight mixing across the visual hierarchy appears to blend incompatible granularities and weaken fine-grained evidence, whereas gated routing keeps connectivity selective and conditioned on the language-layer state. Stacking the ***entropy regularizer*** on top of gated routing brings a further gain, indicating that a well-spread routing distribution prevents collapse onto a few visual layers and stabilizes training. Together, these results show that the improvement is driven by the *where to connect* mechanism—routing that adapts to the language-layer state—rather than by the mere availability of more visual features.

Table 4: **Ablation of DGP components.** We evaluate the contribution of average fusion, gated routing, and entropy regularization.

Avg. Fusion	Gated Routing	Entropy Reg.	*Academic-oriented VQA* GQA	SQA^I	VQA^{v2}	VizWiz	*LMM benchmarks* SEED	$SEED^I$	$LLaVA^W$	MM-Vet	MMB	MMB^{CN}	POPE
✗	✗	✗	62.0	66.8	78.5	50.0	58.6	66.1	63.4	30.5	64.3	58.3	85.9
✓	✗	✗	60.2	64.3	72.0	50.2	57.9	63.3	56.3	25.3	57.6	44.9	77.2
✗	✓	✗	64.1	70.5	78.9	52.0	60.9	66.7	63.6	32.4	65.6	59.1	86.4
✗	✓	✓	**64.6**	**71.0**	**79.7**	**53.1**	**61.5**	**67.5**	**66.5**	**33.8**	**66.7**	**60.5**	**87.0**

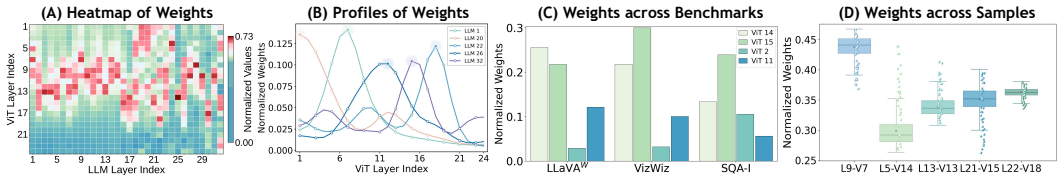


Figure 5: **Gated Pathway Analysis.** (A) Heatmap of routing weights over LLM (x) and ViT (y) layers. (B) Routing curves of representative LLM layers with multimodal patterns. (C) Routing weights of LLM layer 21 across benchmarks, showing task-specific preferences. (D) Per-sample routing distributions within the same benchmark, showing sample-level diversity.

Layer-wise Gated Pathway Analysis. To inspect what DGP actually learns, we visualize the routing weights of every LLM layer over the ViT layers. Figure 5(A) shows a clearly non-uniform, layer-dependent distribution: DGP neither fixates on the last ViT layer nor degenerates into a uniform average. Middle ViT layers are activated most frequently, suggesting that they offer a useful trade-off between local structure and high-level semantics. Shallow and deep features are also selected by certain LLM layers, confirming that the model dynamically reaches into different visual granularities according to where it is in decoding rather than relying on a single visual layer.

Figure 5(B) zooms in on representative LLM layers. Their routing curves peak at different positions: LLM layer 1 favors shallow-to-middle ViT layers, LLM layer 20 leans toward shallow features, whereas LLM layers 22 and 32 peak at mid-deep and deep ViT layers, respectively. The curves are typically multimodal rather than unimodal or flat, showing that DGP selectively combines several visual layers per LLM layer and that different LLM layers prefer different visual granularities depending on their language state. These visualizations confirm that DGP discovers non-uniform, dynamic ViT–LLM connection patterns, exploiting the complementarity of the visual hierarchy.

Task-aware Gated Pathway Analysis. We next ask whether the routing also adapts to the task itself. Figure 5(C) plots the routing weights of LLM layer 21 across benchmarks: each benchmark induces a distinct visual-layer profile. $LLaVA^W$ and VizWiz allocate more mass to mid-level features, in line with the need to balance object-level semantics and local visual cues for open-ended QA and real-world understanding, whereas SQA^I additionally puts noticeable weight on shallow ViT layer 2, reflecting the fine-grained details required in scientific reasoning. Figure 5(D) zooms one level further down: even within a single benchmark, individual samples exhibit substantial routing diversity, indicating that DGP's adaptivity goes beyond benchmark-level differences and operates at the sample level.

Figure 6 fixes the image and varies the question. For questions about attribute recognition or small-region details, DGP activates shallow or shallow-to-middle ViT features—layers that preserve edges, textures and local spatial structure. For questions that rely more on global semantics or abstract reasoning—painting identification, historical significance, action understanding—the routing weight peaks instead at mid-deep or deep ViT layers. For open-ended questions such as "describe the image in detail," which require both local details and holistic semantics, the routing distribution naturally widens into a broader multimodal shape that simultaneously taps multiple visual hierarchies. These patterns indicate that DGP's routing is dynamically conditioned on the layer-wise linguistic context and semantic state of the LLM, rather than being determined solely by fixed visual-layer priors.

Feature Visualization. Figure 7 compares fixed single-layer CLIP features (L23), a uniform average over all 24 ViT layers, and the features produced by DGP at different LLM layers. CLIP (L23) yields coarse spatial responses with noisy activations in the background; naive average fusion brings

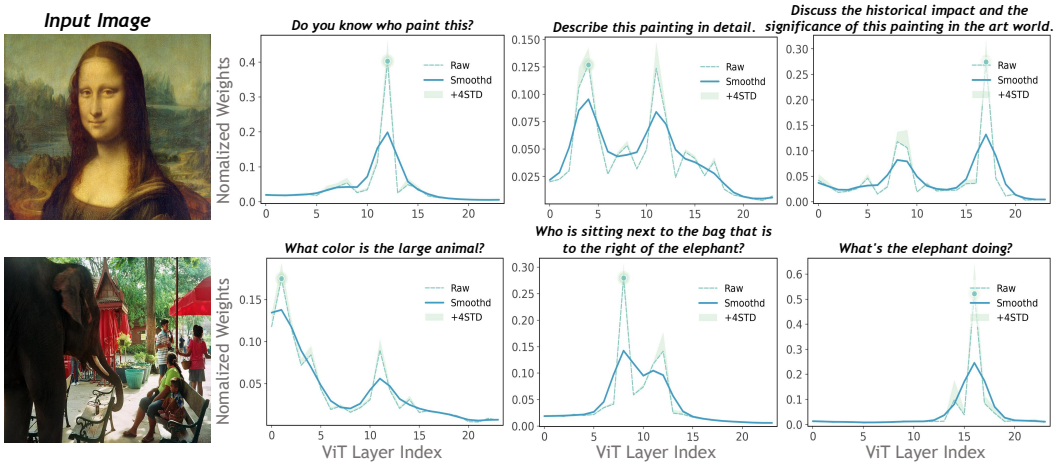


Figure 6: **Task-aware routing for the same image.** As the question changes, routing shifts: attribute recognition activates shallower layers, while global-semantic questions activate deeper ones.

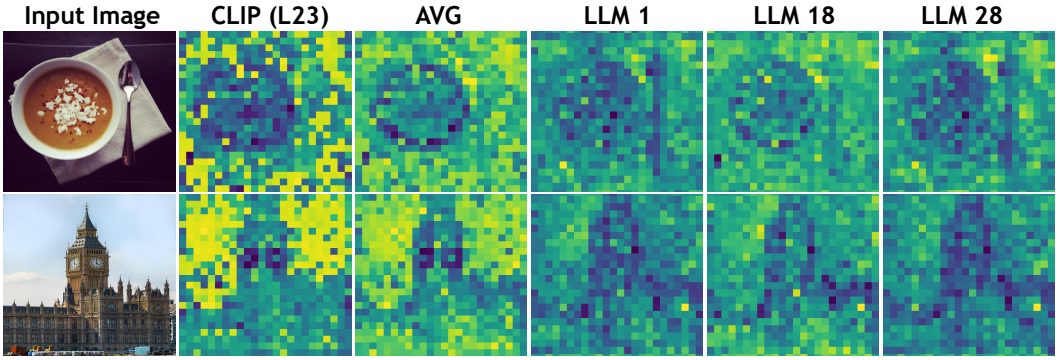


Figure 7: **Fused visual feature visualization.** We compare CLIP layer 23, the uniform average over all 24 ViT layers, and DGP's gated fusion at different LLM layers.

in multi-layer information but, because features at different semantic granularities are superposed without selection, still suffers from diffuse responses and amplified local noise. DGP's LLM-layer-specific features instead show clearly more stable spatial structure, sharper subject contours, and lower background noise, preserving local details such as spoon edges and building boundaries. Crucially, the fused features differ across LLM layers 1, 18, and 28, confirming that DGP generates differentiated visual representations via dynamic routing rather than collapsing into a fixed averaging pattern. Overall, DGP strengthens global semantics while keeping shallow and middle-layer cues intact, mitigating both the information bottleneck of fixed single-layer features and the noise accumulation of naive averaging.

5 Conclusion

In this work, we reframe visual information injection in MLLMs around the question of *where to connect*, and argue that fixed single-layer inputs or predefined sparse or hierarchical connections cannot match the differentiated demands of hierarchical ViT representations and layer-wise LLM semantics. We answer this question with DGP (Dynamic Gated Pathways), an all-to-all cross-layer connection mechanism in which each LLM layer adaptively gates its preference over every ViT layer based on its current semantic state, integrating multi-granularity visual information on demand at every decoding depth. DGP delivers consistent gains across a broad set of multimodal benchmarks, while routing analyses and feature visualizations show that it learns interpretable, layer-dependent connection patterns that explain where the gains come from.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Nos. 62472359 and 62372379).

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Yue Cao, Yong Huang, Wei Zhu, Yangzhou Liu, Zhe Chen, Guangchen Shi, Yong Fa, Yujie Yang, Song Mei, and Tong Lu. Mmfuser: Multimodal multi-layer feature fuser for fine-grained vision-language understanding. In *International Conference on Pattern Recognition*, pages 406–421. Springer, 2026.

[5] Cheng Chen, Yuyu Guo, Pengpeng Zeng, Jingkuan Song, Peng Di, Hang Yu, and Lianli Gao. From one-to-one to many-to-many: Dynamic cross-layer injection for deep vision-language fusion. In *European Conference on Computer Vision*, pages 616–634. Springer, 2026.

[6] Haoran Chen, Junyan Lin, Xinghao Chen, Yue Fan, Jianfeng Dong, Xin Jin, Hui Su, Jinlan Fu, and Xiaoyu Shen. Multimodal language models see better when they look shallower. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6688–6706, 2025.

[7] Jianmin Chen, Jiaqi Tang, Wei Wei, Xiaogang Xu, Jiafei Wu, Zhe Liu, Qianzhou Wang, Yingying Yan, Botong Geng, Yuyang Xia, et al. Remember-r1: Mitigating long-context visual forgetting through reinforcement learning. *arXiv preprint arXiv:2608.01314*, 2026.

[8] Kaibing Chen, Dong Shen, Hanwen Zhong, Huasong Zhong, Kui Xia, Di Xu, Wei Yuan, Yifei Hu, Bin Wen, Tianke Zhang, et al. Evlm: An efficient vision-language model for visual understanding. *arXiv preprint arXiv:2407.14177*, 2024.

[9] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023.

[10] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024.

[11] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267, 2023.

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

[13] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[14] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3608–3617. IEEE, 2018.

[15] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702. IEEE, 2019.

[16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[17] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander Rush, Douwe Kiela, et al. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *Advances in Neural Information Processing Systems*, 36: 71683–71702, 2023.

[18] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

[19] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[20] Xu Li, Yi Zheng, Haotian Chen, Xiaolei Chen, Yuxuan Liang, Chenghang Lai, Bin Li, and Xiangyang Xue. Instruction-guided fusion of multi-layer visual features in large vision-language models. *Pattern Recognition*, 170:111932, 2026.

[21] Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia. Mini-gemini: Mining the potential of multi-modality vision language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[22] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 292–305, 2023.

[23] Chenchen Lin, Sanbao Su, Rachel Luo, Yuxiao Chen, Yan Wang, Marco Pavone, and Fei Miao. Text-guided layer fusion mitigates hallucination in multimodal llms. *arXiv preprint arXiv:2601.03100*, 2026.

[24] Junyan Lin, Haoran Chen, Yue Fan, Yingqi Fan, Xin Jin, Hui Su, Jinlan Fu, and Xiaoyu Shen. Multi-layer visual feature fusion in multimodal llms: Methods, analysis, and best practices. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4156–4166. IEEE, 2025.

[25] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[26] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26296. IEEE, 2024.

[27] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge. `https://llava-vl.github.io/blog/2024-01-30-llava-next/`, 2024.

[28] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024.

[29] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521, 2022.

[30] Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms. *Advances in Neural Information Processing Systems*, 37:23464–23487, 2024.

[31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[33] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[34] Jiaqi Tang, Jianmin Chen, Wei Wei, Xiaogang Xu, Runtao Liu, Xiangyu Wu, Qipeng Xie, Jiafei Wu, Lei Zhang, and Qifeng Chen. Robust-r1: Degradation-aware reasoning for robust visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 9421–9429, 2026.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] Yifan Wang, Hongfeng Ai, Quangao Liu, Maowei Jiang, Ruiyuan Kang, Ruiqi Li, Ruikai Liu, Qi Wang, Tianyu Zhou, Wenyan Zhang, et al. Ccra: Optimizing vision-language consistency via cross-layer regional attention alignment. *Information Fusion*, page 104833, 2026.

[37] Xinyu Wei, Guoli Yang, Jialu Zhou, Mingyue Yang, Leqian Li, Kedi Zhang, and Chunping Qiu. Dynamic embedding of hierarchical visual features for efficient vision-language fine-tuning. *arXiv preprint arXiv:2508.17638*, 2025.

[38] Yingying Yan, Jiaqi Tang, Wei Wei, Qianzhou Wang, Jinjian Wu, Botong Geng, Jianmin Chen, Yuyang Xia, and Lei Zhang. Heattok: Enhancing remote sensing image understanding via thermodiffusion-based tokenization. *arXiv preprint arXiv:2608.22485*, 2026.

[39] Huanjin Yao, Wenhao Wu, Taojiannan Yang, YuXin Song, Mengxi Zhang, Haocheng Feng, Yifan Sun, Zhiheng Li, Wanli Ouyang, and Jingdong Wang. Dense connector for mllms. *Advances in Neural Information Processing Systems*, 37:33108–33140, 2024.

[40] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

[41] Quan-Sheng Zeng, Yunheng Li, Qilong Wang, Peng-Tao Jiang, Zuxuan Wu, Ming-Ming Cheng, and Qibin Hou. A glimpse to compress: Dynamic visual token pruning for large vision-language models. *IEEE Transactions on Circuits and Systems for Video Technology*, 2026.

[42] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11941–11952. IEEE, 2023.

[43] Jingrui Zhang, Feng Liang, Yong Zhang, Wei Wang, Runhao Zeng, and Xiping Hu. Sparse shortcuts: Facilitating efficient fusion in multimodal large language models. *arXiv preprint arXiv:2602.00505*, 2026.

[44] Deyao Zhu, Xiaoqian Shen, Xiang Li, Mohamed Elhoseiny, et al. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *International Conference on Learning Representations*, volume 2024, pages 18378–18394, 2024.

A Methodological Details

Algorithm 1 summarizes the overall procedure of DGP. Table 5 details the token flow and tensor shapes of DGP under the Vicuna-7B setting.

A.1 Algorithm

Algorithm 1 Dynamic Gated Pathways (DGP)

Require: Input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, text tokens $\mathbf{T} \in \mathbb{R}^{N_t \times d}$
Require: Vision encoder $F(\cdot)$ with L_v layers, LLM with L layers
Require: Shared cross-attention $\mathrm{MHA}_{\mathrm{shared}}$, layer-specific routing MLPs $\{\mathrm{MLP}^{(l)}\}_{l=1}^{L}$
Require: Layer-specific projectors $\{\rho_l\}_{l=1}^{L}$, gate scalars $\{a_l\}_{l=1}^{L}$
Require: Learnable glimpse token $\mathbf{g} \in \mathbb{R}^d$, layer ID embeddings $\mathbf{E}_{\mathrm{id}} \in \mathbb{R}^{L_v \times d}$

// Step 1: Vision Encoding and Feature Caching

$\{\mathbf{X}_v^{(\ell)}\}_{\ell=1}^{L_v} \leftarrow F(\mathbf{I})$ ▷ Extract hierarchical visual features
for $\ell = 1, \ldots, L_v$ **do**
  $\mathbf{P}^{(\ell)} \leftarrow \mathbf{X}_v^{(\ell)}[:, 1:]$ ▷ Patch features: $[B, N_v, d_v]$
  $\mathbf{c}^{(\ell)} \leftarrow \mathbf{X}_v^{(\ell)}[:, 0]$ ▷ CLS token: $[B, d_v]$
  $\widetilde{\mathbf{c}}^{(\ell)} \leftarrow \mathrm{Linear}(\mathrm{LN}(\mathbf{c}^{(\ell)}))$ ▷ Project CLS to LLM space: $[B, d]$
end for
$\mathbf{K} = \mathbf{V} \leftarrow \mathrm{Stack}(\widetilde{\mathbf{c}}^{(1)}, \ldots, \widetilde{\mathbf{c}}^{(L_v)}) + \mathbf{E}_{\mathrm{id}}$ ▷ CLS summaries with layer IDs: $[B, L_v, d]$

// Step 2: Initial Visual Token Injection

$\mathbf{V}_0 \leftarrow \mathrm{mm_projector}(\mathbf{P}^{(L_v-1)})$ ▷ Penultimate-layer patches to LLM space: $[B, N_v, d]$
$\mathbf{H}^{(0)} \leftarrow \mathrm{Concat}(\mathbf{V}_0, \mathbf{T}, \mathbf{g})$ ▷ Multimodal input sequence

// Step 3: Layer-wise Dynamic Gated Pathway Injection

for $l = 1, \ldots, L$ **do**
  $\mathbf{h}_g^{(l)} \leftarrow \mathbf{H}^{(l-1)}[\mathrm{glimpse_pos}]$ ▷ Glimpse hidden state: $[B, 1, d]$
  $\mathbf{Q}^{(l)} \leftarrow \mathbf{h}_g^{(l)}$ ▷ Routing query (as in Methodology)
  $\mathbf{r}^{(l)} \leftarrow \mathrm{MHA}_{\mathrm{shared}}(\mathbf{Q}^{(l)}, \mathbf{K}, \mathbf{V}) + \mathbf{Q}^{(l)}$ ▷ Cross-attention with residual
  $\boldsymbol{\omega}^{(l)} \leftarrow \mathrm{Softmax}(\mathrm{MLP}^{(l)}(\mathrm{LN}(\mathbf{r}^{(l)})))$ ▷ Routing weights: $[B, L_v]$
  $\widetilde{\mathbf{P}}^{(l)} \leftarrow \sum_{k=1}^{L_v} \omega_k^{(l)} \cdot \mathbf{P}^{(k)}$ ▷ Weighted patch fusion: $[B, N_v, d_v]$
  $\widehat{\mathbf{Z}}^{(l)} \leftarrow \rho_l(\widetilde{\mathbf{P}}^{(l)})$ ▷ Project to LLM space: $[B, N_v, d]$
  $\gamma_l \leftarrow \sigma(a_l)$ ▷ Gating coefficient
  $\widetilde{\mathbf{H}}_v^{(l-1)} \leftarrow \mathbf{H}_v^{(l-1)} + \gamma_l \cdot \widehat{\mathbf{Z}}^{(l)}$ ▷ Gated residual injection (Methodology Eq. (7))
  $\mathbf{H}^{(l)} \leftarrow \mathrm{LLMLayer}_l(\mathbf{H}^{(l-1)}$ with $\mathbf{H}_v^{(l-1)} \leftarrow \widetilde{\mathbf{H}}_v^{(l-1)})$ ▷ Forward with injected visual segment
end for
Return Final hidden states $\mathbf{H}^{(L)}$ for autoregressive decoding

A.2 Token Flow in DGP (Vicuna-7B Example)

Taking Vicuna-7B (32 LLM layers, hidden dimension 4096) and CLIP-ViT-L/14-336 (24 ViT layers, hidden dimension 1024, $24 \times 24 = 576$ patches) as an example, Table 5 details the token flow and dimension changes at each stage of DGP.

As shown in the table, the glimpse token is inserted immediately before the answer tokens (sequence format: `v...v t...t g a...a`), allowing it to attend to all visual and textual context under causal attention. At each LLM layer, the hidden state at the glimpse position queries the CLS summaries of all 24 ViT layers to produce layer-specific routing weights, which are then used to perform a weighted fusion over the original 1024-dimensional patch features. The fused features are projected to 4096 dimensions by a layer-specific projector and injected into the visual token hidden states via gated residual connections. This mechanism enables each LLM layer to adaptively select the visual granularity that best matches its current semantic state.

Table 5: **Token flow and tensor dimensions in DGP.** We illustrate each processing stage using Vicuna-7B (32 layers, $d = 4096$) and CLIP-ViT-L/14-336 (24 layers, $d_v = 1024$, $N_v = 576$ patches). B denotes batch size, N_t denotes the number of text tokens.

Stage	Operation	Tensor Shape	Description
Vision Encoding (ViT)			
Raw patches	ViT layer ℓ output	$[B, 577, 1024]$	1 CLS + 576 patches per layer
Patch features	Remove CLS	$[B, 576, 1024] \times 24$	Cache all 24 layers for routing
CLS tokens	Extract CLS	$[B, 1, 1024] \times 24$	Layer summaries for cross-attention
CLS Projection for Routing			
CLS projection	$\mathrm{LN}(1024) \to \mathrm{Linear}(1024 \to 4096)$	$[B, 1, 4096] \times 24$	Project each CLS to LLM space
Layer-ID keys/values	$\mathrm{Stack}(\tilde{\mathbf{c}}^{(1)}+\mathbf{e}_1, \ldots, \tilde{\mathbf{c}}^{(L_v)}+\mathbf{e}_{L_v})$	$[B, L_v, d]$	$\mathbf{K}, \mathbf{V}$ for $\mathrm{MHA}_{\mathrm{shared}}$ (L_v=24, d=4096)
Initial LLM Input Construction			
Visual tokens	$\mathrm{mm_projector}(\mathbf{P}^{(L_v-1)})$	$[B, 576, 4096]$	Penultimate ViT patches $\to$ LLM space (L_v=24)
Text tokens	Embedding lookup	$[B, N_t, 4096]$	Standard text embedding
Glimpse token	Learnable + Linear $\to$ RMSNorm	$[B, 1, 4096]$	Inserted before first answer token
Full sequence	$[\mathbf{V}; \mathbf{T}; \mathbf{g}; \mathbf{A}]$	$[B, 576{+}N_t{+}1{+}N_a, 4096]$	`[v v v ... t t t ... g a a a ...]`
Per-Layer Dynamic Gated Pathways (at LLM layer l)			
Glimpse query	$\mathbf{Q}^{(l)} = \mathbf{h}_g^{(l)}$ (glimpse position)	$[B, 1, 4096]$	Routing query from current language-layer state
Cross-attention	$\mathrm{MHA}_{\mathrm{shared}}(\mathbf{Q}^{(l)}, \mathbf{K}, \mathbf{V})$	$[B, 1, 4096]$	Shared cross-attention over ViT CLS summaries
Routing feature	$\mathrm{MHA}_{\mathrm{shared}}$ output + residual	$[B, 1, 4096]$	Input to layer-specific routing MLP
Routing weights	$\mathrm{MLP}^{(l)} \to \mathrm{Softmax}$	$[B, 24]$	Per-layer distribution over ViT layers
Per-Layer Patch Fusion and Injection (at LLM layer l)			
Stacked patches	Stack 24-layer patches	$[B, 24, 576, 1024]$	All ViT layers' patch features
Weighted fusion	$\sum_\ell \omega_\ell^{(l)} \cdot \mathbf{P}^{(\ell)}$	$[B, 576, 1024]$	Softmax-weighted combination
Fusion projection	$\rho_l : \mathrm{LN}(1024) \to \mathrm{Linear}(1024 \to 4096)$	$[B, 576, 4096]$	Layer-specific projector
Gated injection	$\mathbf{H}_v^{(l)} + \gamma_l \cdot \widehat{\mathbf{Z}}^{(l)}$	$[B, 576, 4096]$	Update visual token hidden states

B Training Configuration and Benchmark Descriptions

B.1 Training Configuration

Table 6 lists the detailed training hyperparameters for the pre-training (PT) and supervised fine-tuning (SFT) stages. We follow the LLaVA-1.5 training recipe, using CLIP-ViT-L/14-336 as the vision encoder and Vicuna as the LLM backbone. The vision encoder is kept frozen throughout both stages, each trained for one epoch. The entropy regularizer is enabled during PT to stabilize routing distributions, and disabled during SFT to allow task-specific specialization. All training and efficiency measurements are conducted on NVIDIA L40-48GB GPUs.

Backbone Selection Rationale. Our primary experiments are conducted under the LLaVA-1.5 framework, which remains a widely adopted open-source baseline in the research community. Specifically, we select Vicuna-v1.5 (7B/13B) as the LLM backbone and adopt the standard LLaVA-1.5 training data (LLaVA-558K for PT and LLaVA-mixed-665K for SFT). This design choice serves two purposes: (i) *reproducibility*, as all components are publicly available and well-documented; and (ii) *fair comparison* with modern multi-layer visual fusion methods (e.g., MMFuser [4], IGVA [20], TGIF [23]) that likewise build upon LLaVA-1.5. By holding the backbone architecture, training data, and recipe constant, any observed performance improvement can be cleanly attributed to the proposed DGP module itself, rather than to differences in the underlying LLM, vision encoder, or training corpus.

B.2 Benchmark Descriptions

To validate the effectiveness of DGP under varying visual granularity requirements, we adopt two complementary suites of multimodal benchmarks. Different tasks exhibit significantly different demands on visual granularity: fine-grained OCR and attribute recognition rely more on shallow local features, while scene understanding and abstract reasoning require deep global semantics. This diversity in granularity requirements directly motivates the design of DGP's dynamic gated pathways—enabling different LLM layers to adaptively select the visual granularity that best matches their current needs. Table 7 summarizes the benchmark taxonomy and evaluated capabilities.

Table 6: **Training hyperparameters.** Configuration for pre-training (PT) on LLaVA-558K and supervised fine-tuning (SFT) on LLaVA-mixed-665K.

Pre-training (PT)		**Supervised Fine-tuning (SFT)**	
Global batch size	256	Global batch size	128
Learning rate	1×10^{-3}	Learning rate	2×10^{-5}
Vision encoder LR	frozen	Vision encoder LR	frozen
LR schedule	cosine decay	LR schedule	cosine decay
LR warmup ratio	0.03	LR warmup ratio	0.03
Epochs	1	Epochs	1
Optimizer	AdamW	Optimizer	AdamW
Routing MLP hidden dim	1024	Routing MLP hidden dim	1024
Residual gate init (γ)	0.3	Residual gate init (γ)	0.3
Entropy regularizer λ	0.01	Entropy regularizer λ	0

Table 7: **Benchmark descriptions.** We evaluate on two complementary suites: academic task-oriented benchmarks and instruction-following / LMM benchmarks.

Category	**Benchmark**	**Task Type**	**Evaluated Capabilities**
Academic VQA	VQA^{v2} [13]	General VQA	Basic image understanding, object attribute recognition, and commonsense visual reasoning
	GQA [15]	Compositional VQA	Compositional visual reasoning with structured scene-graph annotations (relations, attributes, and logical consistency)
	VizWiz [14]	Real-world VQA	Real user questions from visually impaired users with often noisy images, testing robust visual understanding
	SQA^{I} [29]	Science QA (image)	Scientific knowledge, image-grounded reasoning, and multi-step inference
LMM Bench	POPE [22]	Hallucination probing	Contrastive image–question pairs to test visual grounding and robustness against object hallucination
	MMBench (MMB/MMB^{CN}) [28]	Multi-choice	Classification-style questions covering perception, reasoning, and instruction following (English/Chinese subsets)
	SEED-Bench (SEED) [18]	Multimodal understanding	Vision-centric instruction-following prompts requiring image understanding, captioning, and grounding
	SEED^{I} [18]	Multi-choice	Multiple-choice questions on images, testing vision-centric understanding and instruction compliance
	LLaVA-Wild (LLaVA^{W}) [25]	Open-ended QA	Open-ended multimodal dialogue, complex instruction following, and response quality
	MM-Vet [40]	Integrated capability	Normalized score over six core vision-language skills, probing alignment and reasoning consistency

The diversity of these benchmarks ensures comprehensive coverage from local details to global semantics, and from perception to reasoning, thereby thoroughly validating the adaptive capability of DGP's dynamic gated pathways under varying visual granularity requirements.

C Hyperparameter Sensitivity Analysis

This section analyzes the design choices of key hyperparameters in DGP, including the entropy regularization coefficient λ, the routing MLP hidden dimension h, and the gating initialization value γ_{init}.

Entropy Regularization Coefficient λ. The entropy regularizer discourages routing collapse onto a small subset of visual layers. We adopt the same stage-wise schedule as TGIF [23]: $\lambda = 0.01$ during pre-training to encourage exploration across layers, and no entropy regularization during supervised fine-tuning (equivalently $\lambda = 0$) so routing can specialize to downstream tasks.

Routing MLP Hidden Dimension h. The routing MLP maps the 4096-dimensional routing feature to a 24-dimensional layer weight distribution. To balance expressiveness and computational efficiency, we set the intermediate hidden dimension to $h = 1024$, aligning with the ViT hidden dimension. This design ensures that the routing network has sufficient nonlinear modeling capacity while avoiding the additional computational overhead of using the full LLM dimension (4096).

Gating Initialization γ_{init}. The gating coefficient $\gamma_l = \sigma(a_l)$ controls the injection strength of visual information at the l-th layer, where a_l is a learnable scalar. We use a default gate initialization of $\gamma_{\text{init}} = 0.3$ and set the gate learning rate to 5×10^{-3}. An initialization value that is too small leads to insufficient visual injection, causing the model to degenerate toward pure language model behavior; an initialization value that is too large may introduce excessive noise in early training, affecting convergence stability. Table 8 shows the impact of different initialization values on performance.

Table 8: **Effect of gating initialization γ_{init}.** We vary the initial value of the residual gate and report performance across the full benchmark suite.

γ_{init}	*Academic-oriented VQA*				*LMM benchmarks*						
	GQA	**SQA^I**	**VQA^{v2}**	**VizWiz**	**SEED**	**SEED^I**	**LLaVA^W**	**MM-Vet**	**MMB**	**MMB^{CN}**	**POPE**
0.1	64.4	70.6	79.3	52.7	61.4	67.3	66.2	33.5	66.2	60.2	**87.2**
0.3 (default)	**64.6**	**71.0**	**79.7**	**53.1**	**61.5**	**67.5**	**66.5**	33.8	**66.7**	**60.5**	87.0
0.5	64.3	70.8	79.1	53.0	61.4	67.1	66.3	**34.0**	66.5	60.4	86.9

The results indicate that $\gamma_{\text{init}} = 0.3$ achieves the best performance on most benchmarks, while $\gamma_{\text{init}} = 0.1$ peaks slightly on POPE and $\gamma_{\text{init}} = 0.5$ peaks slightly on MM-Vet. Too-small gates risk insufficient visual injection, whereas overly large gates may inject noisy visual signals early and mildly hurt broader reasoning benchmarks.

D Efficiency Analysis

A natural concern is whether the dynamic gated pathways introduce prohibitive overhead. Table 9 breaks down the parameters of the DGP module for the 7B and 13B models.

Table 9: Parameter breakdown of the DGP module.

Component	**7B (M)**	**% of DGP**	**13B (M)**	**% of DGP**
Routers (per-layer MLP)	135.3	37.80%	211.2	37.88%
Shared Cross-Attention	67.1	18.75%	104.9	18.81%
CLS Expert	4.2	1.17%	5.3	0.95%
Post-Fusion Experts	134.4	37.55%	210.0	37.66%
Others (embeddings, gates, norms)	16.9	4.72%	26.2	4.70%
Total DGP Module	**357.9**	**100%**	**557.6**	**100%**

The routers and post-fusion experts constitute the majority of additional parameters ($\sim$75%), while the shared cross-attention module accounts for only $\sim$19%, demonstrating efficient parameter sharing across layers.

E Additional Analysis

E.1 Effect of Unfreezing Vision Encoder

A natural question is whether unfreezing the vision encoder during training could further improve performance. Table 10 compares DGP with frozen versus unfrozen ViT. Interestingly, unfreezing the

ViT leads to performance drops on most benchmarks (e.g., GQA, VQA^{v2}, VizWiz, SEED^{I} , POPE), while only MMB shows improvement.

We attribute this to a key property of our method: the dynamic routing mechanism relies on the *stable semantic hierarchy* across ViT layers—shallow layers encode textures and edges, while deeper layers capture high-level semantics. This layerwise division of labor is established during CLIP's large-scale pre-training on 400M image-text pairs. When the ViT is fine-tuned on our comparatively smaller dataset, this hierarchical structure may drift, causing the router to "chase a moving target." In contrast, freezing the ViT provides a stable visual foundation, allowing the router and fusion modules to learn consistent layer-selection strategies.

Table 10: **Effect of unfreezing the vision encoder.** Comparison between DGP with frozen and unfrozen (‡) vision encoder during training.

Method	LLM	*Academic-oriented VQA*				*LMM benchmarks*						
		GQA	SQA^{I}	VQA^{v2}	VizWiz	SEED	SEED^{I}	LLaVA^{W}	MM-Vet	MMB	MMB^{CN}	POPE
LLaVA-1.5 [26]	Vicuna-7B	62.0	66.8	78.5	50.0	58.6	66.1	63.4	30.5	64.3	58.3	85.9
DGP	Vicuna-7B	64.6	71.0	79.7	53.1	61.5	67.5	66.5	33.8	66.7	60.5	87.0
DGP ‡	Vicuna-7B	64.1	71.0	78.9	50.2	59.7	65.3	63.5	33.7	68.3	59.4	84.8

E.2 Full Routing Weight Matrix

Table 11 presents the complete routing weight matrix on the POPE benchmark, showing the average attention each LLM layer assigns to each ViT layer. Values are percentages (summing to 100 per row). Several patterns emerge: (1) different LLM layers exhibit distinct preferences—e.g., LLM layer 3 concentrates 58.8% on ViT layer 13, while LLM layer 19 spreads mass across shallow layers 1–3; (2) middle ViT layers (7–13) receive high weights from many LLM layers, suggesting they provide a useful balance between low-level details and high-level semantics; (3) the routing distribution is highly non-uniform, confirming that DGP learns meaningful layer-selection strategies rather than degenerating into uniform averaging.

E.3 Quantitative Evidence for Routing Patterns

To provide causal evidence that the learned routing patterns are functionally meaningful, we conduct inference-time intervention experiments on the trained DGP-7B model *without retraining*.

Experimental Setup. We compare four routing strategies: (1) **DGP (Learned)**: the learned 24-dim routing weights; (2) **Baseline (Last-2)**: LLaVA-1.5 default using only the penultimate ViT layer; (3) **Uniform**: equal weights across all 24 ViT layers; (4) **Reversed**: flipping the learned routing weights while retaining access to all ViT layers.

Findings. As shown in Table 12, (i) *Learned* $>$ *Baseline* $>$ *Uniform*: Dynamic routing consistently outperforms both single-layer baseline and naive averaging, confirming the necessity of selective multi-layer fusion. (ii) *Reversed degrades but does not collapse*: Since each LLM layer still accesses the same ViT layers via its trained router, performance remains above Uniform. However, perturbing the weight distribution leads to consistent degradation (e.g., LLaVA^{W}: 66.5$\rightarrow$57.0, MMB: 66.7$\rightarrow$61.7). These results demonstrate that performance depends not only on *which* ViT layers are accessible, but also on *how* they are weighted, providing causal evidence that the routing patterns are functionally meaningful.

E.4 Additional Visualizations

Figure 8 shows additional ViT attention heatmaps illustrating how attention shifts from low-level edges to high-level semantics across layers. Figure 9 presents more case-study comparisons between DGP and LLaVA-1.5, demonstrating improved fine-grained reasoning. Figure 10 shows DGP gated-fusion feature maps across all 32 LLM layers, illustrating improved spatial structure and semantic coherence. Figure 11 compares fixed single-layer, uniform-average, and DGP gated-fusion features, showing that our method yields sharper boundaries and more coherent object segmentation. Figure 12 further illustrates question-aware routing: for the same image, different questions trigger distinct

Table 11: **Full routing weight matrix on POPE.** Each cell shows the percentage weight ($\times 100$) that LLM layer l assigns to ViT layer v. Rows sum to 100.

LLM Layer	ViT Layer 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	2.4	2.1	1.9	2.9	16.8	2.2	38.3	2.7	5.8	6.8	2.8	2.2	1.8	1.5	1.6	1.4	1.3	1.1	0.9	0.8	0.7	0.6	0.6	0.6
2	2.0	2.0	1.8	2.6	2.3	2.3	2.7	4.6	4.8	5.1	6.3	4.0	19.1	12.1	10.2	9.5	2.8	1.6	1.0	0.8	0.7	0.6	0.5	0.6
3	1.7	1.5	1.3	1.3	1.4	1.3	1.5	1.5	1.8	1.9	2.1	11.0	58.8	2.5	2.5	1.7	1.4	1.1	1.0	0.7	0.6	0.5	0.4	0.5
4	1.8	1.8	1.7	1.9	1.9	2.0	3.0	3.5	4.3	2.9	3.4	11.6	44.9	2.6	3.3	2.7	1.7	1.2	1.0	0.8	0.6	0.5	0.4	0.5
5	1.7	1.7	1.6	1.6	1.5	1.6	1.8	2.0	3.2	2.6	4.1	20.8	11.3	29.9	3.8	3.2	1.9	1.5	1.0	0.8	0.7	0.6	0.5	0.7
6	1.8	1.8	1.6	1.7	1.9	2.0	4.7	3.7	39.9	21.8	3.0	2.5	2.3	1.6	1.7	1.7	1.2	1.2	0.9	0.7	0.6	0.6	0.6	0.6
7	1.4	1.3	1.1	1.3	1.5	1.7	2.3	3.3	5.3	6.1	3.1	21.6	30.8	4.9	3.2	2.1	2.5	1.8	1.5	0.9	0.7	0.6	0.5	0.7
8	1.9	1.9	1.9	2.2	2.2	2.8	3.2	4.8	4.4	11.5	6.1	34.6	8.7	2.9	2.3	1.8	1.6	1.2	0.9	0.8	0.7	0.5	0.5	0.6
9	2.6	2.4	2.1	2.8	3.5	12.5	43.8	3.5	3.1	6.5	2.4	2.2	2.0	1.5	1.5	1.4	1.2	1.0	0.9	0.8	0.6	0.6	0.5	0.6
10	1.7	1.5	1.5	1.7	1.6	1.7	2.1	2.1	19.5	44.5	4.7	3.6	3.0	1.9	1.5	1.3	1.1	1.0	0.8	0.8	0.7	0.6	0.6	0.6
11	1.8	2.0	1.8	2.2	2.0	2.2	3.5	4.2	28.3	5.9	7.5	11.7	13.7	2.4	2.2	1.7	1.3	1.1	0.9	0.9	0.8	0.6	0.6	0.6
12	1.9	2.1	2.3	2.8	2.5	3.3	11.6	5.3	18.9	4.4	26.1	3.5	3.7	1.9	1.7	1.6	1.2	1.0	0.9	0.8	0.7	0.6	0.6	0.7
13	1.5	1.6	1.5	1.7	1.8	1.9	2.2	2.5	3.3	2.9	3.3	31.4	33.9	1.8	1.4	1.2	1.1	0.9	0.8	0.8	0.7	0.6	0.6	0.6
14	1.6	1.6	1.6	2.2	2.4	2.5	22.4	5.0	26.0	8.1	10.4	2.6	2.3	1.6	1.4	1.4	1.3	1.1	1.0	0.9	0.8	0.7	0.7	0.7
15	1.8	1.9	1.9	2.6	2.8	2.6	2.5	15.7	14.0	36.1	2.8	2.5	2.3	1.6	1.2	1.2	1.0	0.9	0.8	0.8	0.8	0.7	0.7	0.7
16	1.6	1.4	1.6	1.5	1.7	2.0	1.9	2.7	2.1	3.0	5.9	3.0	5.5	10.0	37.9	7.6	3.8	2.5	1.3	0.9	0.7	0.6	0.4	0.5
17	2.3	1.1	1.2	1.1	1.0	1.2	1.2	1.1	1.2	2.1	1.8	4.6	1.6	38.6	14.9	8.7	4.8	4.1	2.5	2.1	0.8	0.7	0.6	0.7
18	2.1	2.5	3.6	8.8	10.6	5.9	2.1	3.1	3.9	3.2	5.9	17.0	8.1	2.3	5.3	5.0	2.7	3.6	1.3	0.8	0.7	0.5	0.3	0.7
19	26.2	12.6	17.2	5.2	4.0	2.9	1.7	3.0	2.5	1.6	1.7	3.9	2.8	3.5	1.7	1.6	1.4	1.4	1.0	0.8	0.5	0.6	0.5	1.6
20	12.1	17.5	11.9	7.5	8.7	4.0	1.9	2.3	1.4	2.0	2.9	8.8	4.1	2.0	2.9	1.6	1.9	1.6	1.4	1.1	0.5	0.4	0.9	0.7
21	4.9	4.6	3.1	2.5	2.0	1.6	2.0	1.8	1.9	1.1	6.5	2.2	2.0	16.8	35.1	3.7	2.0	1.3	1.2	0.9	0.9	0.6	0.5	0.5
22	4.5	3.3	1.6	1.7	2.0	3.3	1.1	2.0	8.1	8.2	1.7	2.9	1.8	2.2	2.4	4.6	3.7	36.3	4.6	1.0	1.5	0.3	0.4	0.5
23	2.3	1.7	1.8	2.2	2.0	1.8	2.5	3.9	25.3	9.8	7.0	4.2	9.4	10.4	4.3	4.3	1.2	1.5	0.9	0.7	0.5	0.6	0.8	0.9
24	4.4	3.1	3.9	53.8	9.0	2.7	3.4	1.7	1.6	1.7	2.6	1.6	1.0	1.1	0.7	0.9	0.7	0.8	0.7	0.7	0.7	0.7	1.1	1.1
25	0.9	0.9	0.8	0.8	0.8	0.9	0.7	0.8	1.3	0.9	0.8	1.7	6.4	72.5	1.7	1.9	1.4	0.8	0.7	0.7	0.6	0.8	0.6	0.5
26	1.9	1.7	1.3	1.0	1.4	0.9	6.2	8.2	4.6	13.3	7.9	13.9	9.6	8.5	2.2	2.7	4.1	3.3	2.6	1.4	0.7	0.4	0.6	1.5
27	1.2	1.2	1.3	1.7	3.1	3.3	4.7	2.5	11.0	54.0	2.9	1.8	1.8	1.0	0.8	1.0	0.8	0.9	0.7	0.8	0.6	0.8	0.7	1.4
28	1.9	1.7	1.3	2.5	1.5	2.3	3.4	62.6	3.4	2.7	3.5	2.4	2.4	1.5	1.0	0.7	0.5	0.6	0.7	0.8	0.6	0.6	0.5	0.7
29	1.1	1.1	1.1	1.5	1.3	1.1	1.1	1.0	1.6	54.6	12.2	5.7	3.7	3.3	1.2	1.1	1.1	1.0	1.2	1.2	0.8	0.7	0.7	0.6
30	1.5	1.4	0.8	1.1	1.0	0.5	0.8	0.9	0.9	1.4	1.3	1.0	1.1	1.3	11.8	4.9	50.0	7.0	4.0	1.3	1.7	1.6	1.8	0.9
31	2.9	2.3	1.5	1.2	1.0	1.1	1.4	2.1	6.9	51.1	10.3	3.5	3.6	2.2	1.2	1.1	0.8	0.6	1.1	0.9	0.8	0.8	0.7	1.0
32	3.4	1.0	3.4	7.8	6.4	1.6	2.7	1.0	2.6	3.8	1.5	0.4	0.4	9.6	14.1	20.2	2.2	0.8	1.6	1.0	1.7	4.2	6.0	2.6

Table 12: **Inference-time routing intervention.** Comparison of different routing strategies on DGP-7B without retraining.

Routing	*Academic-oriented VQA* GQA	SQA^I	VQA^{v2}	VizWiz	*LMM benchmarks* SEED	SEED^I	LLaVA^W	MM-Vet	MMB	MMB^{CN}	POPE
DGP (Learned)	**64.6**	**71.0**	**79.7**	**53.1**	**61.5**	**67.5**	**66.5**	**33.8**	**66.7**	**60.5**	**87.0**
Baseline (Last-2)	62.0	66.8	78.5	50.0	58.6	66.1	63.4	30.5	64.3	58.3	85.9
Uniform	60.2	64.3	72.0	50.2	57.9	63.3	56.3	25.3	57.6	44.9	77.2
Reversed	63.6	67.5	77.8	52.4	61.3	67.1	57.0	32.1	61.7	55.8	86.6

routing patterns across ViT layers. Figure 13 shows routing weight distributions at LLM layer 32 across different benchmarks, revealing that distinct tasks favor different ViT-layer combinations. Figures 14 and 15 provide additional routing weight distributions for various samples, showing that different inputs produce different weighting patterns.

F Limitations and Future Work

The Dynamic Gated Pathways module is designed to be *plug-and-play*: it introduces only lightweight, layer-wise routing components that interface with the LLM hidden states, without modifying the vision encoder or the LLM architecture itself. This modularity suggests that DGP should generalize across vision encoders and LLM backbones, adaptively selecting appropriate visual granularities regardless of the specific encoder–decoder pairing.

In this work, we validate DGP primarily under the LLaVA-1.5 setting (CLIP + Vicuna-7B/13B) and further on smaller-scale alternative frameworks, including Qwen3-VL-2B (SigLIP2 + Qwen3-1.7B) and LLaVA-NeXT (AnyRes tiling). Due to resource and time constraints, however, experiments on

larger LLMs (30B+), broader families of vision encoders, and larger-scale multimodal corpora remain limited. Extending validation along these axes constitutes our primary future work. We anticipate that the dynamic routing mechanism will remain effective—and potentially more beneficial—as model capacity and data scale increase, since richer language representations should enable even finer-grained visual-layer selection.

G Broader Impacts

This work improves hierarchical visual access in MLLMs, which may benefit applications such as accessibility tools and educational assistants. At the same time, stronger visual reasoning could be misused for surveillance or misleading content, and DGP may inherit biases from the pretrained vision and language backbones it fine-tunes. We therefore view DGP as an architectural improvement for research use under the licenses and safety practices of the host models.

H Use of Large Language Models

For transparency, we disclose that general-purpose large language models (LLMs) were used during the preparation of this manuscript *solely as writing-assistance tools* to improve the clarity, concision, and grammaticality of the text. Their use was restricted to language-level editing: rewriting sentences for readability, tightening phrasing, polishing transitions, and correcting typos. No LLM was used to generate scientific claims, design experiments, produce technical content, or decide on methodology; all results, figures, analyses, and conclusions were conceived and verified by the authors, who take full responsibility for the content of every paragraph.

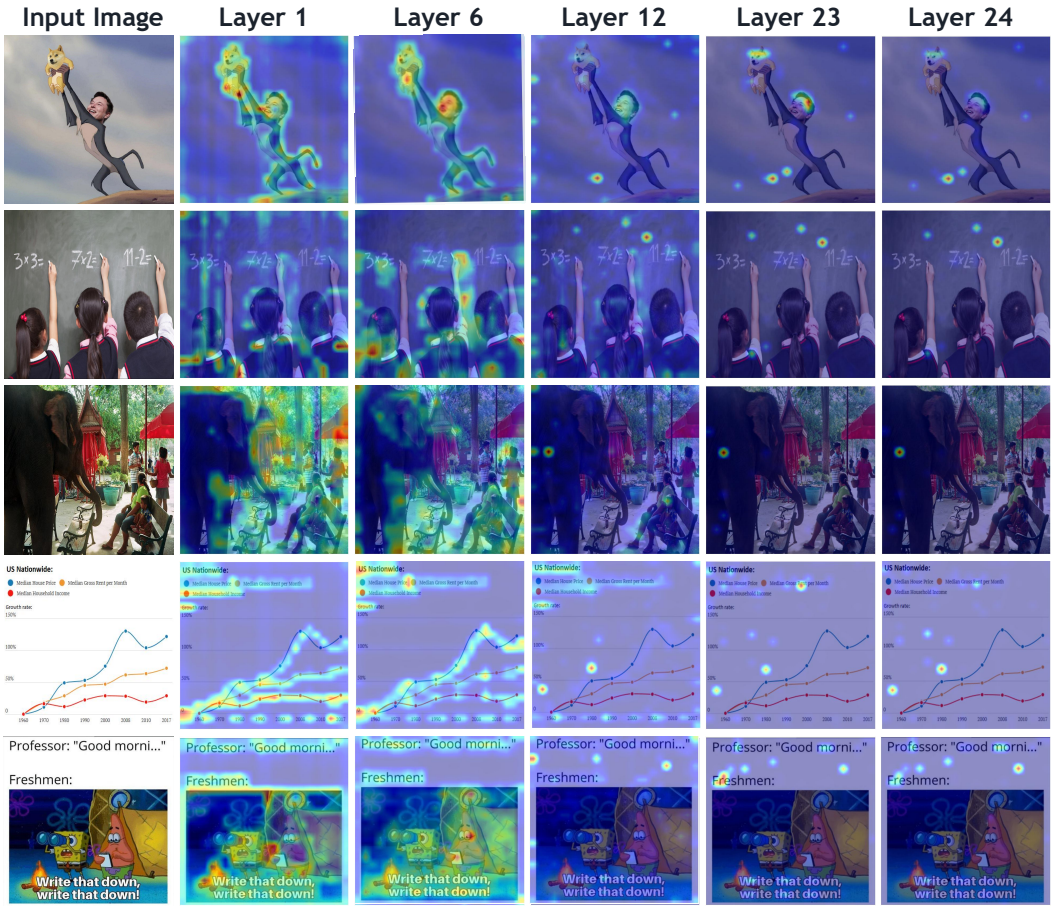


Figure 8: **Additional attention visualizations across ViT layers.** Supplementary heatmaps showing the progression of attention patterns from shallow to deep ViT layers.

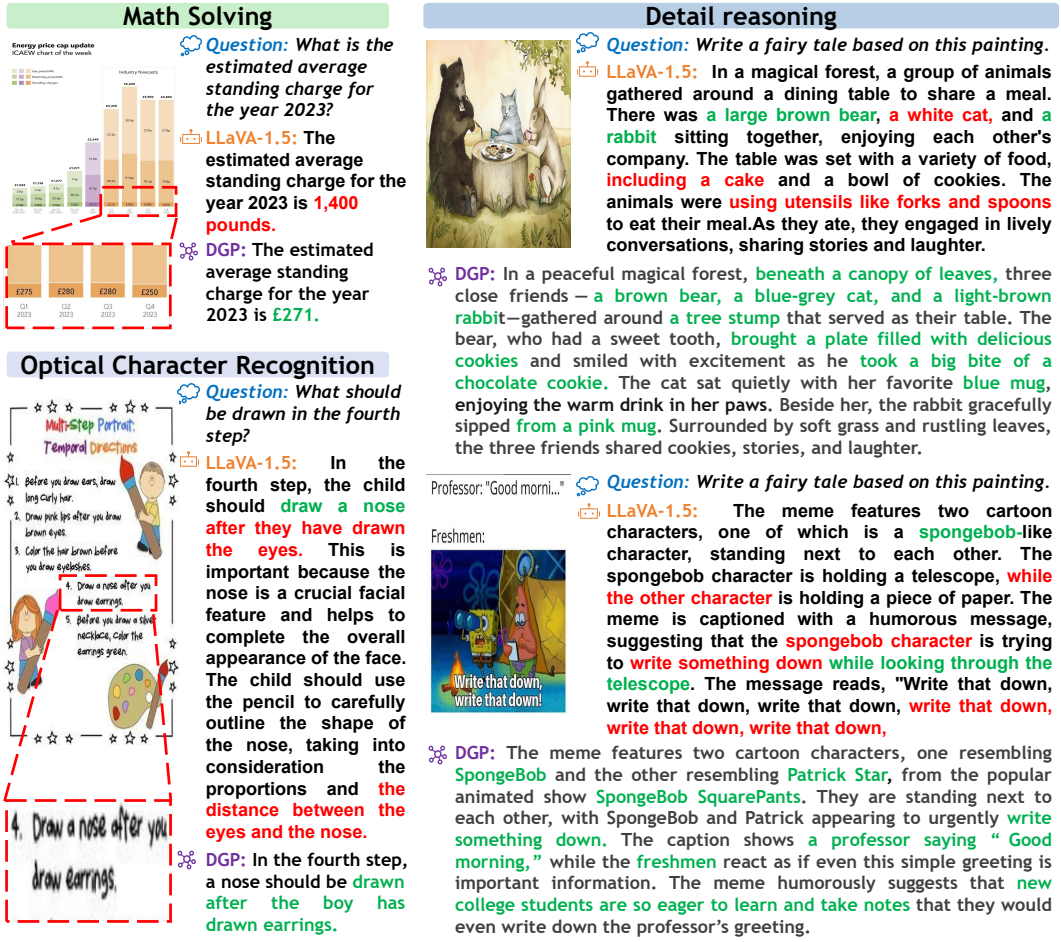


Figure 9: **Additional qualitative comparison examples.** More case studies comparing DGP with LLaVA-1.5 across diverse visual understanding tasks.

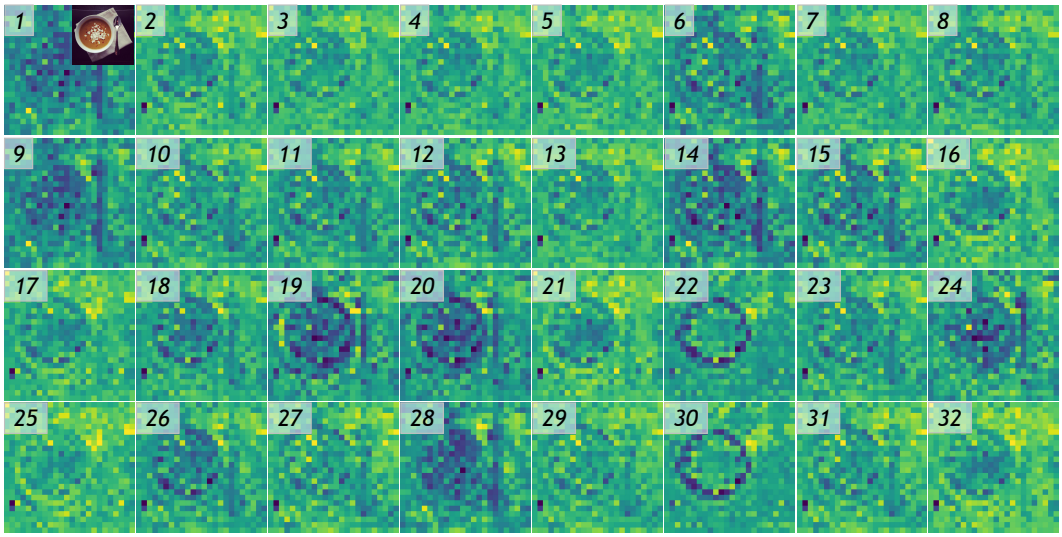


Figure 10: **Additional feature visualizations (Part 1).** Supplementary DGP gated-fusion feature maps across all 32 LLM layers, illustrating improved spatial structure and semantic coherence.

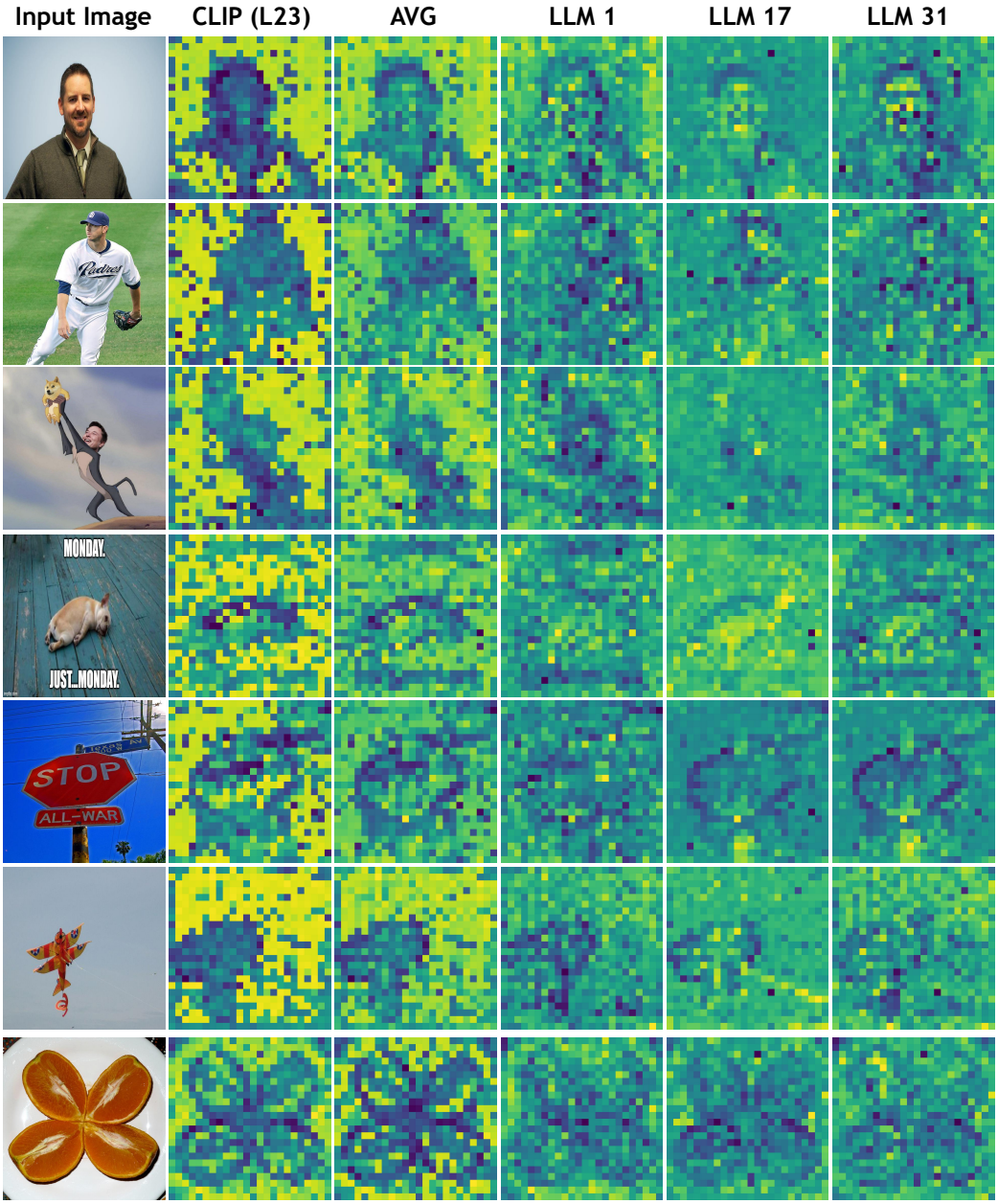


Figure 11: **Additional feature visualizations (Part 2).** More comparisons of fixed single-layer features, uniform average fusion, and DGP's gated fusion at different LLM layers.

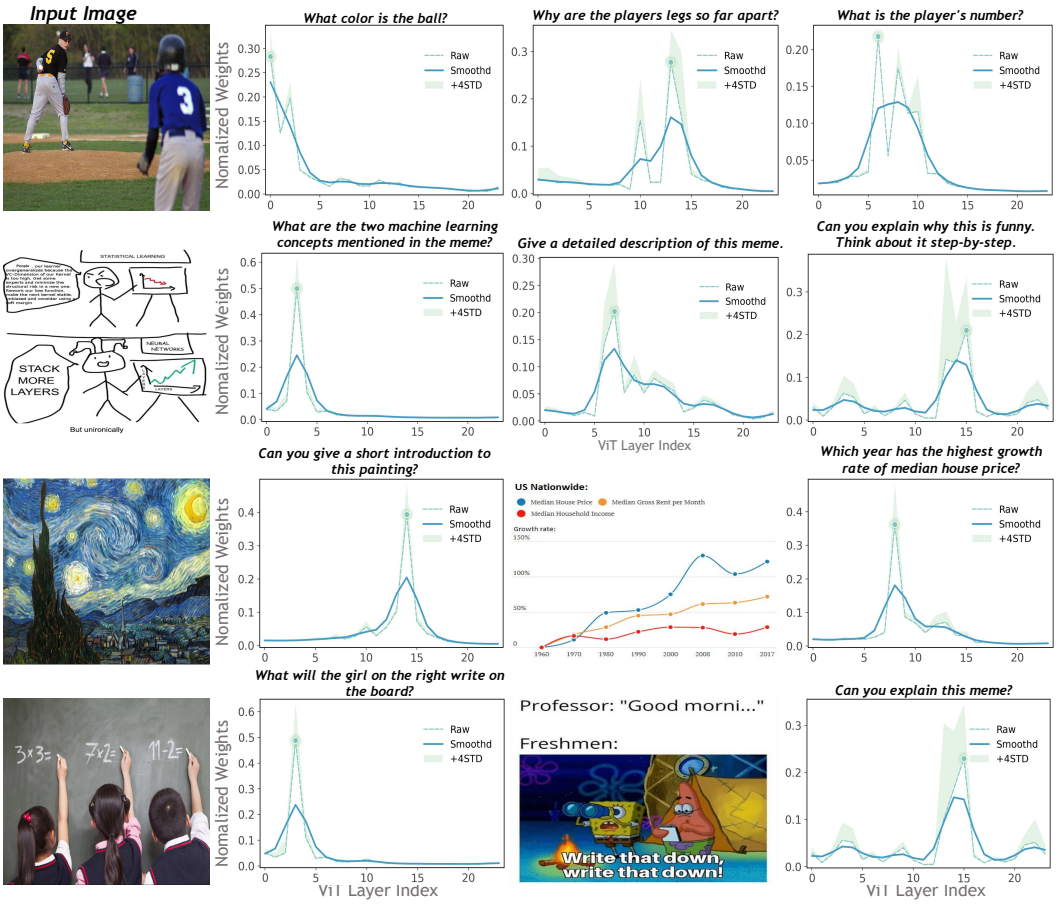


Figure 12: **Additional question-aware routing analysis.** More examples showing how routing weights shift as the question varies for the same image.

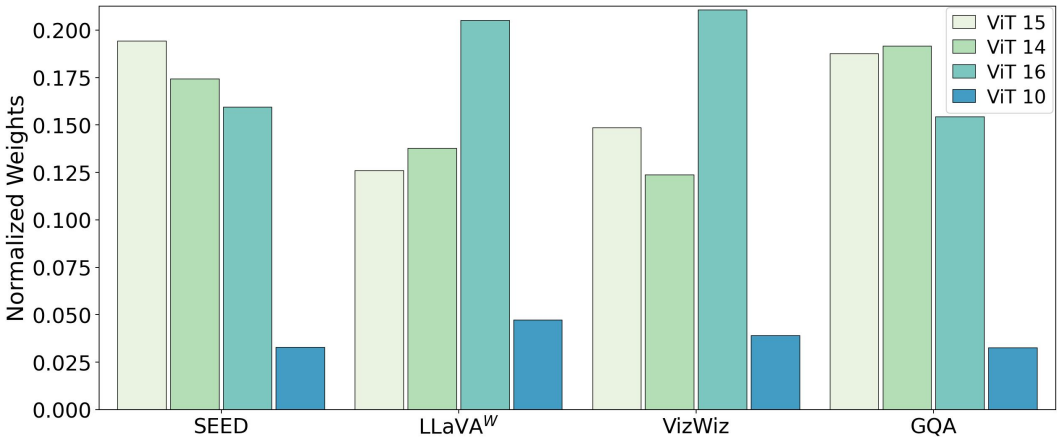


Figure 13: **Routing weight distributions at LLM layer 32.** Weights that the last LLM layer assigns across ViT layers on different benchmarks, showing that distinct tasks favor different ViT-layer combinations.

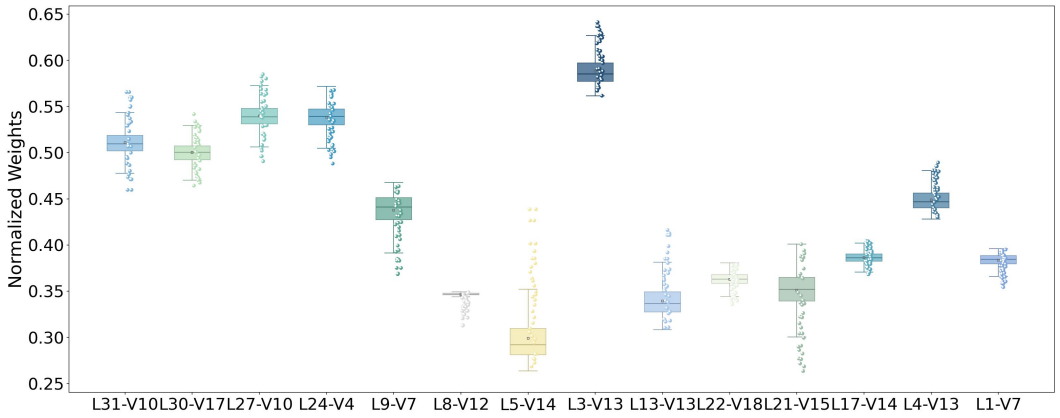


Figure 14: **Additional routing analysis (Part 1).** Supplementary visualizations of sample-wise routing weights.

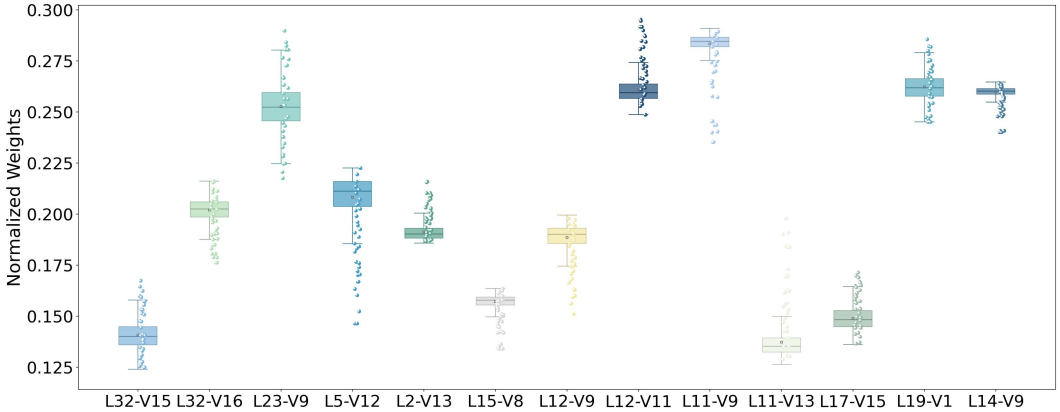


Figure 15: **Additional routing analysis (Part 2).** More examples showing the diversity of routing patterns across different inputs.

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: Abstract and Section 1 state three claims.

   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 F discusses limitations including: (1) primary validation on LLaVA-1.5 and smaller-scale alternative frameworks (e.g., Qwen3-VL-2B, LLaVA-NeXT), and (2) scaling to larger LLMs (30B+) and larger multimodal corpora remains future work.

   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: This paper proposes an empirical framework (DGP) for dynamic ViT-LLM layer routing and conducts extensive experimental validation. No formal theorems or proofs are claimed.

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 and Appendix B provide complete details including: model architecture (CLIP-ViT-L/14-336, Vicuna-7B/13B), all hyperparameters (learning rate, batch size, training epochs), the LLaVA-1.5 training data, and Algorithm 1 with full pseudocode. Table 5 details tensor dimensions at each stage. We additionally report generalization experiments on Qwen3-VL-2B and LLaVA-NeXT.

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: Code and reproduction instructions are available at `https://github.com/YingyingYan1/DGP`. Model weights are hosted at `https://huggingface.co/Yingying11/DGP`. Training data (LLaVA-1.5 instruction data) and evaluation benchmarks (VQAv2, GQA, VizWiz, ScienceQA, SEED, LLaVA-Wild, POPE, MMBench, MM-Vet, etc.) are publicly available.

   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 specifies the experimental setup including optimizer (AdamW), learning rate schedule, batch size, and training epochs. Appendix B provides additional details on training configuration and benchmark descriptions.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Following standard practice in vision-language model evaluation, we report single-run results due to the substantial computational cost of training MLLMs. This is consistent with prior work including LLaVA-1.5 and InstructBLIP.

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: Appendix B states that all training and efficiency measurements are conducted on NVIDIA L40-48GB GPUs. Table 3 reports parameter, FLOPs, and peak-memory overhead for the 7B and 13B models, and Appendix D further breaks down the `DGP` module parameters by component.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses publicly released MLLM checkpoints and public benchmarks, involves no human subjects and no personally identifiable information. We do not collect or release any new scraped data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses potential positive and negative societal impacts in Appendix G. The positive impacts include improved visual grounding for applications such as accessibility tools and educational assistants; the negative impacts include possible inherited biases from pretrained models and potential misuse for surveillance or misleading multimodal content.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release any newly pre-trained foundation model, image generator, or scraped dataset. Our contribution is an architectural improvement (DGP) that fine-tunes existing public models (e.g., CLIP, Vicuna, Qwen3), inheriting their original safety measures and license terms.

    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 third-party assets are credited via citations; all are used within their public research-use licenses.

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: The `DGP` module architecture is fully documented in Section 3 and Appendix A. Code, reproduction instructions, and model checkpoints with documentation are already available at `https://github.com/YingyingYan1/DGP` and `https://huggingface.co/Yingying11/DGP`.

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: No need.

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 human-subject research, so no IRB or equivalent approval is required.

    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: [Yes]

    Justification: As disclosed in Appendix H, general-purpose LLMs were used solely as writing-assistance tools for improving clarity and grammar. No LLM was used for generating scientific claims, designing experiments, or producing technical content.

    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.