UNBUNDLING THE VISION ENCODER IN MULTIMODAL LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

How multimodal LLMs should process images remains unsettled. Most rely on a vision encoder optimized separately for its own objective, while recent recipes train the encoder from scratch alongside the language model or remove it entirely. We view the vision encoder as a bundle of three choices that these recipes keep or discard: a visual prior from separate pretraining, dedicated vision-only capacity, and placement of that capacity before the multimodal backbone (pre-fusion) rather than inside it (in-fusion). Because published models also differ in data, backbone, and compute, we test each choice in a controlled study of 1.7B–8B multimodal LLMs trained from scratch on common data, evaluated on downstream visual question answering and language benchmarks at matched *lifetime compute*. An encoder trained from scratch with the backbone matches pretrained ones, so separate pretraining offers little advantage once its compute is counted. What matters instead is dedicated capacity: encoder-free models trail on visual understanding even with the encoder's parameters added to their backbone. Yet that capacity need not precede fusion, as vision-only modules in early backbone layers (in-fusion) match or slightly exceed the encoder given longer training, while lowering time to first token. For multimodal LLMs trained from scratch, our findings favor a single training stage with dedicated vision capacity early in the network.

1 INTRODUCTION

Most multimodal LLMs acquire visual understanding from a vision encoder optimized separately for its own objective, such as image recognition or image–text alignment, rather than for the language model that consumes its features (He et al., 2016; Radford et al., 2021; Tschannen et al., 2025; Siméoni et al., 2025; Tschannen et al., 2023). Multimodal training has progressed from freezing this encoder (Alayrac et al., 2022; Li et al., 2023b; Liu et al., 2023a) to updating it (Steiner et al., 2024; Bai et al., 2025b), but separate pretraining and architectures persist. Recent recipes bring vision and language closer by discarding parts of the encoder: Kimi K3 trains its encoder from scratch jointly with the multimodal model (Kimi Team, 2026), and encoder-free models project image patches directly into a shared multimodal backbone (Adept AI, 2023; Diao et al., 2024; Gemma Team, 2026; Thinking Machines Lab, 2026). These recipes disagree on what the vision encoder contributes. We ask which of its inherited design choices still benefit multimodal LLMs.

We propose viewing the vision encoder as a bundle of three design choices: a visual *prior* obtained through separate pretraining, dedicated *capacity* in parameters that only vision tokens use, and *pre-fusion* placement of that capacity, in a separate module that runs before vision and text tokens first interact—rather than *in-fusion*, inside the shared backbone. A pretrained encoder keeps all three, a from-scratch encoder gives up only the prior, and an encoder-free model gives up all three. Comparing a from-scratch encoder with an encoder-free model thus conflates capacity and placement.

Existing comparisons cannot isolate these choices, for two reasons. First, they entangle the choices with each other and the rest of the recipe. Published models differ in data, training stages, compute, and backbone. Controlled studies fix multimodal training data and compute, yet comparisons among pretrained encoders (McKinzie et al., 2024; Tong et al., 2024a; Fan et al., 2025; Li et al., 2025) vary which visual prior an encoder imports, not whether it needs one. Studies of encoder-based versus encoder-free models either change all three choices (Tong et al., 2026; Liu et al., 2026) or couple capacity with placement and report mainly validation loss—a proxy for downstream

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

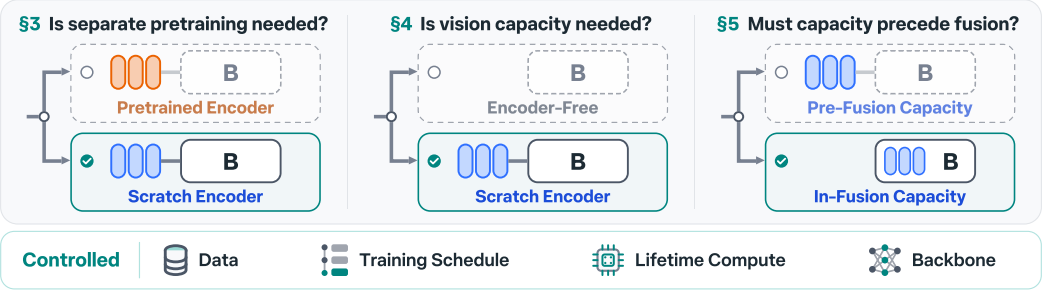

Figure 1: **A controlled study of the vision encoder's design choices in multimodal models.** Each panel asks one question about the encoder, answered in the section given in its title. Pre-fusion capacity lies before the multimodal backbone, and in-fusion capacity inside it. B denotes the backbone, and checks mark the designs our experiments favor. We train every stage from scratch, and the bottom row lists what our study controls.

performance (Shukor et al., 2025; Wang et al., 2026). Second, comparisons with pretrained encoders count only multimodal training, treating encoder pretraining as free despite its cost (*e.g.*, SigLIP 2 trains on 40B image–text samples; Tschannen et al., 2025), even though several studies attribute the encoder's advantage to its pretraining (Lei et al., 2025; Diao et al., 2025; Wang et al., 2026).

We progressively unbundle the vision encoder in a controlled study, testing one design choice at a time (Figure 1): (i) *does the vision encoder need separate pretraining?* (ii) *does visual understanding need dedicated capacity?* and (iii) *does that capacity need to precede multimodal fusion?* We train 1.7B–8B models entirely from scratch, including encoder pretraining, on common data sources with one backbone and optimization recipe, and evaluate them on downstream visual question-answering (VQA) and language benchmarks. Because recipes spend compute in different stages, we compare them at matched *lifetime compute*: the total compute of every training stage, including any separate pretraining of the encoder or language model.

Our experiments answer the three questions with three main takeaways:

1. **Separate pretraining is optional** (§3). At matched lifetime compute, a from-scratch encoder matches pretrained encoders on both text and VQA benchmarks, and leads a pretrained contrastive encoder at 4B and 8B. More encoder pretraining improves recognition but not VQA.
2. **Dedicated vision capacity is key** (§4). Encoder-free models trail encoder-based ones on visual understanding, even with the encoder's parameters added to the encoder-free backbone. The encoder's advantage thus lies in capacity dedicated to vision rather than in total parameter count.
3. **Vision capacity can move into the backbone** (§5). Additive vision-only modules in the earliest backbone layers, with the encoder's parameter budget, match or slightly exceed the encoder recipe on visual understanding, with lower time to first token.

Together, these findings show that multimodal LLMs benefit from dedicated vision capacity early in the model, but not necessarily from a separately pretrained encoder in front of it.

2 A CONTROLLED PROTOCOL FOR COMPARING MULTIMODAL RECIPES

We propose a protocol that varies the vision encoder's three design choices while holding data sources, optimization, compute, and the backbone fixed, in order to isolate their effects and compare current multimodal LLM recipes. Published models cannot be compared this way, because they can differ in multiple factors such as if they are built on different pretrained language models and vision encoders that each have their own data and compute. We therefore run every training stage ourselves, starting from random initialization, use common data sources and training setups, and compare recipes at matched *lifetime compute*, the total compute of all training stages.

Model. Every recipe shares one backbone architecture, a decoder-only Transformer (Vaswani et al., 2017) trained from random initialization at 1.7B, 4B, and 8B parameters. It is trained autoregressively with a cross-entropy loss on text tokens; vision tokens precede the text they condition and carry no loss. Recipes differ in how image patches reach this backbone, which we describe relative to the *fusion boundary*, the first layer in which vision and text tokens can interact: vision-only computation before it is *pre-fusion*, and computation inside the shared backbone is *in-fusion* (Figure 1). The encoder

recipe processes patches pre-fusion with a 27-layer, 0.4B-parameter vision Transformer (Dosovitskiy et al., 2021) whose output is linearly projected into the backbone; pretrained encoders share this architecture and are pretrained by us rather than taken from external checkpoints. The encoder-free recipe maps patches directly into the backbone with a linear patch embedding (Adept AI, 2023; Gemma Team, 2026), followed by RMSNorm and a learned scalar gate (Qi et al., 2025; Li et al., 2026). Section A.1 gives the full architectures of the backbone and the encoder, including image resolution and tokenization, and Section A.2 shows that this encoder gives the best Avg. VQA among the shapes and sizes we tested at matched lifetime compute.

Data. Text-only sequences come from Nemotron-CC (Su et al., 2025), and image–text sequences use MetaCLIP image–text pairs (Xu et al., 2024). Encoder pretraining uses these images with their original alt-text while multimodal training uses captions regenerated once by Qwen2.5-VL-32B (Bai et al., 2025a) and shared by every recipe. Using alt-text for encoder pretraining follows the original MetaCLIP and CapPa recipes, and it outperformed the regenerated captions in a preliminary comparison. Multimodal training mixes text-only and image–text sequences 70/30, the ratio that gave the best visual understanding in our mixture ablation (Section A.5) and is aligned with the optimal mixtures found by Han et al. (2026).

Lifetime compute. Multimodal LLMs often build on separately pretrained components whose training can even exceed the multimodal stage. For example, SigLIP 2, a commonly used encoder, is trained on 40B image–text samples (Tschannen et al., 2025), and language models are typically trained on trillions of tokens (Touvron et al., 2023; Yang et al., 2025). Counting only multimodal training therefore favors recipes that spend more compute in these earlier stages. To make such comparisons fair, we propose *lifetime compute*, the total training compute of every stage that produces the final model,

$$C_{\text{lifetime}} = \sum_{s \in \mathcal{S}} C_s, \tag{1}$$

where $\mathcal{S}$ is the set of training stages and C_s is the training compute of stage s. In our paper, $\mathcal{S}$ contains multimodal training and, for pretrained encoders, encoder pretraining. We report compute in ZFLOPs (10^{21} FLOPs); because recipes differ in cost per token, matched compute means that they process different numbers of tokens. Sections A.6 and A.7 detail the accounting and convert each budget into token-equivalents, the number of tokens a text-only model of the same size would process with the same compute. All recipes use a warmup–stable–decay learning-rate schedule, which lets one run be decayed at any budget (Hu et al., 2024; Wen et al., 2025; Hägele et al., 2024); reported endpoints include the decay over the final 10% of training (Section A.3).

Evaluation. We evaluate recipes on text and visual understanding benchmarks and report the following metrics. **Avg. Text** averages four category means over 12 language benchmarks: commonsense reasoning, science question answering, broad knowledge (MMLU), and reading comprehension. **Avg. VQA** averages the 16 Cambrian visual question-answering benchmarks, spanning general, knowledge, OCR and chart, and vision-centric questions, after instruction tuning each checkpoint for three epochs on Cambrian-7M (Tong et al., 2024a). For reference, a 1.7B text-only model instruction-tuned with blank images (Goyal et al., 2017; Chen et al., 2024; Brown et al., 2025) reaches 27 Avg. VQA. We also report secondary measures, ImageNet linear-probe accuracy on the multimodal model's frozen vision-token features, and text and image-to-text caption perplexities. Section B lists the benchmarks and details each evaluation.

3 Does the vision encoder need separate pretraining?

The first design choice a vision encoder bundles is a visual prior imported through separate pretraining, which recent models such as Kimi K3 forgo by training the encoder from scratch with the language model (Kimi Team, 2026). Under a fixed lifetime budget, the two options trade off directly, since every FLOP spent on encoder pretraining is a FLOP not spent on multimodal training. Whether the prior is needed also shapes the rest of the study, because encoder-free architectures cannot import one. We test whether the prior helps by comparing encoders pretrained with different objectives against the same encoder trained from scratch with the backbone.

Setup. We pretrain a separate encoder with each of four objectives, giving each the same pretraining compute: contrastive image–text alignment (MetaCLIP; Xu et al., 2024), self-supervised learning (WebSSL; Fan et al., 2025), and captioning (Cap and CapPa; Tschannen et al., 2023). Pretrained

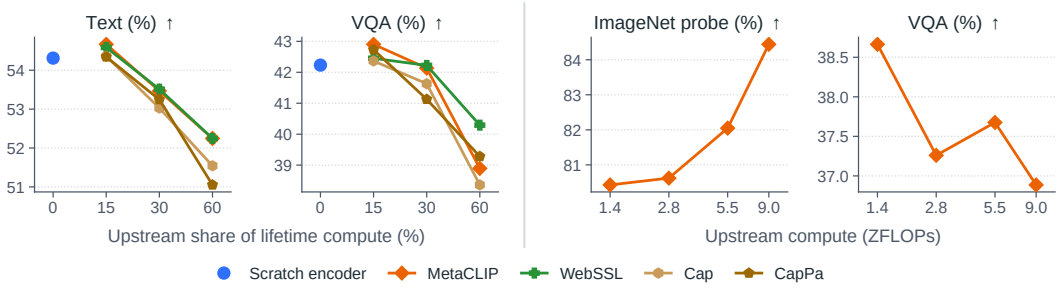

Figure 2: **More encoder pretraining does not improve visual understanding.** Left pair: at a fixed 10.4-ZFLOP lifetime budget, spending a larger share on encoder pretraining lowers Avg. Text and Avg. VQA for every objective; the from-scratch encoder (blue) spends none. Right pair: longer MetaCLIP pretraining raises the encoder's ImageNet linear-probe accuracy but does not improve Avg. VQA, even with the multimodal training budget fixed at 2.7 ZFLOPs.

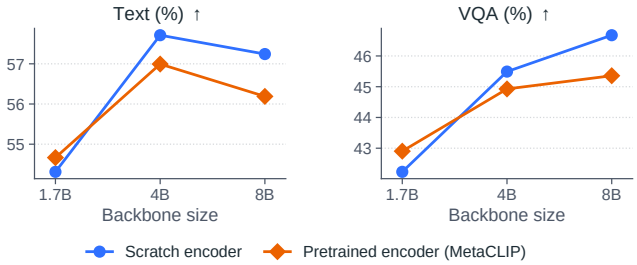

Figure 3: **The from-scratch encoder leads the pretrained MetaCLIP encoder at 4B and 8B.** We train 1.7B models to 10.4 ZFLOPs, 4B to 11.6 ZFLOPs, and 8B to 10.6 ZFLOPs, in terms of lifetime compute. Figure 9 gives per-category results.

encoders are updated during multimodal training and compared with the same encoder trained from scratch at matched lifetime compute; we also test a frozen MetaCLIP encoder. MetaCLIP, an instance of widely used CLIP-style pretraining (Radford et al., 2021; Zhai et al., 2023; Tschannen et al., 2025), serves as the pretrained reference in the remaining experiments. Section C.1 details the objectives and their pretraining and Section A.1 describes the encoder architecture.

Encoder pretraining does not help, even when it is free. Under lifetime compute budgeting, encoder pretraining takes compute away from multimodal training. At 1.7B parameter and a 10.4 ZFLOP budget, all four pretrained encoders are competitive with the from-scratch encoder when pretraining takes 15% of the budget, but performance falls as that share grows, with Avg. VQA dropping from 42 to 38–40 and Avg. Text by 2–3 points at 60% (Figure 2, left). Freezing the MetaCLIP encoder instead of updating it lowers Avg. VQA by 2.4 points (Table 15). Figure 7 follows these 1.7B models through training. Even when pretraining compute is not counted, *i.e.*, "free," more of it does not help. With multimodal training held fixed at 2.7 ZFLOPs, continuing MetaCLIP pretraining from 1.4 to 9.0 ZFLOPs raises the ImageNet linear-probe accuracy of the encoder before multimodal training from 80.4% to 84.4% (Figure 2, right), yet Avg. VQA is highest after the shortest pretraining (38.7) and lowest after the longest (36.9). Prior work advocates evaluating visual representations through multimodal LLMs (Tong et al., 2024a; Li et al., 2025); when visual understanding is the end goal, our results suggest also learning these representations jointly with the multimodal LLM, through its own image-to-text objective (Kimi Team, 2026). Sections C.2 and C.3 give the per-objective results and the full pretraining sweep, Section B.6 the ImageNet probe setup, and Table 14 the four objectives' ImageNet accuracy before multimodal training. Sections C.4 and C.5 repeat the objective comparison at a shorter budget and track how the pretrained representations change during multimodal training.

The from-scratch encoder leads at larger scales. We compare the from-scratch encoder with MetaCLIP on 4B and 8B backbones (Figure 3). Here, the from-scratch encoder leads on both Avg. Text and Avg. VQA, suggesting that the from-scratch encoder scales well with backbone size. Table 20 gives per-benchmark results, and Table 19 compares the other pretraining objectives at 4B.

Vision needs no separate training stage. The encoder therefore needs no separate pretraining stage before multimodal training. Many multimodal LLMs also add a stage in the other direction, introducing vision only after language pretraining (Alayrac et al., 2022; Liu et al., 2023a; Bai et al., 2025b). We test whether vision should enter later by training on text alone for the first 25%, 50%, or

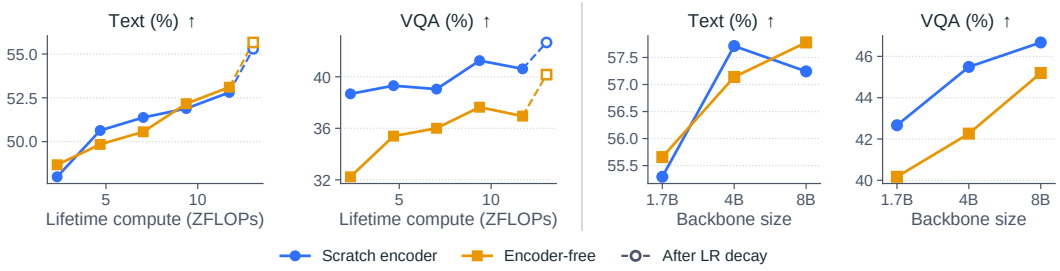

Figure 4: **The encoder leads encoder-free on visual understanding over training and across backbone sizes.** Text performance stays matched, with encoder-free ahead at 8B. Models are trained to 13.0, 11.6, and 10.6 ZFLOPs for 1.7B, 4B, and 8B. Section D.1 gives per-category results.

Architecture	**Added params (B)**		**Text (%) ↑**					**VQA (%) ↑**				
Variant	Vision-only	Shared	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
1.7B backbone												
• Encoder	0.4	–	52.5	65.0	47.2	41.5	56.4	40.2	54.6	34.5	26.9	44.9
▪ Encoder-free	–	–	52.6	65.3	46.6	42.9	55.6	36.9	50.8	34.3	17.5	45.2
+ wider	–	0.4	53.6	65.0	46.7	45.0	**57.8**	38.2	**52.5**	**36.1**	**22.0**	42.1
+ deeper	–	0.4	**53.9**	65.1	**47.9**	45.1	57.7	37.1	52.3	33.0	18.5	44.4
+ wider and deeper	–	0.4	**53.9**	**65.4**	47.4	**46.0**	57.0	**38.5**	52.1	**36.1**	19.4	**46.3**
4B backbone												
• Encoder	0.4	–	57.7	67.7	50.2	52.3	60.7	45.5	59.8	41.5	30.6	50.1
▪ Encoder-free	–	–	57.1	**68.7**	49.9	51.3	58.6	42.3	**57.7**	40.5	23.1	47.8
+ wider	–	0.4	**58.3**	**68.7**	**51.2**	**52.3**	**61.0**	**43.1**	57.5	**41.5**	**25.4**	**47.9**

Table 1: **Enlarging the encoder-free backbone does not close the gap to the encoder recipe.** The indented rows add the encoder's 0.4B parameter budget to the encoder-free's shared backbone as extra width or depth. Every expansion raises Avg. Text, and width helps Avg. VQA more than depth. Models are trained to 5.2 and 11.6 ZFLOPs for 1.7B and 4B. Text categories: commonsense (Com), science (Sci), knowledge (Knw), and reading comprehension (Rdg); VQA categories: general (Gen), knowledge (Knw), OCR and chart (OCR), and vision-centric (Vis).

75% of training before starting multimodal training, with the number of image–text examples (10% of all training examples) and lifetime compute held fixed. At 2.6 ZFLOPs, delaying multimodal training lowers Avg. VQA by about 5 points for every start we test, while Avg. Text remains fairly neutral (Section C.7). Delaying also leaves the model less reliant on vision: before the final learning-rate decay, the VQA gain from using the correct image over a shuffled one (cf. Tong et al., 2024a; Chen et al., 2024) falls from 9.4 to 3.8 points as the start moves from the first update to 75% of training (Figure 11 and Section B.3). This aligns with Han et al. (2026), who find that delaying visual integration degrades visual capabilities. Tables 22 and 23 give results for more schedules, including gradual ramps into image–text training.

Takeaway 1: Upstream encoder pretraining offers little advantage. At matched lifetime compute, an encoder trained from scratch with the backbone matches pretrained encoders and leads a pretrained contrastive encoder at 4B and 8B.

4 Does visual understanding need dedicated capacity?

The from-scratch encoder of Section 3 gives up the imported prior but keeps the second design choice, dedicated vision capacity. Encoder-free models drop this capacity and project image patches directly into the shared backbone (Adept AI, 2023; Diao et al., 2025; Gemma Team, 2026; Thinking Machines Lab, 2026). Dropping the encoder, however, also removes a substantial number of parameters and changes how raw patches enter the backbone, so a gap between the two recipes could reflect fewer total parameters, a weak patch embedding, or missing dedicated capacity. In this section, we test whether visual understanding needs this capacity, and leave its placement to Section 5.

Setup. We compare the from-scratch encoder (Section 3) with the encoder-free recipe (Section 2), which maps each patch into the backbone with a linear patch embedding followed by RMSNorm and

a learned gate. Both recipes share the backbone, data, and lifetime compute, and we compare them at 1.7B, 4B, and 8B; Section A.1 details the architectures.

Encoder outperforms encoder-free on visual understanding. As shown on the left in Figure 4, the encoder-free model trails the encoder by about six Avg. VQA points early in training and shrinks to 2.5 points by 13 ZFLOPs at the end; Avg. Text stays fairly close. The encoder also leads at 4B and 8B model scales for VQA, with encoder-free slightly outperforming on Avg. Text at 8B. Overall, the encoder holds an advantage over encoder-free on visual understanding across training and scale. The gap narrows with both training and scale, to 1.5 Avg. VQA points at 8B, so whether it closes at larger scales remains open. This differs from Shukor et al. (2025), who find that encoder-free models match or slightly outperform encoder-based ones, mainly at smaller model sizes. They compare the two mainly by validation loss, which may not track downstream performance (Tay et al., 2022; Liu et al., 2023b). In our experiments, text and caption perplexity do not reliably track Avg. VQA (Tables 22, 28 and 29).

Matching total parameters does not close the gap. We next widen or deepen the encoder-free backbone by about 0.4B parameters, the size of the encoder, and ask whether this closes the gap (Table 1). Each variant improves Avg. Text, since the added parameters are shared and also used by text tokens, and narrows the Avg. VQA gap, with larger width helping more than depth. Nevertheless, the encoder-free variants cannot match the encoder on VQA at 1.7B or 4B. The gap persists even when encoder-sized expansions train on the encoder's token count, which gives them more compute; only expansions about twice the encoder's size or larger close it, so shared capacity needs roughly twice the parameters to match the encoder's vision-only ones (Section D.2). The remaining gap is primarily in OCR and chart benchmarks, whereas the wider encoder-free is competitive with the encoder on the knowledge VQA category. In all, this experiment suggests that the same capacity is more useful for visual understanding when added as vision encoder parameters.

Stronger patch embeddings narrow the gap. Finally, we tried two richer patch embeddings inspired by Gemma 4 and Inkling (Gemma Team, 2026; Thinking Machines Lab, 2026): a normalized two-layer projection and a four-layer hierarchical MLP (Touvron et al., 2022). Both narrow the gap, and the 44M-parameter hierarchical MLP matches the encoder on Avg. VQA at 1.7B, although both also read higher-resolution images (Section D.3). While relatively small, these networks still represent a form of dedicated vision capacity, so their gains point to its value. For the following experiments, we keep the default patch embedding so that any dedicated capacity is added and placed deliberately.

> **Takeaway 2: Visual understanding benefits from dedicated capacity, not just a wider backbone.** The encoder outperforms the encoder-free across training and scale; additional backbone capacity with the same encoder parameter budget narrows, but does not close the gap.

5 Does dedicated capacity need to precede multimodal fusion?

Section 4 shows that visual understanding benefits from dedicated capacity, but leaves open the third design choice of whether that capacity must precede multimodal fusion. The encoder recipe allocates capacity pre-fusion, in a separate module before the backbone, whereas encoder-free models remove it entirely. In this section, we study whether the dedicated vision capacity can move in-fusion, inside the shared backbone. To test this, we add in-fusion vision capacity, which places the encoder's parameter budget inside the backbone as full-rank, vision-only modules: every token passes through the shared attention and feed-forward layers, and vision tokens also pass through these modules, whose outputs are summed with the shared ones. Comparing in-fusion vision capacity with encoder-based and encoder-free models shows whether dedicated capacity must precede fusion.

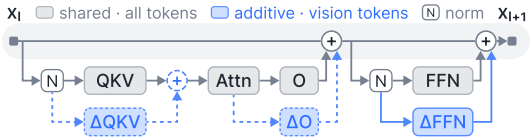

Figure 5: **A backbone block with in-fusion vision capacity.** Vision tokens use both the shared modules (gray) and their own modules (blue); in-fusion FFN omits the dashed attention modules.

5.1 Moving the encoder's capacity inside the backbone

In-fusion vision capacity augments selected blocks of the shared backbone with full-rank, vision-only modules whose outputs add to those of the shared operators (Figure 5). Every token still passes through the shared attention and feed-forward network (FFN). Unlike modality-specific experts (Wang et al., 2024; Liang et al., 2025), which replace the shared weights for vision tokens, these modules

Architecture				Text (%) ↑					VQA (%) ↑				
Variant	Design	Used by	Placement	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
■ Encoder-free		–	–	48.7	62.6	43.8	34.5	53.9	32.2	45.0	29.2	14.3	40.5
How to add													
Shared FFN widening		All	Layers 1–8	47.5	63.0	44.4	29.9	52.5	33.3	47.8	31.4	15.6	38.6
▲ In-fusion FFN		Vision	Layers 1–8	**49.2**	63.0	**44.5**	**37.1**	52.3	**37.2**	**50.3**	**32.3**	**21.6**	**44.4**
Where to add													
In-fusion FFN		Vision	Every layer (narrow)	49.0	**63.5**	**44.5**	34.4	**53.5**	32.6	41.9	27.8	18.7	42.1
In-fusion FFN		Vision	Spread (8 of 28)	48.4	61.7	44.2	36.1	51.7	33.7	43.0	30.6	16.6	**44.4**
In-fusion FFN		Vision	Layers 21–28	47.8	63.0	43.6	33.9	50.5	31.3	38.3	29.8	14.2	43.1

Table 2: **Added in-fusion capacity is most effective when it is vision-only and placed early.** All variants add 0.4B parameters to the 1.7B encoder-free backbone and are trained to 2.3 ZFLOPs at a constant learning rate. Each schematic shows backbone layers from early to late; gray is used by all tokens and blue by vision tokens only, and taller layers are widened.

add to them, so vision tokens still use every shared parameter, and each module's output rejoins the shared residual stream within the same block. For token i with vision indicator $m_i \in \{0, 1\}$, each attention projection gains a vision-only counterpart,

$$P(\mathbf{z}_i) = W_P\,\mathbf{z}_i + m_i\,\Delta W_P\,\mathbf{z}_i, \qquad P \in \{Q, K, V, O\}, \tag{2}$$

so attention remains joint across vision and text tokens, and the FFN gains a vision-only branch,

$$\mathbf{h}_i' = \mathbf{h}_i + F(N(\mathbf{h}_i)) + m_i\,\Delta F(N(\mathbf{h}_i))\,, \tag{3}$$

where F is the shared FFN, N is its normalization, and ΔF is a vision-only FFN of the same form that reads the same normalized input (Section E.1). The vision-only projections ΔW_P and the output projection of ΔF are zero-initialized, so at initialization in-fusion vision capacity equals the shared model. Because these modules act only when $m_i = 1$, text-only inputs are processed exactly as by the shared backbone.

We have two variants: in-fusion FFN uses only the FFN branch; in-fusion FFN+QKVO pairs an FFN branch of smaller hidden width with full-rank, vision-only counterparts of all four attention projections. We size both variants to match the encoder's vision-only parameter budget (Table 27), so comparisons with the encoder hold the amount of dedicated capacity fixed and vary where it sits. Existing vision-specific architectures were not designed for this comparison, as they either add duplicated vision weights that scale with the backbone (Wang et al., 2024; Liang et al., 2025) or use vision adapters that rely on an encoder (Dong et al., 2024; Wang et al., 2025).

5.2 How and where to add in-fusion capacity

First, we compare alternative ways of allocating the encoder's 0.4B-parameter budget inside the encoder-free 1.7B backbone (top of Table 2; Section E.2 adds controls such as a separate branch used by all tokens). Widening the shared FFNs of the first eight layers provides only a modest gain, from 32.2 to 33.3 Avg. VQA. Using the same parameters as a vision-only branch (in-fusion FFN) instead raises Avg. VQA to 37.2 and leaves the text computation unchanged. A separate branch used by all tokens reaches 36.5 in a short control run (Section E.2), so most of this gain comes from the separate branch, and whether routing it to vision tokens adds further benefit is unclear at this budget. The vision-only branch gives vision tokens processing of their own, which can reduce competition between the modalities for the same parameters, while both still share the backbone's attention and FFNs.

Placement of in-fusion capacity matters. With the budget fixed, the placement of in-fusion FFN strongly affects visual understanding. We compare three alternatives to the first eight layers: narrower branches in every layer, eight branches spread evenly across depth, and the last eight layers. All three reach only 31–34 Avg. VQA, close to the encoder-free model's 32.2, whereas in-fusion FFN placed in the first eight layers reaches 37.2. Dedicated in-fusion capacity is therefore most useful where the backbone first processes vision tokens, as with an encoder. We hypothesize that vision tokens benefit most from vision-only processing while they are still close to raw patch features; only the early placement concentrates the budget there, whereas the every-layer and spread placements distribute it across the entire backbone. Consistent with this, early placement helps most on OCR questions (7.3 points over the encoder-free model), which depend on fine visual detail in the raw patch features, whereas late placement gains nothing there. We use the first eight layers from here on.

Architecture	Text (%) ↑					VQA (%) ↑				
Model	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
1.7B backbone, 13.0 ZFLOPs										
■ Encoder-free	55.7	66.5	48.1	48.6	**59.4**	40.2	53.2	37.1	22.9	47.6
● Encoder	55.3	66.2	48.1	49.3	57.6	42.7	**57.0**	37.2	**30.0**	46.5
▲ In-fusion FFN	55.6	66.1	**48.5**	**50.5**	57.3	42.6	**57.0**	37.1	29.1	47.3
▼ In-fusion FFN+QKVO	**55.9**	**66.9**	**48.5**	49.3	59.0	**43.9**	56.8	**38.9**	29.9	**50.1**
4B backbone, 29.0 ZFLOPs										
● Encoder	60.5	69.6	51.3	**57.9**	**63.3**	48.8	63.6	45.9	31.0	54.8
▲ In-fusion FFN	60.2	**70.0**	52.2	56.4	62.4	48.5	63.1	44.2	32.7	53.8
▼ In-fusion FFN+QKVO	**60.7**	69.2	**52.6**	57.8	63.1	**50.0**	**63.7**	**46.5**	**33.7**	**56.0**

Table 3: **In-fusion variants are competitive with the from-scratch encoder.** in-fusion FFN+QKVO slightly outperforms the encoder on both text and VQA. All models train to 1.11T token-equivalents.

5.3 Can in-fusion capacity match a pre-fusion encoder?

The encoder and in-fusion vision capacity can now hold the same dedicated budget on either side of fusion. We compare the from-scratch encoder, in-fusion FFN, and in-fusion FFN+QKVO on 1.7B and 4B backbones at matched lifetime compute, training over five times longer than in Section 5.2.

Table 3 shows that in-fusion capacity matches the encoder at both scales and clearly exceeds the 1.7B encoder-free model on Avg. VQA. In-fusion FFN+QKVO ends about one Avg. VQA point above the encoder at both scales, with its largest gain at 4B on OCR questions, while in-fusion FFN ends even with the encoder. Both variants also match the encoder on Avg. Text. In-fusion FFN+QKVO differs from in-fusion FFN by moving part of its budget into vision-only attention projections, which help on their own, since in two 4B ablations adding them to the encoder-free model improves Avg. VQA and removing them from in-fusion FFN+QKVO lowers it (Section E.5). Over training, the encoder leads both variants early on, but in-fusion capacity catches up as training continues, and in-fusion FFN+QKVO stays ahead for most of training at 4B (Figure 14; Section E.4 lists every checkpoint).

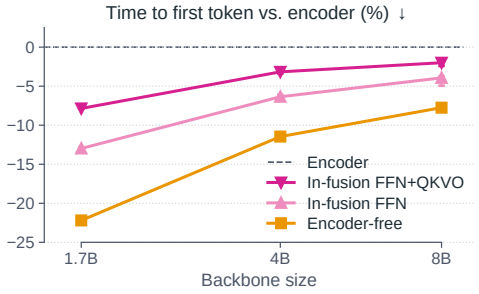

Figure 6: **Encoder-free and in-fusion vision capacity reduce time to first token relative to the encoder.** The time reduction is clearest at 1.7B and diminishes as the backbone grows. Inference is done on a H200 with 16 concurrent requests.

Moving capacity in-fusion also improves efficiency. A separate encoder must finish processing each image before the backbone can start, which adds a serial step to inference. Removing this step shortens the time to first token at every backbone size (Figure 6). The encoder-free model, which adds no vision-only computation, is fastest, both in-fusion variants are faster than the encoder, and the savings shrink as the backbone grows and vision processing becomes a smaller share of the cost. Earlier encoder-free models report similar reductions in time to first token (Luo et al., 2025b;a), although their gains also reflect other efficiency choices. Training is also slightly faster: both variants train at about the speed of the encoder-free model, whereas the encoder recipe is about 3% slower at 4B (Table 33; details in Section F).

Takeaway 3: Dedicated visual capacity can move into the backbone's earliest layers. Placed in the earliest backbone layers, additive vision-only modules match or slightly exceed a from-scratch encoder on visual understanding, with lower time to first token on inference efficiency.

6 Related work

Multimodal LLM recipes. Most multimodal LLMs connect a separately pretrained vision encoder (Radford et al., 2021; Tschannen et al., 2025; Bolya et al., 2025) to a pretrained language model, frozen (Liu et al., 2023a; Li et al., 2023b) or updated (Steiner et al., 2024; Bai et al., 2025b), while others train the encoder from scratch with the language model (Kimi Team, 2026), embed image

patches without an encoder (Adept AI, 2023; Gemma Team, 2026; Thinking Machines Lab, 2026), or quantize images with a separately trained tokenizer (Chameleon Team, 2024; Wang et al., 2026). Because these models may also differ in data, pretrained components, and compute, comparing them cannot isolate any single design choice. We therefore train every stage of each recipe from scratch on common data sources and compare the recipes at matched lifetime compute (Section 2).

Controlled studies. Prior controlled studies of encoder pretraining typically start from a pretrained language model, count only multimodal training, and compare encoders, objectives, and data (Karamcheti et al., 2024; McKinzie et al., 2024; Tong et al., 2024a; Laurençon et al., 2024; Fan et al., 2025; Li et al., 2025) rather than asking whether separate pretraining is needed at all. The few that do ask, by comparing pretrained and from-scratch encoders, disagree (Tian et al., 2025; Kimi Team, 2026; Beyer et al., 2024; Wang et al., 2026), and none matches lifetime compute. Those that favor pretraining use short training runs (Beyer et al., 2024; Wang et al., 2026), mirroring findings in object detection and language modeling that the advantage of pretraining shrinks as downstream training grows (He et al., 2019; Hernandez et al., 2021). Pretraining also confounds encoder *vs*. encoder-free comparisons, which typically set a pretrained encoder against an encoder-free model (Diao et al., 2024; Lei et al., 2025; Diao et al., 2025; Tong et al., 2026; Liu et al., 2026), removing separate pretraining, dedicated capacity, and pre-fusion placement at once, and often attribute the encoder's advantage to pretraining data they do not control. Studies that train both designs from scratch avoid this problem and find little advantage for a separate encoder, but compare mainly by validation loss (Shukor et al., 2025; Wang et al., 2026), which in our experiments does not reliably track visual understanding (Section 4).

Dedicated vision capacity. Testing whether dedicated capacity must precede fusion requires moving the encoder's capacity inside the backbone without changing its amount, which the encoder *vs*. encoder-free comparisons above cannot do. Studies of fusion depth come closer, varying where modalities begin to interact (Nagrani et al., 2021; Hao et al., 2026) or where encoder features enter the backbone (Meng et al., 2024), but none moves a fixed amount of vision-only capacity into layers that vision and text share. Other multimodal architectures do add vision-specific parameters to the backbone, but not in a form that fits this comparison. Modality-expert designs duplicate the backbone's weights for vision tokens, typically in every layer, so their dedicated capacity scales with the backbone rather than matching an encoder (Bao et al., 2022; Wang et al., 2024; Luo et al., 2025b; Tian et al., 2025; Lin et al., 2024; Diao et al., 2025; Liang et al., 2025; Shi et al., 2025). Adapter designs fine-tune a pretrained model through small parallel or low-rank modules shared by all tokens (He et al., 2022; Chen et al., 2022; Hu et al., 2022), and their multimodal variants keep an encoder alongside the adapters (Dong et al., 2024; Shapourian et al., 2026) or distill from one (Wang et al., 2025), retaining its pre-fusion capacity or pretraining. To address this gap, we add full-rank vision-only modules beside the shared ones with the encoder's parameter budget. Placed in early backbone layers, in-fusion vision capacity matches or slightly exceeds the encoder on VQA, suggesting that dedicated vision capacity need not precede fusion (Section 5).

7 Conclusion

With multimodal LLMs revisiting the vision encoder, we asked which of its design choices are still beneficial. We proposed unbundling the encoder into three such choices, a visual prior from separate pretraining, dedicated vision capacity, and a separate pre-fusion stage, and tested each by training models from scratch under a controlled protocol and matched lifetime compute. Under these controls, we find the encoder's benefit stems from its dedicated vision capacity, rather than its separate pretraining nor placement before the backbone. Pretraining the encoder separately on the same data offers little advantage over spending that compute on joint training, while its parameters help visual understanding only when dedicated to vision, whether before the backbone or in its early layers. For multimodal LLMs trained from scratch, this points to a recipe with one training stage and dedicated vision capacity in the model's early layers. Extending this study to other modalities and to post-training are promising next steps: just as optimizing for visual understanding rather than recognition changes how the encoder is best trained, visual agentic tasks may again shift which design choices matter for multimodal LLMs.

AI USE STATEMENT

Multimodal training uses image captions generated by Qwen2.5-VL-32B-Instruct (Bai et al., 2025a) (Sections A.4 and 2). LLM-based assistants (Claude and Codex) were used throughout this work under the authors' direction and review to implement methods and experiments, launch experiments, aggregate results, create data visualizations, provide feedback on experiments, propose candidate interpretations of results, search and summarize related literature, verify bibliography entries, and draft and edit the paper. We did not use generative AI tools to develop theoretical claims; formulating or proving mathematical claims, translation, and qualitative data analysis are not applicable to this work. The research questions, overall experimental design, schematic figures, and all final interpretations and claims are the authors' own. The authors reviewed all AI-assisted content and take full responsibility for the contents of this paper.

ETHICS STATEMENT

Our models are trained on web-crawled data, Nemotron-CC text and MetaCLIP image–text pairs with captions regenerated by Qwen2.5-VL-32B, and may inherit biases and harmful content present in web data and in the captioning model. The study involves no human subjects, and we use the models only to compare training recipes on existing benchmarks.

REPRODUCIBILITY STATEMENT

Section 2 describes the shared backbone, data, and lifetime-compute protocol, and Section A gives the full architectures, optimization and learning-rate schedules, training data and recaptioning, and compute accounting for every recipe. Section B details instruction tuning and all benchmarks.

REFERENCES

Adept AI. Fuyu-8B: A multimodal architecture for AI agents, 2023. URL https://www.adept.ai/blog/fuyu-8b/.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2204.14198.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025a. URL https://arxiv.org/abs/2502.13923.

Shuai Bai et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025b. URL https://arxiv.org/abs/2511.21631.

Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, Songhao Piao, and Furu Wei. VLMo: Unified vision-language pre-training with mixture-of-modality-experts. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32897–32912, 2022. doi: 10.52202/068431-2384. URL https://arxiv.org/abs/2111.02358.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. PaliGemma: A versatile 3b VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024. URL https://arxiv.org/abs/2407.07726.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. URL https://arxiv.org/abs/1911.11641.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025. URL https://arxiv.org/abs/2504.13181.

Ellis Brown, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should "train on the test set" to expose exploitable non-visual shortcuts. *arXiv preprint arXiv:2511.04655*, 2025. URL https://arxiv.org/abs/2511.04655.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. URL https://arxiv.org/abs/2405.09818.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2403.20330.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. AdaptFormer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2205.13535.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1, pp. 2924–2936, 2019. URL https://arxiv.org/abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Haiwen Diao, Yufeng Cui, Xiaotong Li, Yueze Wang, Huchuan Lu, and Xinlong Wang. Unveiling encoder-free vision-language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52545–52567, 2024. doi: 10.52202/079017-1665. URL https://arxiv.org/abs/2406.11832.

Haiwen Diao, Xiaotong Li, Yufeng Cui, Yueze Wang, Haoge Deng, Ting Pan, Wenxuan Wang, Huchuan Lu, and Xinlong Wang. EVEv2: Improved baselines for encoder-free vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21014–21025, 2025. URL https://arxiv.org/abs/2502.06788.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, et al. InternLM-XComposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *arXiv preprint arXiv:2401.16420*, 2024. URL https://arxiv.org/abs/2401.16420.

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. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, and Saining Xie. Scaling language-free visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 370–382, 2025. URL https://arxiv.org/abs/2504.01017.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. MME: A comprehensive evaluation benchmark for multimodal large language models. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025. URL https://arxiv.org/abs/2306.13394.

Itai Gat, Idan Schwartz, and Alexander Schwing. Perceptual score: What data modalities does your model perceive? In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

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*, 2017. URL https://arxiv.org/abs/1612.00837.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.18392.

Junlin Han, Shengbang Tong, David Fan, Minghao Chen, Philip Torr, Filippos Kokkinos, and Mike Lewis. Towards physics of multimodal pretraining: Knowledge flow, modality synergy, early unification, and recipes. *arXiv preprint arXiv:2608.05000*, 2026. URL https://arxiv.org/abs/2608.05000.

Jitai Hao, Hao Liu, Xinyan Xiao, Qiang Huang, and Jun Yu. Uni-X: Mitigating modality conflict with a two-end-separated architecture for unified multimodal models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.24365.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.04366.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://arxiv.org/abs/1512.03385.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. URL https://arxiv.org/abs/1811.08883.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021. URL https://arxiv.org/abs/2102.01293.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06395.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. URL https://arxiv.org/abs/1902.09506.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the design space of visually-conditioned language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.07865.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European Conference on Computer Vision*, pp. 235–251, 2016. URL https://arxiv.org/abs/1603.07396.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026. URL https://arxiv.org/abs/2607.24653.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017. URL https://arxiv.org/abs/1704.04683.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? In *Advances in Neural Information Processing Systems*, volume 37, pp. 87874–87907, 2024. doi: 10.52202/079017-2789. URL https://arxiv.org/abs/2405.02246.

Weixian Lei, Jiacong Wang, Haochen Wang, Xiangtai Li, Jun Hao Liew, Jiashi Feng, and Zilong Huang. The scalability of simplicity: Empirical analysis of vision-language learning with a single transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2504.10462.

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*, 2023a. URL https://arxiv.org/abs/2307.16125.

Bozhou Li, Xinda Xue, Sihan Yang, Yang Shi, Xinlong Chen, Yushuo Guan, Yuanxing Zhang, and Wentao Zhang. The unseen bias: How norm discrepancy in pre-norm MLLMs leads to visual information loss. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2512.08374.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.11794.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023b. URL https://arxiv.org/abs/2301.12597.

Xianhang Li et al. OpenVision: A fully-open, cost-effective family of advanced vision encoders for multimodal learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3977–3987, 2025. URL https://arxiv.org/abs/2505.04601.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2411.04996.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755, 2014. URL https://arxiv.org/abs/1405.0312.

Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. MoMa: Efficient early-fusion pre-training with mixture of modality-aware experts. *arXiv preprint arXiv:2407.21770*, 2024. URL https://arxiv.org/abs/2407.21770.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2304.08485.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 22188–22214, 2023b. URL https://arxiv.org/abs/2210.14199.

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*, 2024a. URL https://arxiv.org/abs/2307.06281.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b. doi: 10.1007/s11432-024-4235-6.

Zhiheng Liu, Weiming Ren, Xiaoke Huang, Shoufa Chen, Tianhong Li, Mengzhao Chen, Yatai Ji, Sen He, Jonas Schult, Belinda Zeng, Tao Xiang, Wenhu Chen, Ping Luo, Luke Zettlemoyer, and Yuren Cong. Tuna-2: Pixel embeddings beat vision encoders for multimodal understanding and generation. *arXiv preprint arXiv:2604.24763*, 2026. URL https://arxiv.org/abs/2604.24763.

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. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2209.09513.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.02255.

Gen Luo, Wenhan Dou, Wenhao Li, Zhaokai Wang, Xue Yang, Changyao Tian, Hao Li, Weiyun Wang, Wenhai Wang, Xizhou Zhu, Yu Qiao, and Jifeng Dai. Mono-InternVL-1.5: Towards cheaper and faster monolithic multimodal large language models. *arXiv preprint arXiv:2507.12566*, 2025a. URL https://arxiv.org/abs/2507.12566.

Gen Luo, Xue Yang, Wenhan Dou, Zhaokai Wang, Jiawen Liu, Jifeng Dai, Yu Qiao, and Xizhou Zhu. Mono-InternVL: Pushing the boundaries of monolithic multimodal large language models with endogenous visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b. URL https://arxiv.org/abs/2410.08202.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022. URL https://arxiv.org/abs/2203.10244.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE Winter Conference on Applications of Computer Vision*, 2021. URL https://arxiv.org/abs/2007.00398.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, Anton Belyi, et al. MM1: Methods, analysis and insights from multimodal LLM pre-training. In *Computer Vision – ECCV 2024*, Lecture Notes in Computer Science, pp. 304–323, 2024. URL https://arxiv.org/abs/2403.09611.

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 23464–23487, 2024. URL https://arxiv.org/abs/2406.04334.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://arxiv.org/abs/1809.02789.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In *Advances in Neural Information Processing Systems*, volume 34, pp. 14200–14213, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2304.07193.

Jianing Qi, Jiawei Liu, Hao Tang, and Zhigang Zhu. Beyond semantics: Rediscovering spatial awareness in vision-language models. *arXiv preprint arXiv:2503.17349*, 2025. URL https://arxiv.org/abs/2503.17349.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*. PMLR, 2021. URL https://arxiv.org/abs/2103.00020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. URL https://arxiv.org/abs/1907.10641.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQA: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019. URL https://arxiv.org/abs/1904.09728.

Hassan Shapourian, Kasra Hejazi, Olabode M. Sule, and Beren Millidge. ZAYA1-VL-8B technical report. *arXiv preprint arXiv:2605.08560*, 2026. URL https://arxiv.org/abs/2605.08560.

Weijia Shi, Xiaochuang Han, Chunting Zhou, Weixin Liang, Xi Victoria Lin, Luke Zettlemoyer, and Lili Yu. LMFusion: Adapting pretrained language models for multimodal generation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 10128–10147, 2025. URL https://arxiv.org/abs/2412.15188.

Mustafa Shukor, Enrico Fini, Victor Guilherme Turrisi da Costa, Matthieu Cord, Joshua Susskind, and Alaaeldin El-Nouby. Scaling laws for native multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2504.07951.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025. URL https://arxiv.org/abs/2508.10104.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. URL https://arxiv.org/abs/1904.08920.

Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, et al. PaliGemma 2: A family of versatile VLMs for transfer. *arXiv preprint arXiv:2412.03555*, 2024. URL https://arxiv.org/abs/2412.03555.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025. doi: 10.18653/v1/2025.acl-long.123. URL https://arxiv.org/abs/2412.02595.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2109.10686.

Thinking Machines Lab. Inkling: Our open-weights model. Blog post, 2026. URL https://thinkingmachines.ai/news/introducing-inkling/.

Changyao Tian, Hao Li, Gen Luo, Xizhou Zhu, Weijie Su, Hanming Deng, Jinguo Zhu, Jie Shao, Ziran Zhu, Yunpeng Liu, Lewei Lu, Wenhai Wang, Hongsheng Li, and Jifeng Dai. NaViL: Rethinking scaling properties of native multimodal large language models under data constraints. In *Advances in Neural Information Processing Systems*, volume 38, pp. 95305–95333, 2025. doi: 10.52202/085713-2868.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024a. URL https://arxiv.org/abs/2406.16860.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9568–9578, 2024b. doi: 10.1109/CVPR52733.2024.00914. URL https://arxiv.org/abs/2401.06209.

Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, et al. Beyond language modeling: An exploration of multimodal pretraining. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.03276.

Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. In *Computer Vision – ECCV 2022*, Lecture Notes in Computer Science, pp. 497–515, 2022. URL https://arxiv.org/abs/2203.09795.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Michael Tschannen, Manoj Kumar, Andreas Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. Image captioners are scalable vision learners too. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.07915.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. URL https://arxiv.org/abs/2502.14786.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Han Wang, Yongjie Ye, Bingru Li, Yuxiang Nie, Jinghui Lu, Jingqun Tang, Yanjie Wang, and Can Huang. Vision as LoRA. *arXiv preprint arXiv:2503.20680*, 2025. URL https://arxiv.org/abs/2503.20680.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Keqin Chen, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. CogVLM: Visual expert for pretrained language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121475–121499, 2024. doi: 10.52202/079017-3860. URL https://arxiv.org/abs/2311.03079.

Xinlong Wang et al. Multimodal learning with next-token prediction for large multimodal models. *Nature*, 650:327–333, 2026. doi: 10.1038/s41586-025-10041-x. URL https://doi.org/10.1038/s41586-025-10041-x.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05192.

xAI. Grok-1.5 Vision preview. https://x.ai/news/grok-1.5v, 2024.

Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying CLIP data. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16671.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.16502.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://arxiv.org/abs/1905.07830.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2303.15343.

APPENDIX

§A **Setup** (Section 2): architectures, optimization, data, compute accounting, and how we chose the encoder and data mixture.
§B **Evaluation** (Section 2): instruction tuning, VQA and language benchmarks, image controls, perplexity, and ImageNet probes.
§C **Encoder pretraining** (Section 3): objectives, upstream budgets, representation dynamics, backbone scale, and when image–text data enters training.
§D **Dedicated capacity** (Section 4): where the encoder's advantage comes from, shared-capacity controls, and richer patch embeddings.
§E **In-fusion capacity** (Section 5): in-fusion vision capacity implementation, ablation and attention-projection controls, the 4B comparison, and longer training.
§F **Systems** (Section 5.3): measurement setup, training time, inference latency, and encoder-side optimizations.
§G **Limitations** (Section 7): controlled encoder data and the scope of scales, architectures, and modalities.

A MODELS, TRAINING, DATA, AND COMPUTE

Unless stated otherwise, every comparison uses the settings below, which detail the protocol of Section 2.

A.1 ARCHITECTURES

The recipes differ in where their vision-only capacity sits and how it is trained: in a pre-fusion vision encoder that is pretrained and then frozen or updated, or trained from scratch; in no module beyond a patch embedding (encoder-free); or in vision-only modules inside the shared backbone (in-fusion vision capacity; Section E.1). Unless a quantity is the stated ablation, comparisons hold the backbone, the 256 vision tokens per image, the data, the optimization recipe, and lifetime compute fixed, and Table 4 lists the backbone and encoder architectures.

Backbone. The shared backbone is a dense decoder-only Transformer that follows the Qwen3 architecture (Yang et al., 2025) at 1.7B, 4B, and 8B parameters and is trained from random initialization. Its FFNs are SwiGLU networks, and its attention uses grouped key–value heads with RMSNorm on queries and keys. The vision-only FFN branches of in-fusion vision capacity use the same form and read the same normalized input as the shared FFN (Section E.1).

Vision encoder. The encoder follows MoonViT, the vision Transformer of Kimi K3 (Kimi Team, 2026), in depth, patch size, normalization, and size: 27 layers, patch size 14, RMSNorm, and about 0.4B parameters in total. It has width 1,024 with 4,096-wide MLPs and uses learned spatial position embeddings combined with interleaved two-dimensional rotary position embeddings. Each 224×224 image is divided into 14×14 patches, producing 256 vision tokens. For an image x, the encoder recipe computes

$$v_{\mathrm{enc}}(x) = W_{\mathrm{bridge}} f_\phi(x), \tag{4}$$

where f_ϕ is the vision encoder and the linear projection W_{bridge} maps its output width to the backbone width. Pretrained variants (Section 3) use the same architecture and differ only in the pretraining objective. MetaCLIP and WebSSL use a class token during upstream training; we discard it and pass only the 256 patch tokens to the backbone.

Encoder-free patch embedding. The encoder-free recipe projects each flattened 14×14 RGB patch linearly into the backbone width, as in Fuyu and Gemma 4 (Adept AI, 2023; Gemma Team, 2026), and applies RMSNorm and a learned scalar gate (Qi et al., 2025; Li et al., 2026), giving 256 backbone tokens per image:

$$v_{\mathrm{free}}(x) = g\ \mathrm{RMSNorm}\left(W_{\mathrm{patch}}\,\mathrm{Patchify}(x) + p_{\mathrm{2D}}\right), \tag{5}$$

where p_{2D} is a fixed two-dimensional sine–cosine position embedding and the learned scalar gate g is initialized to 0.1. The in-fusion models of Section E.1 use the same patch embedding.

	Backbone			Vision encoder
	1.7B	4B	8B	0.4B
Layers	28	36	36	27
Width	2,048	2,560	4,096	1,024
Query heads	16	32	32	12
FFN width	6,144	9,728	12,288	4,096
Tied embeddings	yes	yes	no	–
Parameters (B)	1.72	4.02	8.19	0.40
Training FLOPs / text token	1.17×10^{10}	2.61×10^{10}	4.75×10^{10}	–
	Shared by all three backbones			
Key–value heads	8			12
Head dimension	128			128
FFN activation	SwiGLU			GELU
Normalization	RMSNorm, also on queries and keys			RMSNorm
Position encoding	1D RoPE, base 10^6			learned 2D + 2D RoPE
Input	151,936-token vocabulary			14×14 patches
Tokens	4,096 per sequence			256 per image

Table 4: **Every recipe shares one backbone per scale, and the encoder adds a 27-layer, 0.4B vision Transformer.** The encoder's 12 attention heads of dimension 128 give an attention width of 1,536, wider than its residual width. Parameter counts exclude the linear projection from the encoder into the backbone. FLOPs per text token set the token-equivalent budgets (Section A.7).

A.2 Choosing the encoder configuration

We chose the from-scratch encoder's shape and size at matched lifetime compute, and every main comparison uses the resulting 27-layer, 0.4B encoder. The shape and size comparisons use the 1.7B backbone at 2.6 ZFLOPs, including the final learning-rate decay (Table 5).

The configurations in Table 5 vary the structure of the dedicated visual computation. Half-Depth, Wide, and Shallow + Wide are pre-fusion encoders that change the default encoder's depth, width, or both. The backbone-style encoder is also a pre-fusion encoder: it has its own patch embedding and eight Transformer blocks shaped like those of the 1.7B backbone, and it connects to the backbone through the linear projection of Equation (4). The vision-only prefix layers keep the same eight blocks but drop the separate encoder: a linear patch projection maps patches directly to the backbone width, as in the encoder-free recipe, and the eight blocks process only the vision tokens before they join the text tokens in the shared backbone. The Default, Shallow + Wide, backbone-style, and prefix variants use about the same vision-only parameter budget; the Half-Depth and Wide variants test smaller and larger budgets, respectively. The Size block scales the default encoder's depth and width together, from Small (0.05B) to Huge (0.92B).

The default encoder gives the best Avg. VQA (39.5), ahead of Wide (39.1), which uses more vision-only parameters. Avg. Text does not follow encoder size: among the four sizes, Small (50.9) and Huge (50.7) score highest and the default scores lowest (49.7). Among the shapes, the vision-only prefix layers give the highest Avg. Text (51.4) but the lowest Avg. VQA (34.3), and the backbone-style encoder reaches 38.2 Avg. VQA.

At 4B. To test whether the size comparison transfers to a larger backbone, we train the Base, Large, and Huge encoders with the same from-scratch recipe on the 4B backbone (Table 6). At a constant learning rate, the default Large encoder again gives the best Avg. VQA (42.8) but the lowest Avg. Text (53.2), and Base gives the best Avg. Text (54.8). After the final decay, the same ordering holds: Large gives the best Avg. VQA (43.4) and Base the best Avg. Text (55.9), but Large's Avg. Text deficit to Base narrows from 1.5 to 0.4 points. At both scales, the default encoder gives the highest Avg. VQA but not the highest Avg. Text.

Architecture				Text (%) ↑					VQA (%) ↑				
Variant	Depth	Width	Vision-only params (B)	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
• Large (default)	27	1,024	0.40	49.7	63.9	45.0	36.7	53.4	**39.5**	**52.9**	**35.0**	25.0	45.0
Shape													
Half-Depth	14	1,024	0.21	49.6	63.6	45.8	37.5	51.5	37.8	51.0	32.9	21.2	**46.1**
Wide	27	2,048	1.26	49.2	63.6	44.8	35.2	53.2	39.1	52.2	32.7	**25.8**	45.7
Shallow + Wide	9	2,048	0.42	49.9	64.1	44.8	36.3	54.2	38.0	52.5	33.7	21.4	44.3
Backbone-style	8	2,048	0.40	50.1	63.6	44.1	37.7	54.9	38.2	52.7	33.9	23.8	42.5
Vision-only prefix	8	2,048	0.40	**51.4**	64.4	45.0	**40.3**	55.8	34.3	48.7	31.4	13.7	43.6
Size													
Small	12	512	0.05	50.9	62.9	45.8	39.1	**55.9**	36.0	50.3	32.8	19.9	41.0
Base	18	768	0.15	49.9	64.8	**46.1**	35.4	53.2	37.5	50.8	32.8	21.3	45.0
Huge	40	1,280	0.92	50.7	**65.1**	44.8	37.8	55.1	37.7	51.1	33.0	23.8	42.9

Table 5: **The default 27-layer, 0.4B encoder gives the best Avg. VQA among the tested shapes and sizes.** Every row uses the 1.7B backbone at 2.6 ZFLOPs of lifetime compute, including the final learning-rate decay. Vision-only parameters exclude the linear projection into the backbone; at matched lifetime compute, encoders with more parameters process fewer tokens. The shaded row is the default encoder that every main comparison uses; bold marks the best value in each column.

Architecture		Text (%) ↑					VQA (%) ↑				
Variant	Vision-only params (B)	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
Constant learning rate											
Base	0.15	**54.8**	65.8	**48.6**	**46.8**	**57.9**	42.0	56.5	38.1	25.6	47.8
• Large (default)	0.40	53.2	65.9	46.6	42.6	**57.9**	**42.8**	**56.9**	**39.1**	**27.0**	**48.1**
Huge	0.92	54.1	**66.3**	47.7	44.6	**57.9**	41.0	56.7	37.8	24.4	45.2
After the final decay											
Base	0.15	**55.9**	**67.2**	49.0	**49.2**	58.1	42.9	56.5	38.2	27.3	**49.8**
• Large (default)	0.40	55.5	**67.2**	48.1	47.4	59.1	**43.4**	**58.2**	39.1	**28.1**	48.3
Huge	0.92	55.8	67.0	**49.3**	47.9	**59.2**	42.5	56.9	**39.2**	26.6	47.6

Table 6: **On the 4B backbone, the default Large encoder gives the best Avg. VQA, and Base the best Avg. Text.** The top block compares Base, Large, and Huge from-scratch encoders at a constant learning rate and matched lifetime compute (5.2 ZFLOPs). The bottom block continues all three encoders through a final learning-rate decay to 5.8 ZFLOPs, which raises Avg. Text by 1.1, 2.2, and 1.7 points and Avg. VQA by 0.9, 0.7, and 1.5 points for Base, Large, and Huge, respectively. The shaded row is the default encoder; bold marks the best value within each block.

A.3 Optimization and learning-rate schedule

Table 7 lists the optimization settings for multimodal training and instruction tuning. We train with Muon (Jordan et al., 2024), following Kimi K3 (Kimi Team, 2026), and use AdamW for embeddings, output heads, normalization parameters, and biases. The peak learning rate scales with the inverse square root of the backbone parameter count, and training follows a warmup–stable–decay schedule (Hu et al., 2024; Wen et al., 2025; Hägele et al., 2024). Checkpoints taken before the final decay use a constant learning rate; the decay lowers the learning rate linearly to 5% of its peak over the last 10% of updates, so a run whose stable phase ends at compute C finishes at $C/0.9$ (for example, 11.7 and 13.0 ZFLOPs at 1.7B and 26.1 and 29.0 ZFLOPs at 4B; Table 10). Section B.1 describes instruction tuning, and Section C.1 gives the settings for encoder pretraining.

	Multimodal training	Instruction tuning
	1.7B, 4B, and 8B backbones	every checkpoint
Optimizer	Muon (weight matrices) + AdamW (others)	AdamW
Peak learning rate	3×10^{-4} (1.7B), 2×10^{-4} (4B), 1.4×10^{-4} (8B)	1×10^{-5}
Warmup updates	4,000 (linear)	1,000
Schedule	warmup–stable–decay; linear decay to 5% of peak over the final 10%	cosine to 5% of peak
Weight decay	0.1	0.1
Gradient-norm clip	1.0	5.0
Batch (sequences × length)	$1{,}024 \times 4{,}096$	$256 \times 4{,}096$
Trainable parameters	all, except a frozen pretrained encoder	
Data	text-only and image–text sequences, 70/30	Cambrian-7M, 3 epochs

Table 7: **All backbone sizes share one optimization recipe, with the peak learning rate scaled to backbone size.** Muon and AdamW use the same peak learning rate.

A.4 Training data and recaptioning

Text-only sequences come from Nemotron-CC (Su et al., 2025), and image–text sequences use MetaCLIP image–text pairs (Xu et al., 2024). Every training sequence has 4,096 tokens and packs several documents; attention is bidirectional among the vision tokens of each image, causal otherwise, and does not cross document boundaries. Multimodal training uses a 70/30 mixture of text and image–text sequences, which we select in Section A.5. Each image is processed at 224×224 resolution and yields 256 vision tokens (Section A.1). An image–text sequence packs about 12.8 MetaCLIP pairs, each one image and its caption, so vision tokens fill about 80% of an image–text sequence and, under the 70/30 mixture, about 24% of all training tokens. The mixture ablation (Section A.5) varies the ratio, the study of when image–text data enters training (Section C.7) uses a 90/10 mixture, and the richer patch embeddings of Section D.3 read higher-resolution images.

Recaptioning. Multimodal training uses captions regenerated once for the MetaCLIP images by Qwen2.5-VL-32B-Instruct (Bai et al., 2025a) and shared by every recipe, while encoder pretraining uses the original alt-text (Section 2). We use nucleus sampling with temperature 0.6 and top-p 0.95, generating at most 128 new tokens. The system prompt is:

```
You are an expert image captioner.  Write one specific,
information-dense caption (1–2 sentences).  Do not begin
with 'A', 'An', or 'The'.  Name the main subject first, then
key objects, attributes, actions, and scene context.  Avoid
hedging and meta phrases.
```

The user prompt is:

```
Generate a caption for this image.
```

A.5 Choosing the data mixture

We select the 70/30 mixture of text and image–text sequences by an ablation on the 1.7B backbone with a from-scratch vision encoder, consistent with controlled studies of data balance in early multimodal training (Han et al., 2026). At the same lifetime compute (2.34 ZFLOPs, at a constant learning rate), we compare it with 100/0, 90/10, and 50/50 mixtures (Table 8). The 70/30 mixture gives the best Avg. VQA (38.7), whereas 90/10 keeps Avg. Text closer to the text-only model (49.3 versus 50.1) and gives the lowest text perplexity (16.7).

The mixture also changes how much the instruction-tuned model relies on the image (Table 9). From 90/10 to 70/30, aligned Avg. VQA rises by 4.1 points, the question-only score falls, and the shuffled-image score barely changes, so the question-only gap grows from 9.9 to 15.7 points and the shuffled-image gap from 9.2 to 12.7 (Section B.3).

We use 70/30 for the main comparisons because the paper prioritizes visual understanding, at some cost in language ability (Table 8).

Data mixture	Text (%) ↑					VQA (%) ↑					Diagnostics
Text/image–text	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis	PPL_5-GM ↓
100/0	50.1	64.3	44.8	36.2	55.3	27.2*	32.8*	29.9*	3.6*	42.3*	17.6
90/10	**49.3**	62.8	44.3	**36.3**	**53.9**	34.6	48.3	31.7	16.9	41.5	16.7
70/30	48.0	62.6	**44.5**	31.7	53.2	**38.7**	**52.7**	**32.8**	23.5	**45.7**	17.1
50/50	47.7	**63.7**	44.2	31.4	51.5	37.2	49.7	31.8	**23.6**	43.6	17.9

Table 8: **The 70/30 mixture favors visual tasks, whereas 90/10 preserves more language ability.** Ratios give the balance of text and image–text training sequences. For 100/0, images are replaced with zeros during instruction tuning and evaluation, so its starred VQA entries are blank-image controls shown for context (Section B.3). Avg. Text gives equal weight to commonsense, science QA, broad knowledge, and reading comprehension, and Avg. VQA to the general, knowledge, OCR and chart, and vision-centric categories; the four columns after each average show its components. PPL_5-GM is the geometric mean of text perplexity over five held-out corpora (Section B.5). Bold marks the best image-trained mixture in each Text and VQA column; the shaded text-only reference row and PPL are not bolded.

Data mixture	Avg. VQA (%) ↑			Gap to control	
Text/image–text	Aligned	Question-only	Shuffled	Question-only	Shuffled
90/10	34.6	24.7	25.4	9.9	9.2
70/30	38.7	23.0	26.0	15.7	12.7
50/50	37.2	20.7	–	16.4	–

Table 9: **Reliance on the image grows with the image–text fraction, while aligned-image Avg. VQA peaks at 70/30.** Question-only evaluation removes the image from the instruction-tuned model's input; shuffled-image evaluation substitutes another image from the same benchmark, keeping the input layout fixed (Section B.3). Each gap subtracts a control score from the aligned score; the shuffled-image gap grows less from 90/10 to 70/30 than the question-only gap (3.5 versus 5.8 points). 50/50 has no shuffled-image evaluation.

A.6 Lifetime compute accounting

Lifetime compute (Equation (1)) sums the training compute of every stage that produces the final model: multimodal training for every recipe and, for pretrained encoders, encoder pretraining. Each term uses the same algorithmic count of forward- and backward-pass matrix multiplication and convolution FLOPs for the modules active in that stage. The count excludes activation recomputation, optimizer updates, communication, elementwise operations, and input processing. A frozen encoder receives no backward pass during multimodal training, so it is charged only its forward pass, one third of its training cost.

Charging pretrained checkpoints. A pretrained encoder is charged the full compute of its upstream training up to the checkpoint used, including MetaCLIP's text encoder. Because a MetaCLIP example costs less compute than a captioning example, the same upstream compute buys MetaCLIP about 12% more passes. The objective comparisons therefore take MetaCLIP checkpoints at the compute of 2, 4, and 8 captioning passes (1.54, 3.08, and 6.16 ZFLOPs; Section C.1), whereas the sweep that holds multimodal training fixed uses MetaCLIP's own 2-, 4-, 8-, and 13-pass checkpoints (1.4, 2.8, 5.5, and 9.0 ZFLOPs; Section C.3).

A.7 Converting compute to token-equivalent budgets

We report compute in ZFLOPs. Because recipes differ in cost per token, matched lifetime compute means that they process different numbers of tokens. For intuition about scale, we also express each budget in *token-equivalents*, the number of tokens that a text-only model of the same backbone size would train on with the same compute:

$$H_{\text{eq}} = \frac{C_{\text{lifetime}}}{c_{\text{LM}}}, \tag{6}$$

where c_{LM} is that text-only model's training FLOPs per token (Table 4). For example, a budget of 200 billion token-equivalents is 2.34 ZFLOPs at 1.7B, and Table 10 converts every budget used in this paper.

Backbone		Lifetime compute (ZFLOPs) at each token-equivalent budget							
Size	FLOPs / token ($\times 10^{10}$)	100B	200B	222.2B	400B	444.4B	888.9B	1T	1.11T
1.7B	1.17	1.17	2.34	2.60	4.69	5.21	10.41	11.72	13.02
4B	2.61	2.61	5.21	5.79	10.42	11.58	23.16	26.06	28.95
8B	4.75	4.75	9.50	10.56	19.01	21.12	42.24	47.52	52.80

Table 10: **The same token-equivalent budget costs more compute at larger backbones.** The 222.2B, 444.4B, 888.9B, and 1.11T budgets include the final learning-rate decay after 200B, 400B, 800B, and 1T (Section A.3).

Table 11 shows how architecture and data mixture change the number of updates at 2.34 ZFLOPs. Each update processes $1{,}024 \times 4{,}096$ tokens (Table 7), so the text-only model's 47.7K updates correspond to 200B tokens.

Configuration			Training length
Architecture	Text/image–text	Vision-only params (B)	Updates (K)
Text only	100/0	–	47.7
• Encoder	90/10	0.40	46.9
• Encoder	70/30	0.40	45.3
• Encoder	50/50	0.40	43.9
Half-Depth encoder	70/30	0.21	46.4
Wide encoder	70/30	1.26	41.2
▪ Encoder-free	70/30	–	47.7

Table 11: **Architecture and data mixture change the number of updates available at fixed compute.** Every configuration is a 1.7B model at the same lifetime compute (2.34 ZFLOPs, or 200B token-equivalents), taken at a constant learning rate. The shaded text-only row is the reference that defines the budget. Encoder rows use the from-scratch encoders of Table 5, and vision-only parameters exclude the linear projection into the backbone.

B Evaluation

This section defines the two main metrics of Section 2, Avg. VQA and Avg. Text, and the three secondary measures: ImageNet linear-probe accuracy and text and caption perplexity.

B.1 Instruction tuning

Following Cambrian-1, we use instruction tuning as an evaluation protocol: every multimodal-training checkpoint is instruction-tuned with one fixed recipe before its visual understanding is measured (Tong et al., 2024a). The recipe runs three epochs of Cambrian-7M (Tong et al., 2024a), and we evaluate the model after the third. Cambrian-7M is a curated instruction-tuning mixture spanning general visual tasks, OCR and charts, counting, math, science, code, and language-only data. Table 7 gives the optimizer settings and batch size. Parameters frozen during multimodal training, such as the frozen MetaCLIP encoder in Table 15, stay frozen, and all other parameters are updated.

B.2 Visual question-answering benchmarks

Evaluation follows the Cambrian-1 visual understanding suite (Tong et al., 2024a), using greedy decoding with at most 256 new tokens and at most 5,000 examples per task. Table 12 lists the 16 benchmarks in four categories (general, knowledge, OCR and chart, and vision-centric), with the split, metric, and number of examples for each. Each category mean averages its four benchmarks.

The suite contains 15 benchmark scores scaled to $[0, 100]$ and MME-Perception (MME-P). Let $\mathcal{T}$ denote the 15 tasks, with scores s_t, and let $m_k = 100(\text{accuracy} + \text{accuracy-plus}) \in [0, 200]$ be the native score of each of the ten MME-P subtasks. We compute

$$\text{MME-P} = \frac{1}{20}\sum_{k=1}^{10} m_k, \qquad \text{Avg. VQA} = \frac{\sum_{t\in\mathcal{T}} s_t + \text{MME-P}}{16}. \tag{7}$$

Avg. VQA is computed only when all 15 task scores and all ten MME-P subtask scores are available.

Benchmark	Reference	Split	Metric	Examples
General (Gen)				
MME-P	Fu et al. (2025)	test	MME score / 20	2,374
MMBench-EN	Liu et al. (2024a)	dev	Exact match	4,329
SEED-Bench	Li et al. (2023a)	test	Exact match	5,000
GQA	Hudson & Manning (2019)	test-dev	Exact match	5,000
Knowledge (Knw)				
ScienceQA (image)	Lu et al. (2022)	test	Exact match	2,017
MMMU	Yue et al. (2024)	validation	Exact match	900
MathVista	Lu et al. (2024)	testmini	Exact match	936
AI2D	Kembhavi et al. (2016)	test	Exact match	3,088
OCR and chart (OCR)				
ChartQA	Masry et al. (2022)	test	Relaxed accuracy	2,500
OCRBench	Liu et al. (2024b)	test	Exact match	1,000
TextVQA	Singh et al. (2019)	validation	VQA accuracy	5,000
DocVQA	Mathew et al. (2021)	validation	Exact match	5,000
Vision-centric (Vis)				
MMVP	Tong et al. (2024b)	full	Exact match	300
RealWorldQA	xAI (2024)	test	Exact match	765
CV-Bench-2D	Tong et al. (2024a)	test	Exact match	1,438
CV-Bench-3D	Tong et al. (2024a)	test	Exact match	1,200

Table 12: **Avg. VQA weights 16 benchmarks equally across four categories of four.** MME-P is normalized to $[0, 100]$ before averaging (Equation (7)). The Examples column counts the questions evaluated, at most 5,000 per benchmark in dataset order; the MME count includes the cognition subtasks, which MME-P excludes. Exact match compares the normalized parsed answer or option letter with the reference answers, relaxed accuracy tolerates small numeric errors, and VQA accuracy is the TextVQA consensus metric.

B.3 Image controls

Three controls separate visual understanding from what the question alone supports. The *blank-image* control replaces every image with zeros during both instruction tuning and evaluation while keeping the vision-token positions. We use it only for the text-only reference of Section 2, which has no trained visual pathway. The *question-only* control evaluates the standard instruction-tuned model without the image, measuring how much of each benchmark the question and language prior can answer (Goyal et al., 2017). Because it removes the vision tokens, it changes the input layout as well as the visual content. The *shuffled-image* control keeps the input layout fixed: it evaluates the standard instruction-tuned model with another question's image from the same benchmark, chosen by a seeded derangement (Gat et al., 2021). Section 3 measures reliance on the image by the Avg. VQA gain from the correct image over a shuffled one. A shuffled image can still admit the original answer, so this gain measures reliance on the specific image rather than the full visual effect. We report aligned-minus-control differences, where the aligned score uses the correct image.

B.4 Language benchmarks

Language evaluations use the multimodal-training checkpoint before instruction tuning. Table 13 lists the 12 benchmarks, their categories, and how each is scored. MMLU uses five fixed, subject-matched demonstrations and likelihood ranking over four choices (Hendrycks et al., 2021). Its score is the

unweighted macro-average over 57 subjects. The other tasks are zero-shot and rank the answer choices by log-likelihood, summed or averaged per completion token as listed in the table.

Task	Reference	Shots	Scoring
Commonsense reasoning (Com)			
COPA	Roemmele et al. (2011)	0	Summed log-likelihood
HellaSwag	Zellers et al. (2019)	0	Mean token log-likelihood
PIQA	Bisk et al. (2020)	0	Mean token log-likelihood
Social IQa	Sap et al. (2019)	0	Summed log-likelihood
WinoGrande	Sakaguchi et al. (2020)	0	Summed log-likelihood
Science question answering (Sci)			
ARC-Easy	Clark et al. (2018)	0	Mean token log-likelihood
ARC-Challenge	Clark et al. (2018)	0	Mean token log-likelihood
OpenBookQA	Mihaylov et al. (2018)	0	Mean token log-likelihood
Broad knowledge (Knw)			
MMLU (57 subjects)	Hendrycks et al. (2021)	5	Likelihood ranking, macro-average
Reading comprehension (Rdg)			
BoolQ	Clark et al. (2019)	0	Summed log-likelihood
RACE-middle (1,436 test)	Lai et al. (2017)	0	Summed log-likelihood
RACE-high (3,498 test)	Lai et al. (2017)	0	Summed log-likelihood

Table 13: **Avg. Text weights four categories equally, however many tasks each holds.** Mean token log-likelihood ranks answer choices by their mean log-likelihood per completion token; summed log-likelihood uses the total. RACE-middle and RACE-high are combined by their test-question counts before entering the reading-comprehension category (Equation (8)).

Avg. Text gives equal weight to four categories: commonsense reasoning (Com), science question answering (Sci), broad knowledge (Knw), and reading comprehension (Rdg). Let Com average COPA, HellaSwag, PIQA, Social IQa, and WinoGrande; let A average ARC-Easy and ARC-Challenge; let Sci average A and OpenBookQA; and let Knw be the 57-subject MMLU macro-average. We combine the 1,436 RACE-middle and 3,498 RACE-high test questions by their question counts, then let Rdg average this combined RACE accuracy and BoolQ. All constituents keep the scoring rules in Table 13, and

$$\text{Avg. Text} = \frac{\mathrm{Com} + \mathrm{Sci} + \mathrm{Knw} + \mathrm{Rdg}}{4}. \tag{8}$$

We report the four category scores with the aggregate. As with Avg. VQA, Avg. Text is computed only when all of its constituents are available.

B.5 Text and caption perplexity

Perplexity measures how well a model predicts held-out text and image captions. For corpus d with N_d target tokens and per-token negative log-likelihood $\ell_{d,t}$, we compute

$$\mathrm{PPL}_d = \exp\left(\frac{\sum_t \ell_{d,t}}{N_d}\right). \tag{9}$$

PPL_5-GM is our equal-weight geometric mean over five held-out text corpora: DCLM, C4, Wikipedia, and two text collections that are not publicly available (Li et al., 2024; Raffel et al., 2020; Tong et al., 2026),

$$\mathrm{PPL}_{5\text{-GM}} = \exp\left(\frac{1}{5}\sum_d \log \mathrm{PPL}_d\right). \tag{10}$$

COCO caption perplexity scores the first of the five human reference captions for each of the 5,000 COCO validation images (Lin et al., 2014). Caption PPL_3-GM is the equal-weight geometric mean of caption perplexity over three held-out sets of 5,000 images each: the COCO set, MetaCLIP image–text pairs (Xu et al., 2024), and a licensed stock-photo collection. The MetaCLIP pairs use their original alt-text.

B.6 IMAGENET LINEAR PROBES

The paper uses three linear probes on ImageNet-1K (Deng et al., 2009), which differ in which features they read and how much training data they use.

Downstream probe of the multimodal model. This probe mean-pools the final normalized vision-token states of the shared backbone, with the vision encoder and backbone frozen. It trains an ℓ_2-regularized linear classifier on 50 training images per class: a fixed 10% holdout selects the regularization strength, and the classifier is then refit on all 50,000 images. We report top-1 and top-5 accuracy on the full 50,000-image validation split. It gives the ImageNet accuracy in the multimodal result tables and the linear-probe accuracy of Section 2.

Upstream probe of the pretrained encoders. The upstream probe in Table 14 evaluates a pretrained vision encoder before multimodal training. It also trains a linear classifier on 50 images per class, and the table uses it to compare the four pretraining objectives at the same upstream compute.

Full-data probe of the pretrained encoders. The full-data probe follows the DINOv2 linear evaluation (Oquab et al., 2024) on the frozen pretrained encoder before multimodal training. Linear classifiers trained for 10 epochs on the full ImageNet-1K training set read the class token or mean-pooled patch tokens from the last one or four encoder blocks at 13 learning rates, and we report the best top-1 accuracy on the 50,000-image validation set. This probe gives the upstream ImageNet accuracy in Figure 2 (right) and Section C.3. Because it trains on the full training set, its accuracies are not comparable with the two 50-image-per-class probes above.

C SEPARATE ENCODER PRETRAINING

This section details the pretrained-encoder comparisons of Section 3.

C.1 OBJECTIVES AND UPSTREAM TRAINING

We compare four upstream objectives while retaining the same vision encoder architecture (Section A.1). MetaCLIP (Xu et al., 2024) uses a symmetric image–text contrastive objective (Radford et al., 2021). If z_i and w_i are normalized image and text representations in a batch of size B, its loss is

$$\mathcal{L}_{\text{clip}} = -\frac{1}{2B}\sum_{i=1}^{B}\left[\log\frac{\exp(z_i^\top w_i/\tau)}{\sum_j \exp(z_i^\top w_j/\tau)} + \log\frac{\exp(w_i^\top z_i/\tau)}{\sum_j \exp(w_i^\top z_j/\tau)}\right]. \tag{11}$$

Here τ is the softmax temperature. Cap (Tschannen et al., 2023) uses a causal decoder during upstream pretraining to predict caption tokens from the encoder output $f_\phi(x)$ (Equation (4)),

$$\mathcal{L}_{\text{cap}} = -\mathbb{E}_{(x,y)}\left[\frac{1}{|y|}\sum_j \log p(y_j \mid f_\phi(x), y_{<j})\right]. \tag{12}$$

CapPa (Tschannen et al., 2023) combines this causal objective with fully masked parallel caption prediction. Writing $\mathcal{L}_{\text{par}}$ for the loss that predicts all caption positions in parallel from the image and a fully masked caption input, its expected loss is

$$\mathcal{L}_{\text{cappa}} = 0.25\,\mathcal{L}_{\text{cap}} + 0.75\,\mathcal{L}_{\text{par}}. \tag{13}$$

WebSSL (Fan et al., 2025) follows DINOv2 (Oquab et al., 2024) and combines class-token self-distillation and KoLeo regularization with iBOT masked-patch prediction,

$$\mathcal{L}_{\text{webssl}} = \lambda_{\text{DINO}}\mathcal{L}_{\text{DINO}} + \lambda_{\text{iBOT}}\mathcal{L}_{\text{iBOT}} + \lambda_{\text{KoLeo}}\mathcal{L}_{\text{KoLeo}}. \tag{14}$$

The text tower, caption decoder, and self-supervised prediction heads are discarded before multimodal training. We follow the original objective definitions, and MetaCLIP, Cap, and CapPa train on the original alt text of the MetaCLIP image–text pairs.

We compare the objectives at matched upstream compute. Each method uses the first checkpoint that reaches the compute of two passes of the Cap objective over the image–text pairs (1.54 ZFLOPs).

For Cap and CapPa, this is two billion image–caption pairs; MetaCLIP and WebSSL reach the same compute after a different number of examples. The objective comparisons in Sections C.2 and C.4 use these checkpoints; Section A.6 explains why the sweep of Section C.3 uses different MetaCLIP checkpoints.

Table 14 summarizes the four objectives and probes each upstream encoder at this matched compute before multimodal training. The objectives span 61.5% (WebSSL) to 73.0% (MetaCLIP) top-1 accuracy, yet after multimodal training they reach nearly the same Avg. VQA (Section C.2), so upstream recognition accuracy does not predict downstream visual understanding.

Upstream objective		**Frozen ImageNet (%) ↑**	
Objective	Supervision	Top-1	Top-5
MetaCLIP	Image–text pairs	73.0	93.1
WebSSL	Images	61.5	86.1
Cap	Image–text pairs	72.0	91.6
CapPa	Image–text pairs	69.6	90.2

Table 14: **Upstream objectives produce different frozen ImageNet representations at matched training compute.** Every objective trains the same vision encoder architecture for 1.54 ZFLOPs. We report the ImageNet-1K validation accuracy (Deng et al., 2009) of a linear classifier trained on 50 images per class. This probe is not comparable with the full-data probe of Table 17: MetaCLIP reaches 73.0% here at 1.54 ZFLOPs but 80.4% there at 1.38 ZFLOPs.

C.2 Objective comparison at 10.4 ZFLOPs

At 1.7B, Table 15 compares the from-scratch encoder with encoders pretrained on each of the four objectives and updated during multimodal training, and with a frozen MetaCLIP encoder. The updated pretrained rows are the 15% allocations in Figure 2 (left), and the from-scratch encoder spends no compute on pretraining. All updated encoders fall within 0.7 Avg. VQA points and 0.4 Avg. Text points of one another. No objective dominates: contrastive MetaCLIP gives the highest Avg. VQA, self-supervised WebSSL the best vision-centric score, captioning with CapPa the best VQA knowledge score, and the from-scratch encoder the best general and OCR and chart scores. The pretrained priors help most on vision-centric questions, where the from-scratch encoder trails. Freezing MetaCLIP preserves the best Avg. Text but costs 2.4 Avg. VQA points, so an encoder that adapts during multimodal training is the stronger choice.

Encoder recipe	**Text (%) ↑**					**VQA (%) ↑**				
Variant	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
Scratch encoder	54.3	66.1	47.6	47.3	56.3	42.2	**57.6**	37.6	**28.5**	45.2
MetaCLIP	54.7	65.9	**48.3**	48.0	56.5	**42.9**	57.1	38.2	27.0	49.4
MetaCLIP (frozen)	**55.1**	66.4	48.0	47.5	**58.7**	40.5	54.0	38.2	23.9	45.9
WebSSL	54.6	66.1	46.5	**48.5**	57.3	42.5	54.0	38.1	27.1	**50.6**
Cap	54.4	66.1	47.7	46.2	57.5	42.4	55.9	38.7	26.7	48.2
CapPa	54.3	**66.5**	47.5	46.3	57.0	42.7	56.8	**39.5**	27.5	47.2

Table 15: **Pretrained and from-scratch encoders perform similarly at 1.7B.** Pretrained encoders use 1.54 ZFLOPs of upstream compute, and every model is trained to 10.4 ZFLOPs of lifetime compute, including a final learning-rate decay. No objective leads every category; freezing MetaCLIP gives the best Avg. Text but the lowest Avg. VQA. Bold marks the best value in each column, including the shaded scratch-encoder reference row.

Pretraining share. Table 16 lists the values behind every point in the left pair of Figure 2, where encoder pretraining takes 15%, 30%, or 60% of the same 10.4-ZFLOP lifetime budget. Every objective loses as its share grows: Avg. VQA falls from 42.4–42.9 at a 15% share to 38.4–40.3 at 60%, and Avg. Text from 54.3–54.7 to 51.1–52.2. WebSSL degrades least, and at a 60% share every objective trails the from-scratch encoder.

Encoder	Text (%) ↑			VQA (%) ↑		
Upstream share (%)	15	30	60	15	30	60
Scratch encoder		54.3			42.2	
MetaCLIP	**54.7**	**53.5**	**52.2**	**42.9**	42.1	38.9
WebSSL	54.6	**53.5**	**52.2**	42.5	**42.2**	**40.3**
Cap	54.4	53.0	51.5	42.4	41.6	38.4
CapPa	54.3	53.2	51.1	42.7	41.1	39.3

Table 16: **A larger pretraining share lowers Avg. Text and Avg. VQA for every objective.** Encoder pretraining takes 15%, 30%, or 60% (1.54, 3.08, or 6.16 ZFLOPs) of a 10.4-ZFLOP lifetime budget at 1.7B, and multimodal training takes the rest, including the final learning-rate decay. The shaded scratch encoder spends no compute on pretraining, so one value serves as the reference at every share. Bold marks the best objective at each share.

Training trajectories. Figure 7 follows the from-scratch, updated MetaCLIP, and frozen MetaCLIP encoders through training at the 15% share. Every checkpoint before the last precedes the final learning-rate decay (Section A.3), so only the final points match Table 15.

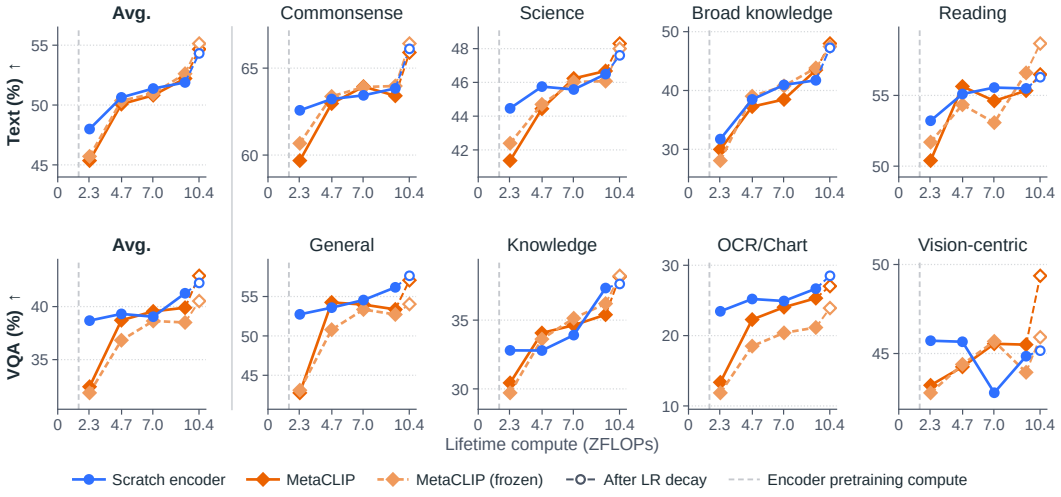

Figure 7: **Joint training keeps pace with a pretrained encoder whose upstream compute is charged.** Curves start after the compute spent on encoder pretraining (gray dashed line). The from-scratch encoder leads updated MetaCLIP on Avg. VQA at most checkpoints through 9.4 ZFLOPs; after the final learning-rate decay, updated MetaCLIP leads by 0.7 points and frozen MetaCLIP gives the best Avg. Text. Models are trained to 10.4 ZFLOPs or 888.9B-equivalent tokens. Filled markers are checkpoints before the final learning-rate decay; open markers show results after it.

C.3 Upstream pretraining at fixed multimodal compute

The right pair of Figure 2 plots a sweep that holds multimodal training fixed and does not charge encoder pretraining, unlike the lifetime-budget comparison in the left pair. Every model starts from a MetaCLIP checkpoint taken after 2, 4, 8, or 13 passes over the image–text pairs (1.4, 2.8, 5.5, or 9.0 ZFLOPs; Section A.6) and then receives the same multimodal training (2.7 ZFLOPs at 1.7B, 70/30 mixture, final learning-rate decay). Longer pretraining raises the pretrained encoder's full-data ImageNet linear-probe accuracy (Section B.6) but does not improve Avg. VQA after multimodal training; the shortest pretraining gives the best score in every VQA category (Table 17). From the shortest to the longest pretraining, ImageNet accuracy rises from 80.4% to 84.4% while Avg. VQA falls from 38.7 to 36.9.

Compute (ZFLOPs)		Text (%) ↑					VQA (%) ↑					Diagnostics
Upstream	Lifetime	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis	ImageNet ↑
1.4	4.1	50.5	63.9	45.2	**39.9**	52.9	**38.7**	**52.4**	**32.9**	**23.5**	**45.8**	80.4
2.8	5.5	**51.0**	**64.7**	**45.4**	38.6	55.4	37.3	49.5	31.2	23.2	45.1	80.6
5.5	8.3	**51.0**	64.5	45.3	39.2	54.9	37.7	50.2	32.7	22.3	45.5	82.0
9.0	11.7	50.7	64.2	**45.4**	37.9	**55.5**	36.9	49.5	31.9	22.3	43.9	84.4

Table 17: **At fixed multimodal compute, longer encoder pretraining improves ImageNet accuracy but not visual understanding.** Upstream is the MetaCLIP pretraining compute; lifetime adds 2.7 ZFLOPs of multimodal training. ImageNet is the full-data linear-probe accuracy of the pretrained encoder before multimodal training.

C.4 Objective comparison at a shorter budget

We repeat the objective comparison of Section C.2 but stop at 4.7 ZFLOPs with a constant learning rate, before the final decay. This gives a second budget at which pretraining takes about a third of lifetime compute, and the same runs provide the checkpoints for Section C.5. Each pretrained encoder spends 1.54 ZFLOPs upstream, which leaves it 60.8K multimodal updates against 90.6K for the from-scratch encoder and 95.4K for the encoder-free model. At this budget the from-scratch encoder scores highest on Avg. VQA (Table 18), 0.6 points above updated MetaCLIP and 0.7 above WebSSL, while Cap, CapPa, and the frozen MetaCLIP encoder trail by 1.6 to 2.5 points. The ordering differs from that at 10.4 ZFLOPs (Table 15), where pretraining takes 15% of the budget, and fits the pretraining-share trend: a larger share lowers Avg. VQA for every objective (Table 16).

Setup	Text (%) ↑					VQA (%) ↑					Diagnostics	
Vision setup	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis	COCO caption PPL ↓	PPL_5-GM ↓
Text-only reference	52.4	64.2	45.0	43.6	56.7	28.1*	–	–	–	–	–	16.4
Scratch encoder	**50.6**	63.2	**45.8**	38.5	55.1	**39.3**	53.6	32.8	**25.2**	**45.7**	32.0	16.3
MetaCLIP	50.1	63.0	44.4	37.3	**55.6**	38.7	**54.3**	**34.1**	22.3	44.3	35.0	16.8
MetaCLIP (frozen)	50.4	63.4	44.7	**39.0**	54.3	36.8	50.8	33.6	18.5	44.4	33.0	16.7
WebSSL	49.7	63.5	43.7	36.7	55.0	38.6	54.2	33.6	23.5	42.9	31.8	16.7
Cap	49.4	63.6	45.3	34.3	54.2	37.7	50.6	32.3	23.0	44.8	34.4	16.7
CapPa	50.1	**63.8**	44.5	37.1	55.1	37.3	51.8	32.2	21.5	43.6	35.7	16.7
Encoder-free	49.8	**63.8**	44.2	37.9	53.5	35.4	48.6	32.9	16.5	43.6	34.0	16.4

Table 18: **At 4.7 ZFLOPs without a final decay, the from-scratch encoder leads Avg. VQA, 0.6 points ahead of updated MetaCLIP.** Columns give Avg. Text and Avg. VQA with their category means (Sections B.2 and B.4) and caption and text perplexity (Section B.5). Bold marks the best vision setup in each Text and VQA column at one-decimal precision; perplexity is not bolded. The shaded text-only reference is not bolded and has no VQA category means; * marks its blank-image Avg. VQA control.

C.5 Representation dynamics during multimodal training

We track Avg. VQA and two representation diagnostics in the runs of Section C.4, at checkpoints up to 4.7 ZFLOPs with a constant learning rate.

We compute both representation diagnostics from mean-pooled final-layer patch tokens for a fixed sample of $n = 512$ COCO validation images. Let $X \in \mathbb{R}^{n \times d}$ contain one feature vector per image at a given checkpoint, and let X_0 contain the corresponding features at the initialization of the same model. Writing $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, we center the features across images as $\widetilde{X} = HX$ and $\widetilde{X}_0 = HX_0$. We compute linear centered kernel alignment (CKA) (Kornblith et al., 2019) as

$$\mathrm{CKA}(X, X_0) = \frac{\left\|\widetilde{X}^\top \widetilde{X}_0\right\|_F^2}{\left\|\widetilde{X}^\top \widetilde{X}\right\|_F \left\|\widetilde{X}_0^\top \widetilde{X}_0\right\|_F}. \tag{15}$$

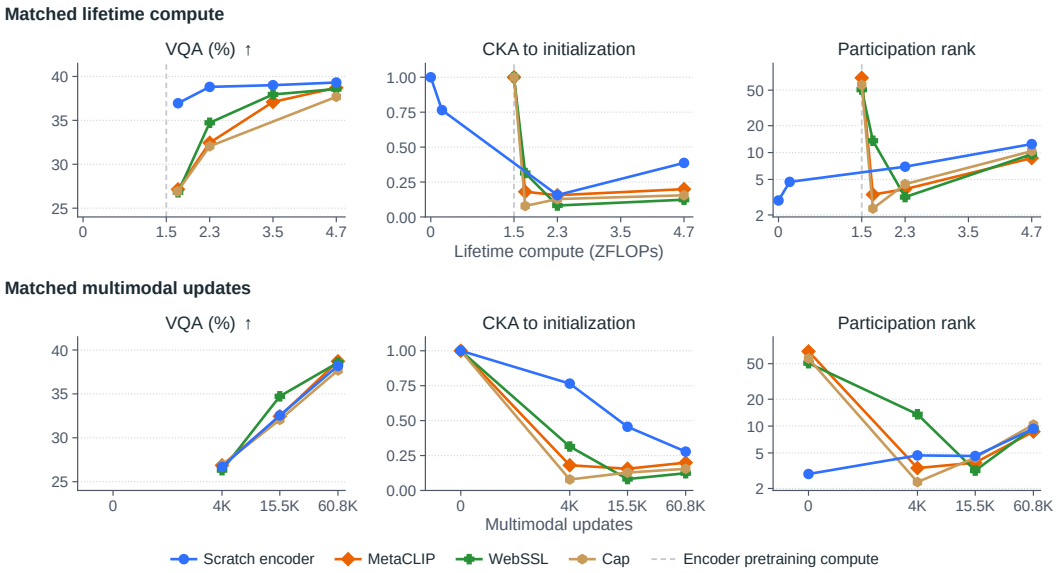

Figure 8: **Pretrained representations change sharply once multimodal training begins, without a consistent Avg. VQA head start.** Top: against lifetime compute, so pretrained curves start after their 1.54-ZFLOP upstream spend; MetaCLIP, WebSSL, and Cap end 0.6, 0.7, and 1.6 points below the scratch encoder, and Cap was not evaluated at 3.5 ZFLOPs. Bottom: the same models after equal numbers of multimodal updates. CKA compares each checkpoint with its own model's initialization (random for the scratch encoder). All checkpoints use a constant learning rate; the participation-rank axis is logarithmic.

Each reported value compares a checkpoint with its own model's initialization. CKA equals 1 when two centered representations match up to rotation and isotropic scaling.

For participation rank, we form the feature covariance $\Sigma_X = \frac{1}{n}\widetilde{X}^\top \widetilde{X}$ with eigenvalues $\{\lambda_i\}_{i=1}^d$ and compute

$$r_{\mathrm{PR}}(\Sigma_X) = \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2} = \frac{\mathrm{tr}(\Sigma_X)^2}{\mathrm{tr}(\Sigma_X^2)}. \tag{16}$$

The rank equals k when variance is distributed equally across k directions and approaches 1 as variation concentrates in a single direction. Both diagnostics summarize the global structure of the pooled representations; mean pooling discards spatial arrangement.

Matched lifetime compute. The top row of Figure 8 compares the pretrained encoders with the from-scratch encoder at matched lifetime compute. The pretrained encoders' CKA and participation rank fall sharply in the first few thousand updates. Their Avg. VQA rises as training continues, while participation rank partially recovers.

Matched multimodal updates. At fixed lifetime compute, pretrained recipes spend part of their budget upstream and therefore receive fewer multimodal updates than the from-scratch recipe. To separate initialization from the amount of multimodal training, the bottom row of Figure 8 compares the same encoders after the same number of multimodal updates. Pretraining gives no consistent Avg. VQA head start: after 4K updates all four encoders score within 0.6 points of one another, and after 60.8K updates the from-scratch encoder (38.2) lies between the pretrained encoders (37.7 to 38.7). The three pretrained encoders begin with participation ranks between 51 and 69, against 2.9 for the from-scratch encoder, yet all four encoders lie between 3.2 and 4.6 after 15.5K updates and between 8.7 and 10.4 after 60.8K updates. The diagnostics do not rank models by Avg. VQA: WebSSL leads Avg. VQA at 15.5K updates yet has the lowest CKA and rank among the pretrained models.

C.6 Results across backbone scale

Figure 9 breaks the from-scratch versus MetaCLIP comparison of Figure 3 down by VQA category, and Table 19 compares three upstream objectives with the from-scratch encoder at 4B. The objective comparison uses a shorter budget, 5.8 ZFLOPs of lifetime compute including the final decay, rather than the 11.6 ZFLOPs of Figure 3. Each objective uses the same 1.54 ZFLOPs of upstream compute as the objective comparisons at 1.7B (Section A.6). Among the objectives, WebSSL scores highest and MetaCLIP lowest on both Avg. Text and Avg. VQA, so the at-scale comparison of Section 3 is against CLIP-style pretraining, which was not the strongest objective here. At the same budget, the from-scratch encoder leads Avg. VQA (43.4 against 42.5 for WebSSL) and every VQA category, and also leads Avg. Text (55.5 against 54.4).

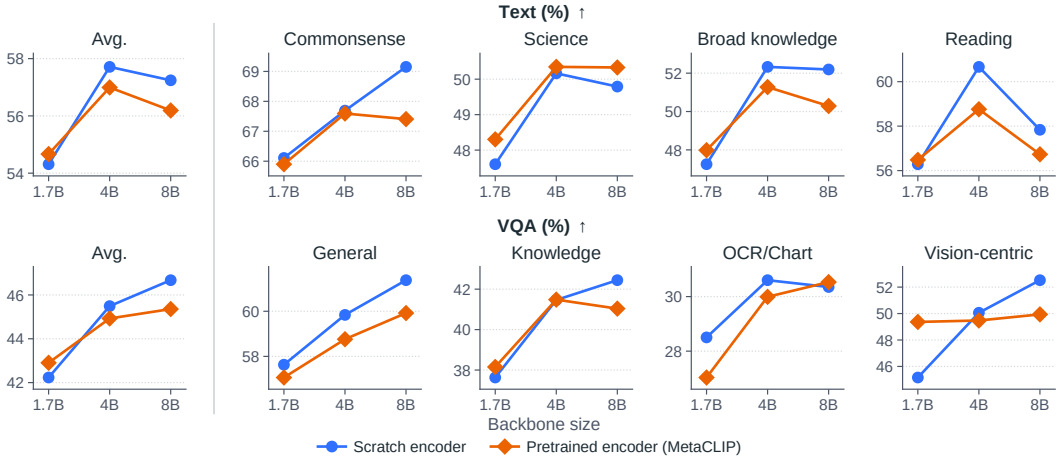

Figure 9: **At 4B and 8B, the from-scratch encoder matches or exceeds MetaCLIP in nearly every VQA category.** At 1.7B, MetaCLIP leads on vision-centric VQA. MetaCLIP uses 1.54 ZFLOPs of upstream compute (Section A.6); lifetime budgets, including upstream compute, are 10.4, 11.6, and 10.6 ZFLOPs at 1.7B, 4B, and 8B.

Encoder	Text (%) ↑					VQA (%) ↑					Diagnostics	
	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis	COCO caption PPL ↓	PPL_5-GM ↓
● Scratch encoder	**55.5**	**67.2**	48.1	**47.4**	**59.1**	**43.4**	**58.2**	**39.1**	**28.1**	**48.3**	27.0	13.7
◆ MetaCLIP	53.2	66.1	47.8	43.6	55.5	41.4	56.0	36.4	27.1	46.0	32.4	14.1
✚ WebSSL	54.4	66.5	48.4	45.4	57.4	42.5	58.0	37.6	27.6	46.8	32.0	14.1
● Cap	53.7	66.1	**48.5**	44.6	55.8	41.8	56.2	38.8	25.6	46.5	33.0	14.1

Table 19: **At 4B, the from-scratch encoder leads both Avg. Text and Avg. VQA; among the upstream objectives, WebSSL scores highest on both averages.** All rows are trained to 5.8 ZFLOPs of lifetime compute, including the final decay, of which each pretrained encoder spends 1.54 ZFLOPs upstream. The from-scratch encoder also leads every VQA category and every text category except science, and it has the lowest COCO caption perplexity and PPL_5-GM. Among the objectives, WebSSL also has the lowest COCO caption perplexity, and PPL_5-GM is tied. Bold marks the best value in each text and VQA column.

Table 20 gives the per-benchmark results behind Figure 9. The from-scratch encoder is ahead of MetaCLIP on 7 of the 16 benchmarks at 1.7B and on 10 at both 4B and 8B.

Benchmark	VQA (%) ↑					
	1.7B		4B		8B	
	Scratch encoder	MetaCLIP	Scratch encoder	MetaCLIP	Scratch encoder	MetaCLIP
Avg. VQA	42.2	**42.9**	**45.5**	44.9	**46.7**	45.4
General						
Category mean	**57.6**	57.1	**59.8**	58.8	**61.4**	59.9
MMBench-EN	61.8	**62.7**	**65.9**	63.6	66.5	**66.7**
SEED-Bench	53.7	**55.8**	**60.1**	59.5	**62.1**	60.8
GQA	**50.0**	49.8	50.0	**51.3**	51.4	51.4
MME-P	**65.0**	60.0	**63.3**	60.6	**65.6**	60.8
Knowledge						
Category mean	37.6	**38.2**	41.5	41.5	**42.4**	41.0
ScienceQA	51.5	**52.6**	57.7	**58.4**	**61.2**	57.3
MMMU	28.4	**31.1**	**29.9**	29.2	**30.1**	27.7
MathVista	**28.4**	25.3	**29.2**	28.7	27.9	**30.2**
AI2D	42.2	**43.6**	49.1	**49.6**	**50.6**	48.9
OCR and chart						
Category mean	**28.5**	27.0	**30.6**	30.0	30.4	**30.5**
ChartQA	**37.6**	35.9	**41.9**	39.8	42.4	**42.9**
OCRBench	**19.2**	16.3	20.0	**20.2**	19.4	**19.8**
TextVQA	**42.1**	41.2	42.2	**43.1**	**43.1**	42.7
DocVQA	**15.1**	14.7	**18.3**	16.8	16.5	**16.7**
Vision-centric						
Category mean	45.2	**49.4**	**50.1**	49.5	**52.5**	49.9
MMVP	52.0	**60.7**	**61.7**	58.3	**60.0**	55.0
RealWorldQA	41.4	**42.2**	**47.6**	46.9	**46.0**	43.5
CV-Bench-2D	41.0	**43.3**	43.3	**48.5**	**50.0**	47.2
CV-Bench-3D	46.2	**51.2**	**47.7**	44.2	**54.1**	54.0

Table 20: **At 4B and 8B, the from-scratch encoder is ahead of MetaCLIP on 10 of the 16 VQA benchmarks.** Per-benchmark results for Figures 3 and 9, at the same budgets and with the final learning-rate decay; MetaCLIP is updated during multimodal training. Each italicized group begins with its category mean; Avg. VQA weights the 16 benchmarks equally (Section B.2). Bold marks the higher value within each backbone size.

C.7 When image–text data enters training

Two studies vary when image–text training starts and how quickly the mixture changes once it starts.

Schedules. Both studies allocate 10% of lifetime training examples to image–text data, a 90/10 mixture, whereas the main architecture results use the constant 70/30 mixture (Section A.5). We compare schedules that begin image–text training at different points while keeping the total number of image–text examples and lifetime compute fixed (Figure 10). The schedule that starts immediately uses 10% image–text training throughout; its linear counterpart ramps to about 20% to match the same total. Later schedules continue the language model and optimizer, add a randomly initialized vision encoder and projector, and warm up the new parameters for 4,000 updates.

Training start. Starting image–text training at the first update gives the strongest visual understanding. After the final learning-rate decay (2.6 ZFLOPs), starting at 25%, 50%, or 75% of training lowers Avg. VQA from 36.7 to 31.2–32.3, with the largest drops on general and OCR questions, while Avg. Text stays within 50.8–52.0 (Table 21). Table 22 adds linear ramps, evaluated with the abrupt schedules at an earlier point before that decay (2.34 ZFLOPs), so its values are not comparable with Table 21. There, a linear ramp recovers part of the Avg. VQA lost to a later start, most clearly for the earlier starts. Earlier image–text training also strengthens visual grounding: the Avg. VQA gain from the correct image over a shuffled one (the shuffled-image gap; Section B.3) falls as the start moves later under both schedule shapes (Figure 11, left). Frozen ImageNet accuracy of the encoder

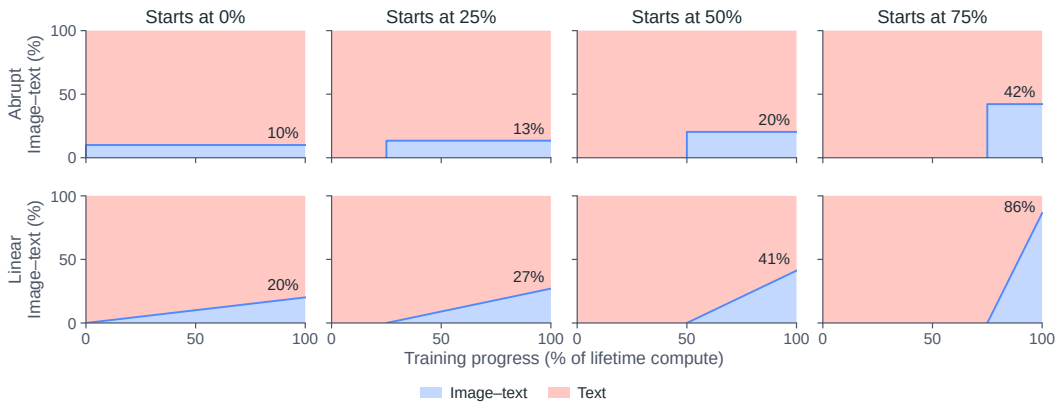

Figure 10: **Later starts concentrate the same image–text data into fewer updates.** Columns give the point in training at which image–text data is introduced; rows compare an abrupt switch with a linear ramp. Every schedule trains the same model for the same lifetime compute and sees the same total number of image–text examples, so the blue areas are approximately equal. Labels give each schedule's final image–text fraction, which rises from 10% to 42% for the abrupt switch and from 20% to 86% for the linear ramp as the start moves later.

is highest when image–text training starts at the first update but does not decline steadily across later starts (Figure 11, right).

Image–text start	**Text (%) ↑**					**VQA (%) ↑**				
	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
From first update	50.8	**64.6**	46.2	38.0	54.4	**36.7**	**51.2**	32.3	**19.2**	44.1
25% start	**52.0**	**64.6**	**46.5**	**41.9**	**55.0**	31.8	41.1	29.8	11.2	**45.0**
50% start	51.0	64.2	46.3	40.2	53.1	32.3	42.4	**32.7**	10.1	43.8
75% start	50.8	**64.6**	46.0	39.2	53.4	31.2	40.2	31.5	10.8	42.1

Table 21: **Starting image–text training from the first update gives the strongest visual understanding.** Every schedule sees the same total number of image–text examples (a 90/10 text to image–text mixture overall) and the same lifetime compute (2.6 ZFLOPs, including a final learning-rate decay). Figure 10 defines the schedules, and Table 22 and Figure 11 add linear ramps and the shuffled-image gap.

Image–text start	**VQA (%) ↑**					**Diagnostics**	
	Avg	Gen	Knw	OCR	Vis	COCO caption PPL ↓	PPL_5-GM ↓
Linear ramp							
From first update	**36.0**	**50.0**	**32.7**	16.4	**44.7**	35.5	16.8
25% start	34.1	47.7	29.9	16.3	42.3	34.3	16.8
50% start	32.5	43.1	30.2	12.8	43.7	35.6	17.2
75% start	31.1	41.6	29.6	11.4	41.9	34.5	19.5
Abrupt switch							
From first update	34.6	48.3	31.7	**16.9**	41.5	33.9	16.7
25% start	30.2	40.8	28.8	10.2	41.0	40.7	16.6
50% start	30.0	38.7	30.1	9.0	42.3	39.8	16.8
75% start	30.5	39.4	29.5	9.0	43.9	35.3	17.3

Table 22: **Linear ramps mitigate, but do not remove, the cost of delaying image–text training.** Linear and abrupt rows are evaluated at the same point, before the final learning-rate decay (2.34 ZFLOPs); the linear ramp leads by 3.9, 2.5, and 0.6 Avg. VQA points at the 25%, 50%, and 75% starts.

Transition duration. The transition-duration study asks how quickly the mixture should change once image–text training begins (Table 23); unlike the training-start study, it fixes the start point,

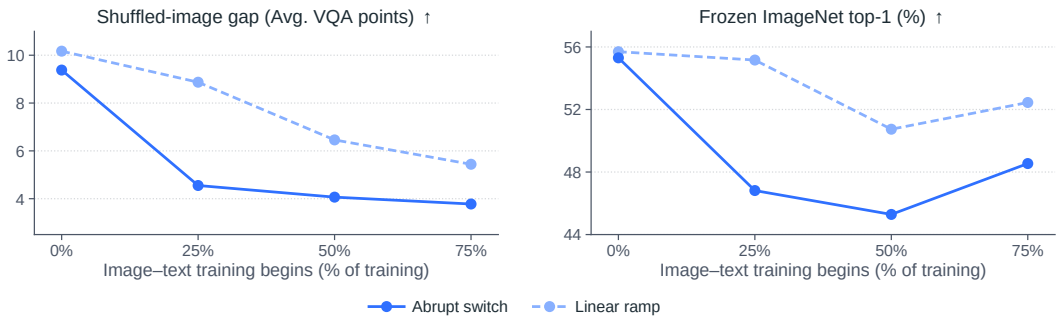

Figure 11: **Under both schedule shapes, starting image–text training later shrinks the gain from the correct image over a shuffled one.** Left: this shuffled-image gap in Avg. VQA falls by 4.7–5.6 points from the first-update start to the 75% start, and its paired 95% bootstrap intervals are narrower than ± 0.75 points. Right: frozen ImageNet top-1 of the encoder is highest for the first-update start but does not decline steadily across later starts. The schedules are those of Table 22, read at the same point before the final learning-rate decay (2.34 ZFLOPs).

and every schedule trains to 2.34 ZFLOPs of lifetime compute. Four schedules add a randomly initialized vision encoder and projector to the same text-only checkpoint halfway through training (100B token-equivalents), train on a 2% image–text floor from that point, and then ramp to a common final image–text fraction of 41% over 0, 1,000, 4,000, or 16,000 updates, centered on a shared transition midpoint about three quarters of the way through training. A control without the 2% floor sees no image–text data until shortly before that midpoint and then switches abruptly, concentrating the same data into fewer updates. The four schedules with the 2% floor span 1.8 Avg. VQA points (32.0–33.8), and the control scores below all of them (30.1), so a small early image–text phase matters more than the length of the ramp. Text perplexity is unaffected: DCLM perplexity is 17.5 for every schedule. Caption perplexity does not follow the VQA ordering: the control has the lowest COCO caption PPL (33.9, against 34.6–36.3 with the floor).

Training schedule		**VQA (%) ↑**				
Early phase	Ramp updates	Avg	Gen	Knw	OCR	Vis
2% floor	0	32.7	44.4	30.2	12.7	43.6
2% floor	1,000	**33.8**	**46.5**	31.1	12.7	**44.8**
2% floor	4,000	32.0	44.6	28.8	13.0	41.3
2% floor	16,000	33.5	**46.5**	**31.5**	**13.2**	42.9
No floor	0	30.1	38.6	29.7	10.1	41.9

Table 23: **A small early image–text phase matters more than the length of the ramp.** Every schedule sees the same total number of image–text examples and ends at 2.34 ZFLOPs. Rows with a 2% floor add the vision encoder halfway through training, train on 2% image–text data from 100B token-equivalents, and ramp to the common final mixture over the given number of updates around a transition midpoint about three quarters of the way through training. The shaded no-floor control sees no image–text data until shortly before that midpoint and then switches abruptly. Bold marks the best of the four floor schedules in each column.

D Dedicated visual capacity

This section gives category breakdowns and experimental details for Section 4.

D.1 Where the encoder's advantage comes from

Figures 12 and 13 break the left and right pairs of Figure 4 down by category. At 1.7B, the encoder's lead over training comes mostly from OCR and chart questions and, to a lesser extent, general questions (Figure 12). Avg. Text stays close, knowledge questions show a smaller gap that varies in sign, and vision-centric questions favor neither recipe consistently. Across backbone sizes, the OCR

and chart gap is large at 1.7B and 4B and narrows at 8B, where vision-centric questions show the largest remaining gap (Figure 13). Knowledge questions stay close at every size.

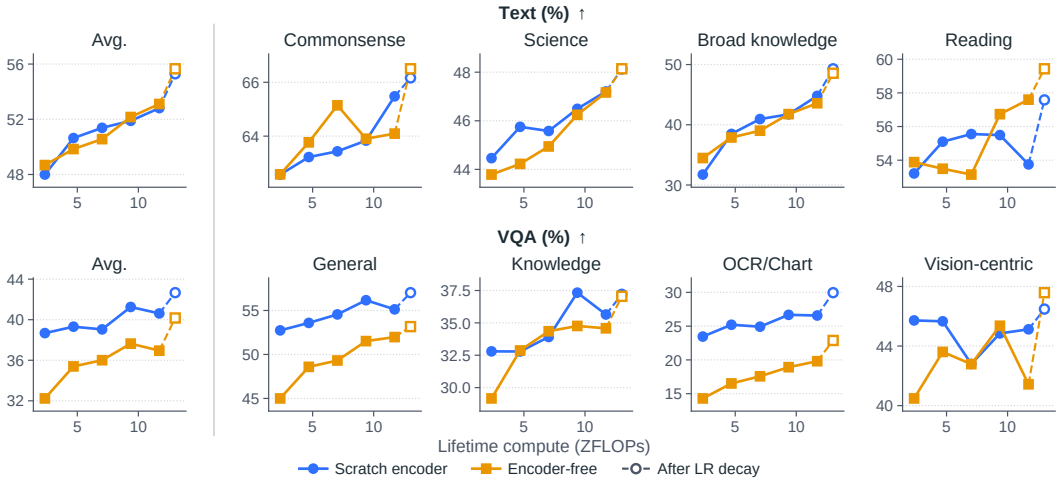

Figure 12: **At 1.7B, the encoder's lead over training is concentrated in OCR and chart questions.** Category breakdown of the 1.7B encoder and encoder-free trajectories. Filled markers are checkpoints before the final learning-rate decay; open markers show results after it, at 13.0 ZFLOPs.

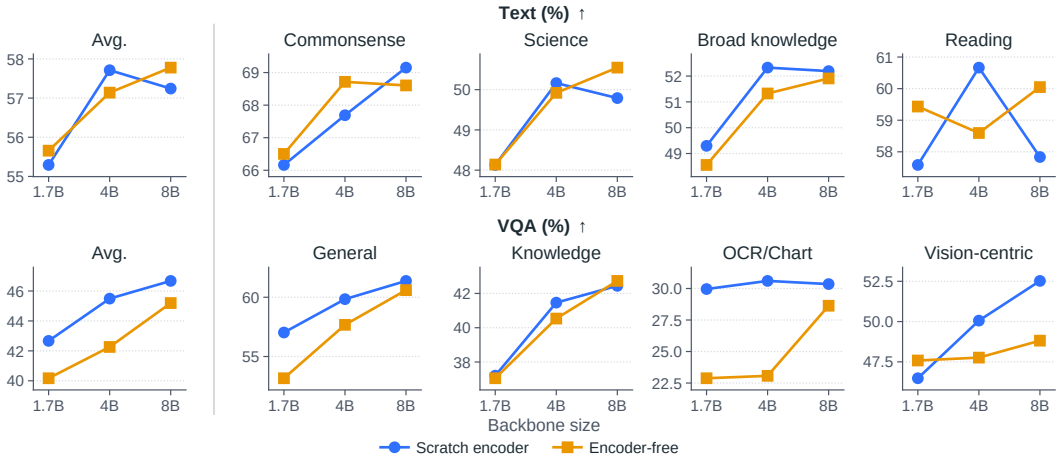

Figure 13: **The OCR and chart gap narrows at 8B, where vision-centric questions show the largest remaining gap.** Category breakdown of the encoder and encoder-free comparison across backbone sizes. All models include the final learning-rate decay, at 13.0, 11.6, and 10.6 ZFLOPs for 1.7B, 4B, and 8B.

D.2 Shared-capacity controls

Parameter-matched expansions. The parameter-matched shared expansions in Table 1 give the encoder-free backbone about 0.4B extra parameters. At 1.7B (28 layers of hidden size 2,048), the wider backbone has hidden size 2,304, the deeper backbone has 36 layers, and the wider and deeper backbone has 30 layers of hidden size 2,176, each with about 2.1B parameters. At 4B (36 layers of hidden size 2,560), the wider backbone has hidden size 2,688 and 4.4B parameters. Here, wider means a larger hidden size; the fixed-token study below instead widens only the FFNs and keeps the hidden size at 2,048.

Shared capacity at a fixed token count. Table 1 matches lifetime compute, so each larger shared backbone trains on fewer tokens than the encoder. We therefore also train 1.7B encoder-free backbones with wider FFNs or more layers on the encoder's token count, which gives each expansion more

compute than the encoder (Table 24). We evaluate these runs at a constant learning rate, with the encoder at 4.7 ZFLOPs and the expansions at 5.6 to 6.7 ZFLOPs, whereas Table 1 reports results after the final decay, at 5.2 ZFLOPs. The encoder therefore scores lower here than in Table 1, so we compare rows only within Table 24. At about the encoder's parameter count, shared capacity still trails: 495M of FFN width or 403M of depth leaves the backbone 1.9 Avg. VQA points behind the encoder. Larger expansions narrow the gap: at about 800M, width trails by 0.6 points and depth by 1.1, and 991M of width scores 0.6 points above the encoder. Every expansion scores higher than the encoder on Avg. Text, consistent with shared parameters also serving language.

Architecture						Text (%) ↑	VQA (%) ↑
Variant	Layers	FFN width	Vision-only params (M)	Shared params (M)	ZFLOPs	Avg	Avg
• Encoder	28	6,144	401	–	4.7	50.6	39.3
▪ Encoder-free	28	6,144	1	–	4.7	49.8	35.4
+ wider FFN	28	9,024	1	495	5.6	51.5	37.4
+ wider FFN	28	10,752	1	793	6.3	51.7	38.7
+ wider FFN	28	11,904	1	991	6.7	53.3	39.9
+ deeper	36	6,144	1	403	5.6	52.8	37.4
+ deeper	44	6,144	1	805	6.6	54.7	38.2

Table 24: **With the encoder's parameter count and training tokens, shared capacity still trails the encoder.** 1.7B backbones at a constant learning rate. The encoder and the shared expansions train on the same 380B tokens, so the expansions use more lifetime compute; the encoder-free baseline instead matches the encoder's compute. The encoder-free rows use the default patch embedding: a linear projection, RMSNorm, and a learned gate.

D.3 Richer patch embeddings

The default encoder-free patch embedding is a single linear projection followed by RMSNorm and a learned gate (Section A.1). We compare it with two richer patch embeddings inspired by Gemma 4 and Inkling (Gemma Team, 2026; Thinking Machines Lab, 2026): a normalized two-layer projection and a four-layer hierarchical MLP (Touvron et al., 2022). They are our implementations trained under our protocol, not faithful reproductions of those models. Table 25 specifies each variant. The normalized two-layer projection changes patch size, resolution, position embeddings, and normalization together. It normalizes each patch with LayerNorm, projects it linearly, adds learned factorized 2D position embeddings, and then applies LayerNorm, a parameter-free RMSNorm, and a bias-free linear map into the backbone. The hierarchical MLP has four bias-free linear layers that fold neighboring pixels into channels in stages, with RMSNorm and GELU between them, a final RMSNorm, and no position embedding. All variants use the same 1.7B backbone and 5.2 ZFLOPs of lifetime compute, and each produces a 16×16 grid of 256 vision tokens. The richer variants also read higher-resolution images (768×768 and 640×640 rather than 224×224), so this comparison confounds the patch network with input resolution. Both richer variants narrow the gap to the encoder, and the hierarchical MLP matches it on Avg. VQA at 1.7B (Table 26). As small vision-only networks applied before fusion, they are themselves a form of dedicated capacity. The largest gain is on OCR and chart questions, where the default patch embedding trails the encoder most, while Avg. Text changes little across patch embeddings.

Architecture				Input	
Variant	Vision-only layers	Vision-only params (M)	Position embedding	Patch size	Resolution
• Encoder	27 Transformer blocks	401	learned + 2D RoPE	14×14	224×224
▪ Encoder-free	1 linear	1	fixed 2D sine–cosine	14×14	224×224
Normalized two-layer projection	2 linear	18	learned factorized 2D	48×48	768×768
Hierarchical MLP	4 linear	44	none	40×40	640×640

Table 25: **The richer patch embeddings add vision-only layers and read larger patches of higher-resolution images.** Every variant feeds a 16×16 grid of 256 vision tokens to the same 1.7B backbone. The encoder and the default patch embedding are specified in Section A.1.

Architecture	Params		Text (%) ↑					VQA (%) ↑				
Variant	Vision-only (M)	Total (B)	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
• Encoder	401	2.1	52.5	65.0	47.2	41.5	56.4	40.2	54.6	34.5	26.9	44.9
▪ Encoder-free	1	1.7	52.6	65.3	46.6	42.9	55.6	36.9	50.8	34.3	17.5	**45.2**
Normalized two-layer projection	18	1.7	52.3	**65.9**	**46.7**	43.5	53.2	38.8	52.8	34.4	23.7	44.2
Hierarchical MLP	44	1.8	**52.8**	65.1	46.1	**44.3**	**55.7**	**40.2**	**54.2**	**35.9**	**26.7**	44.2

Table 26: **At 640×640 input, a 44M-parameter hierarchical MLP matches the encoder on Avg. VQA and recovers nearly all of its OCR and chart lead.** The encoder and the default patch embedding read 224×224 images (Table 25), so this comparison currently confounds the patch network with input resolution. Results at 5.2 ZFLOPs, after the final decay; category abbreviations follow Table 1. The encoder and encoder-free rows repeat the 1.7B rows of Table 1. Bold marks the best of the three encoder-free patch embeddings in each column; the shaded encoder row is the reference.

E In-fusion vision capacity

Section 5 adds in-fusion vision capacity: vision-only modules inside the backbone.

E.1 Implementation, sizing, and fixed-budget allocation

In each augmented layer, in-fusion FFN adds a vision-only SwiGLU branch of width F beside the shared FFN. For backbone width D and L_a augmented layers, this adds $P_{\text{FFN}} = 3L_aDF$ parameters. In-fusion FFN+QKVO additionally uses zero-initialized, full-rank, vision-only query, key, value, and output projections with the same output geometry as the shared ones. For H_Q query heads, H_{KV} key–value heads, and head dimension d_h, their parameter count is $P_{\text{QKVO}} = 2L_aDd_h(H_Q + H_{KV})$. Each vision-only module reads the same normalized input as its shared counterpart, and the vision-only attention projections add no normalization of their own; QK normalization, rotary embeddings, and attention are applied once to the combined projections, so there is no second attention stream.

Table 27 lists the resulting FFN widths and parameter counts. Every allocation augments the first eight layers ($L_a = 8$).

Full-rank vision-only attention projections in every layer would take 0.35B parameters at 1.7B, leaving room only for width-256 FFN branches in a model matched to the every-layer variant of Table 2. At 4B they would take 0.94B, more than the entire budget of about 0.4B. The depth ablation therefore uses in-fusion FFN, whose width can be resized continuously. Low-rank or partial-head attention projections could support a more balanced every-layer allocation, which we leave to future work.

Table 27 compares the two in-fusion allocations at a fixed added-parameter budget: in-fusion FFN spends the budget on the FFN branch, and in-fusion FFN+QKVO moves part of it into vision-only attention projections. At 1.7B, in-fusion FFN+QKVO improves 8 of the 16 VQA tasks, and CV-Bench-2D contributes 74% of its Avg. VQA gain. At 4B, the same contrast at 5.2 ZFLOPs is +0.6

[−0.1, 1.3]. The 1.7B in-fusion FFN model was trained separately from the Table 2 model, so its Avg. VQA differs slightly (37.0 versus 37.2). The 4B pair also appears in Table 29.

Architecture		Added params (M)		Text (%) ↑					VQA (%) ↑					
Variant	FFN width	FFN	QKVO	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis	Δ Avg [95% CI]
1.7B backbone														
▲ In-fusion FFN	8,192	402.7	0.0	**48.8**	**62.5**	**44.4**	35.9	**52.6**	37.0	**51.6**	32.6	**21.5**	42.2	–
▼ In-fusion FFN+QKVO	6,144	302.0	100.7	48.7	62.2	43.4	**37.1**	51.9	**38.0**	**51.6**	**34.5**	21.0	**45.0**	+1.1 [0.4, 1.7]
4B backbone														
▲ In-fusion FFN	6,528	401.1	0.0	55.8	66.5	49.0	49.5	58.1	43.8	57.7	39.6	26.5	**51.5**	–
▼ In-fusion FFN+QKVO	3,136	192.7	209.7	**56.7**	**67.3**	**49.5**	**49.8**	**60.0**	**44.8**	**60.2**	**40.2**	**28.4**	50.2	+1.0 [0.3, 1.6]

Table 27: **Vision-only attention projections give a small Avg. VQA gain at both scales.** Within each scale, the two allocations match added parameters; the attention share of the budget grows with backbone width. Both pairs are evaluated before the final decay, at 2.34 ZFLOPs (1.7B) and 10.4 ZFLOPs (4B), and the two models of each pair share seed, initialization, and data order. Δ Avg is in-fusion FFN+QKVO minus in-fusion FFN, with a paired 95% bootstrap interval over the questions of each VQA task (10,000 draws). Bold marks the better allocation within each scale.

E.2 Controls for the routing and depth ablation

Table 28 adds diagnostics and a matched-exposure comparison to the routing and depth ablation of Table 2. Shared FFN widening grows the shared FFN of layers 1–8 from width 6,144 to 14,336, so every token uses the added capacity. The separate FFN branch adds a width-8,192 SwiGLU branch to each of layers 1–8 and applies it to all tokens; in-fusion FFN is the same branch applied to vision tokens only. The depth variants move the same eight width-8,192 branches to eight layers spread across depth or to layers 21–28, or place width-2,304 branches in all 28 layers.

Architecture		Text (%) ↑	VQA (%) ↑	Diagnostics			
Variant	Used by	Avg	Avg	PPL_5-GM ↓	Caption PPL_3-GM ↓	MME-P ↑	ImageNet ↑
Encoder-free	–	48.7	32.2	17.11	54.83	50.35	53.01
Encoder	Vision	48.0	38.7	17.12	46.79	–	–
Routing (layers 1–8)							
Shared FFN widening	All	47.5	33.3	16.77	48.58	56.54	46.04
Separate FFN branch	All	**49.4**	36.5	16.74	46.78	54.71	53.05
In-fusion FFN	Vision	49.2	**37.2**	17.08	49.16	55.60	59.04
Depth of the in-fusion FFN branch							
Every layer (narrow)	Vision	49.0	32.6	17.05	54.14	47.00	63.89
Spread (8 of 28)	Vision	48.4	33.7	17.07	54.20	48.01	58.18
Layers 21–28	Vision	47.8	31.3	17.10	54.88	41.29	56.88
Routing at matched exposure (layers 1–8)							
Shared FFN widening	All	**49.6**	33.8	16.57	52.18	52.35	46.72
Separate FFN branch	All	49.5	36.2	16.60	49.92	47.88	54.46
In-fusion FFN	Vision	49.2	**37.2**	17.08	49.16	55.60	59.04

Table 28: **Routing the FFN branch to vision tokens gives a small Avg. VQA gain and a larger frozen-ImageNet gain.** Each design adds about 0.4B parameters to the 1.7B encoder-free backbone; all rows are checkpoints before the final learning-rate decay, at 2.34 ZFLOPs except the all-token matched-exposure rows (about 2.7 ZFLOPs). Diagnostics are defined in Sections B.5 and B.6; MME-P is MME-Perception. Bold marks the best Avg. Text and Avg. VQA within the routing and depth rows and within the matched-exposure rows; dashes mark values not evaluated. The runs share the training recipe but not the example order.

Routing at matched exposure and matched compute. In-fusion FFN and the separate FFN branch have the same parameterization and differ only in which tokens use the branch. Because the all-token designs also run their added capacity on text tokens, they cost more compute per training token, so at matched lifetime compute they train on fewer tokens than in-fusion FFN. The matched-exposure rows

instead train them on as many tokens as in-fusion FFN. Under both comparisons, routing the branch to vision tokens gives a small Avg. VQA gain (37.2 versus 36.5 at matched compute and 37.2 versus 36.2 at matched exposure) and a larger frozen-ImageNet gain (6.0 and 4.6 points), while all-token routing keeps a slight edge on Avg. Text and text perplexity. Whether routing adds benefit beyond the separate branch is therefore unclear at this budget (Section 5.2). With the same parameters, the separate FFN branch also beats shared FFN widening by 3.2 Avg. VQA points at matched compute. These short constant-learning-rate runs do not separate the effect of the branch's structure from that of its zero-initialized output projection.

Early versus late placement. The late variant moves the same eight width-8,192 branches from layers 1–8 to layers 21–28 and holds parameter count, routing, active compute, data recipe, and optimization fixed. Early placement leads by 5.8 Avg. VQA, 1.5 Avg. Text, and 2.2 ImageNet points, raises MME-P from 41.3 to 55.6, and lowers caption perplexity, whereas text perplexity differs by less than 0.3%.

E.3 Diagnostics and combined capacity at 10.4 ZFLOPs

Table 29 compares the 4B designs at 10.4 ZFLOPs, the constant-learning-rate checkpoint they share with a model that combines the encoder with in-fusion FFN, and adds likelihood and representation diagnostics. The main-text comparison instead uses the decayed endpoints (Section E.4). Among the single designs, in-fusion FFN+QKVO has the highest Avg. Text, Avg. VQA, MME-P, and frozen ImageNet top-1, the encoder has the lowest caption perplexity, and in-fusion FFN has the highest vision-centric VQA score.

Architecture		**Text (%) ↑**					**VQA (%) ↑**				
Variant	Placement	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
Single designs											
■ Encoder-free	–	55.5	66.7	49.2	50.0	56.1	41.1	54.9	39.4	22.7	47.6
● Encoder	Pre-fusion	56.3	67.7	48.7	49.1	59.6	43.8	59.3	41.7	26.2	48.3
▲ In-fusion FFN	In-fusion	55.8	66.5	49.0	49.5	58.1	43.8	57.7	39.6	26.5	**51.5**
▼ In-fusion FFN+QKVO	In-fusion	**56.7**	**67.3**	**49.5**	**49.8**	**60.0**	**44.8**	**60.2**	**40.2**	**28.4**	50.2
Combined pre- and in-fusion capacity (about 0.8B added)											
Encoder + In-fusion FFN	Both	55.1	67.8	48.1	49.3	55.3	45.2	–	–	–	–

Architecture		**Diagnostics**			
Variant	Added vision-only params (M)	PPL_5-GM ↓	Caption PPL_3-GM ↓	MME-P ↑	ImageNet ↑
■ Encoder-free	0.0	14.07	42.20	51.61	61.69
● Encoder	401.2	14.00	39.99	61.74	66.46
▲ In-fusion FFN	401.1	14.07	41.66	58.99	64.20
▼ In-fusion FFN+QKVO	402.4	14.04	41.16	63.26	66.66
Encoder + In-fusion FFN	802.3	13.99	39.45	63.93	68.85

Table 29: **Beyond the two main averages, the best 4B design at 10.4 ZFLOPs depends on the metric.** All rows are 4B models trained for 10.4 ZFLOPs at constant learning rate; the lower panel gives added vision-only parameters and diagnostics (Sections B.5 and B.6). The combined model processes about 2% fewer tokens than the other rows, in a different data order. Bold marks the better in-fusion design on Text and VQA; the shaded row is the encoder reference, and dashes mark values not evaluated.

Combining pre-fusion and in-fusion capacity. We also add the encoder to the in-fusion FFN allocation at 4B, for about 0.8B added vision-only parameters in total (combined block of Table 29). The single-design rows are the 10.4-ZFLOP checkpoints of the 4B runs in Figure 14. Relative to the better of the encoder and in-fusion FFN on each metric, the combined model improves Avg. VQA by 1.3 points, MME-P by 2.19 points, and frozen ImageNet top-1 by 2.39 points, and reduces caption PPL_3-GM by 1.35%. Against in-fusion FFN+QKVO, which adds half as many parameters, its Avg. VQA gain shrinks to 0.4 points, and its Avg. Text is 1.5 points lower.

E.4 LONGER TRAINING AND THE FINAL DECAY

Table 31 breaks down the decayed endpoints of Table 3 by Text and VQA category, next to the constant-learning-rate checkpoints from which each decay starts. Each decay continues a 1T-token-equivalent checkpoint to 1.11T token-equivalents, or 13.0 ZFLOPs at 1.7B and 29.0 ZFLOPs at 4B. After the decay, in-fusion FFN+QKVO leads the encoder in Avg. Text and Avg. VQA at both scales (43.9 versus 42.7 Avg. VQA at 1.7B and 50.0 versus 48.8 at 4B) and, at 4B, in every VQA category. In-fusion FFN ends even with the encoder at both scales (42.6 versus 42.7 and 48.5 versus 48.8 Avg. VQA).

Figure 14 shows the training trajectories behind these endpoints, and Table 30 lists Avg. VQA at every constant-learning-rate checkpoint. The encoder leads early, but in-fusion capacity catches up by 4.7 ZFLOPs at 1.7B and by 10.4 ZFLOPs at 4B. At 4B, in-fusion FFN+QKVO leads the encoder at every checkpoint from 10.4 ZFLOPs onward and averages 48.0 Avg. VQA over the last four checkpoints, against 44.7 for the encoder and 45.5 for in-fusion FFN. At 1.7B the corresponding means, 41.2, 41.4, and 42.0 for the encoder, in-fusion FFN, and in-fusion FFN+QKVO, lie within 0.8 points.

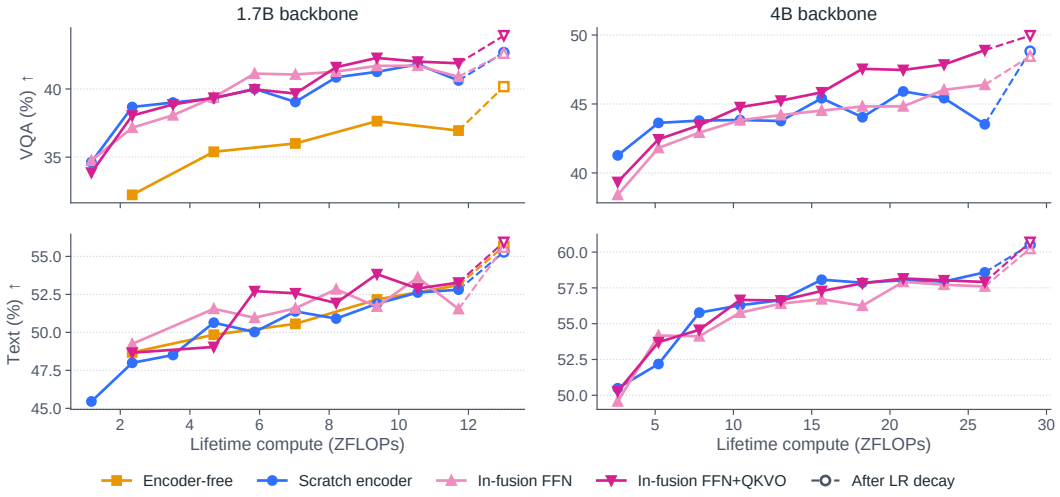

Figure 14: **The encoder leads early on Avg. VQA; in-fusion capacity catches up by 4.7 ZFLOPs at 1.7B and 10.4 ZFLOPs at 4B.** Open markers joined by dashed lines are the decayed endpoints of Table 3; all other points use a constant learning rate. At 4B, the lead of in-fusion FFN+QKVO over the encoder peaks at 5.4 Avg. VQA points before the decay and is 1.1 after it. The encoder-free model reaches this horizon only at 1.7B. Tables 30 and 31 list the values.

Schedule sensitivity. The decay can change the ordering of in-fusion FFN and the encoder. At 4B, in-fusion FFN leads the encoder by 46.4 to 43.5 Avg. VQA at the last constant-learning-rate checkpoint, where the encoder dips after 45.9 and 45.4 at the two previous checkpoints, but the two end even after the decay (48.5 and 48.8). The decay also narrows the 4B lead of in-fusion FFN+QKVO over the encoder from 5.4 to 1.1 points.

Architecture	VQA (%) ↑ at each checkpoint										Mean
Token-equiv.	100B	200B	300B	400B	500B	600B	700B	800B	900B	1T	Last 4
1.7B backbone											
ZFLOPs	*1.2*	*2.3*	*3.5*	*4.7*	*5.9*	*7.0*	*8.2*	*9.4*	*10.5*	*11.7*	*8.2–11.7*
● Encoder	34.6	38.7	39.0	39.3	40.0	39.0	40.9	41.3	41.8	40.6	41.2
▲ In-fusion FFN	**34.7**	37.2	38.1	**39.4**	**41.1**	**41.1**	41.3	41.7	41.7	40.9	41.4
▼ In-fusion FFN+QKVO	33.9	**38.0**	**38.9**	**39.4**	40.0	39.7	**41.6**	**42.3**	**42.0**	**41.9**	**42.0**
4B backbone											
ZFLOPs	*2.6*	*5.2*	*7.8*	*10.4*	*13.0*	*15.6*	*18.2*	*20.8*	*23.5*	*26.1*	*18.2–26.1*
● Encoder	41.3	43.6	43.8	43.8	43.8	45.4	44.0	45.9	45.4	43.5	44.7
▲ In-fusion FFN	38.4	41.8	42.9	43.8	44.2	44.5	44.8	44.8	46.0	46.4	45.5
▼ In-fusion FFN+QKVO	**39.3**	**42.4**	**43.5**	**44.8**	**45.2**	**45.8**	**47.5**	**47.5**	**47.9**	**48.9**	**48.0**

Table 30: **At constant learning rate, in-fusion FFN+QKVO pulls ahead of the encoder at 4B, whereas at 1.7B the three designs stay within about two points.** Avg. VQA behind Figure 14, at checkpoints spaced every 100B token-equivalents; the ZFLOPs rows give their lifetime compute. The last column averages the last four checkpoints. Bold marks the better in-fusion value at each checkpoint within each backbone; shaded rows are the encoder reference.

Architecture	Text (%) ↑					VQA (%) ↑				
Model	Avg	Com	Sci	Knw	Rdg	Avg	Gen	Knw	OCR	Vis
1.7B backbone, decayed, 13.0 ZFLOPs										
■ Encoder-free	55.7	66.5	48.1	48.6	59.4	40.2	53.2	37.1	22.9	47.6
● Encoder	55.3	66.2	48.1	49.3	57.6	42.7	57.0	37.2	30.0	46.5
▲ In-fusion FFN	55.6	66.1	**48.5**	**50.5**	57.3	42.6	**57.0**	37.1	29.1	47.3
▼ In-fusion FFN+QKVO	**55.9**	**66.9**	**48.5**	49.3	**59.0**	**43.9**	56.8	**38.9**	**29.9**	**50.1**
4B backbone, decayed, 29.0 ZFLOPs										
● Encoder	60.5	69.6	51.3	57.9	63.3	48.8	63.6	45.9	31.0	54.8
▲ In-fusion FFN	60.2	**70.0**	52.2	56.4	62.4	48.5	63.1	44.2	32.7	53.8
▼ In-fusion FFN+QKVO	**60.7**	69.2	**52.6**	**57.8**	**63.1**	**50.0**	**63.7**	**46.5**	**33.7**	**56.0**
1.7B backbone, constant LR, 11.7 ZFLOPs										
■ Encoder-free	53.1	64.1	47.2	43.6	57.6	36.9	52.0	34.6	19.8	41.4
● Encoder	52.8	65.5	47.2	44.8	53.7	40.6	55.1	35.6	26.6	45.1
▲ In-fusion FFN	51.5	64.3	46.5	40.6	54.8	40.9	55.9	36.0	25.1	46.6
▼ In-fusion FFN+QKVO	**53.3**	**65.0**	**47.9**	**45.1**	**55.1**	**41.9**	**56.3**	**37.9**	**25.5**	**47.8**
4B backbone, constant LR, 26.1 ZFLOPs										
● Encoder	58.6	67.8	51.7	53.6	61.3	43.5	58.2	40.1	27.0	48.9
▲ In-fusion FFN	57.6	67.9	**49.9**	51.6	**61.0**	46.4	60.7	43.0	29.9	52.0
▼ In-fusion FFN+QKVO	**57.9**	**68.5**	48.7	**53.8**	60.6	**48.9**	**62.8**	**45.4**	**31.6**	**55.8**

Table 31: **After the final decay, in-fusion FFN+QKVO leads the encoder at both scales and, at 4B, in every VQA category.** The decayed blocks are the endpoints of Table 3; the constant-learning-rate blocks are the 1T-token-equivalent checkpoints from which those decays start (Figure 14). The encoder-free model is trained this far only at 1.7B. Category abbreviations follow Table 1. Bold marks the better of the two in-fusion models in each block; shaded rows are the encoder reference. The 4B models share seed, initialization, and data order; the 1.7B models were trained separately.

E.5 Controls for the vision-only attention projections

Two 4B contrasts, both at checkpoints before the final decay, test whether the vision-only attention projections contribute on their own (Table 32).

The first adds only the attention projections of in-fusion FFN+QKVO (about 0.21B parameters in the first eight layers) to the encoder-free model. The second compares in-fusion FFN+QKVO with a model that has only its width-3,136 FFN branch. The attention projections alone improve Avg. VQA by 1.3 points (95% CI: 0.6–2.0), and adding them to the same-width FFN branch improves it by 2.4 points (95% CI: 1.7–3.0), with lower caption perplexity and higher ImageNet accuracy in both cases.

Matched-exposure contrast	VQA (%) ↑	Diagnostics	
	Δ Avg [95% CI]	Δ Caption PPL_3-GM	Δ ImageNet top-1
4B QKVO-only − encoder-free	+1.3 [0.6, 2.0]	−5.71%	+3.73
4B FFN+QKVO − same-width FFN	+2.4 [1.7, 3.0]	−1.28%	+0.75

Table 32: **Vision-only attention projections improve visual quality both alone and alongside an FFN branch.** Each row is the difference between two 4B models trained on the same number of tokens (matched exposure); intervals use the paired bootstrap of Table 27. Caption-perplexity changes are relative (negative is better); ImageNet changes are in top-1 points.

F Systems measurements

Section 5.3 argues that moving visual capacity into the backbone also removes the encoder's serial cost. This section compares the encoder-based recipe with encoder-free processing, in-fusion FFN, and in-fusion FFN+QKVO on the same backbones in training and at inference. We compare training cost in two ways: per-step comparisons fix the number of optimizer updates, whereas equal-FLOP comparisons fix total modeled training compute. Because encoder computation counts toward that budget, the encoder-based model completes fewer updates at equal FLOPs, so its equal-FLOP time premium is smaller than its per-step premium. In the matched 4B measurement of Table 33, for example, the encoder-based recipe takes 5.3% longer per step than encoder-free processing but only 2.9% longer at equal FLOPs. Inference comparisons report time to first token (TTFT) and output throughput on a single GPU.

F.1 Measurement setup

All comparisons hold the language backbone and data fixed and change only how images are processed, so the timing differences isolate the cost of each visual path.

Training. Training measurements time complete optimizer updates on H200 GPUs with fully sharded data parallelism (FSDP). The matched 4B comparison in Table 33 runs all four designs on 32 GPUs with a global batch of 1,024 sequences. Each row projects training time to the same modeled lifetime compute, 10.4 ZFLOPs (400B token-equivalents at 4B; Section A.7), from measured steady-state update times. Figure 15a compares per-step time at 1.7B, 4B, and 8B. Its encoder-free series comes from a separate scaling benchmark, so its 4B value differs from the paired measurement of Table 33. For in-fusion FFN, only the 4B point comes from that paired measurement; the 1.7B and 8B points compare the same architectures across separately collected runs and therefore characterize only the scale trend. Figure 15b uses the 1.7B backbone on 8 GPUs with a global batch of 128 sequences and sets the share of image rows to 10%, 30%, or 50%. Its intervals are paired 95% circular moving-block intervals over common timing windows.

Inference. Inference measurements run all four designs in the same native implementation on one H200, so they share the same decoder code. Closely following Mono-InternVL (Luo et al., 2025b), each request generates 120 tokens, and we measure time to first token and throughput at concurrency 16 as well as at concurrency 1. Each request contains one image and 256 text tokens, and each backbone is measured at 224, 336, and 448 pixels, corresponding to 256, 576, and 1,024 vision tokens. Each configuration is measured in four runs that rotate the order in which the models are built; points are medians and intervals are full ranges over the four runs. The shared and vision-only FFNs run serially, and during text-only decoding the implementation skips the vision-only FFNs, which have no vision tokens to process. In-fusion FFN+QKVO's vision-only attention projections are not skipped, so we report only its TTFT. The repeated-image diagnostic in Section F.3 instead uses vLLM 0.19.1 with encoder-output caching enabled and decoder prefix caching disabled, so a repeated image skips the encoder but not the decoder prefill.

F.2 Training time

Table 33 times the approximately 0.4B vision-only allocations of Table 29 under the 4B training setup of Section F.1. At equal lifetime compute, both in-fusion intervals include zero, so neither

variant measurably changes training time relative to encoder-free processing, whereas the encoder takes 2.6% longer than in-fusion FFN+QKVO.

Architecture	Training time	
Variant	Δ vs. encoder-free (%)	95% CI
Encoder-free	Reference	
Encoder	+2.9	$[+1.9, +4.2]$
In-fusion FFN	−0.3	$[-0.8, +0.4]$
In-fusion FFN+QKVO	+0.3	$[-0.5, +1.1]$

Table 33: **At equal training compute, the encoder takes 2.9% longer than the encoder-free recipe, while both in-fusion variants stay within measurement uncertainty of it.** Each row reports projected training time at equal modeled lifetime compute (10.4 ZFLOPs, i.e., 400B token-equivalents at 4B) relative to the encoder-free recipe for the architectures in Table 29. Intervals are paired 95% bootstrap intervals over 17 common timing windows. Timing and the quality results of Table 29 come from separate runs, so the rows do not pair quality with cost.

Scale and visual workload. Both encoder-free processing and in-fusion FFN reduce training step time relative to the encoder-based design, but encoder-free processing saves more because it adds no replacement visual stack (Figure 15a,b). In-fusion FFN still executes vision-only FFNs for every vision token, replacing about 95% of the removed encoder FLOPs at each measured image ratio. For encoder-free processing, the per-step saving shrinks as the backbone grows and increases with the share of image rows; the smaller in-fusion FFN savings are not monotonic across the three measured scale or image-ratio points. Across 10%, 30%, and 50% image rows at 1.7B, encoder-free processing reduces step time by 10.0–13.2% and in-fusion FFN by 6.0–7.4%.

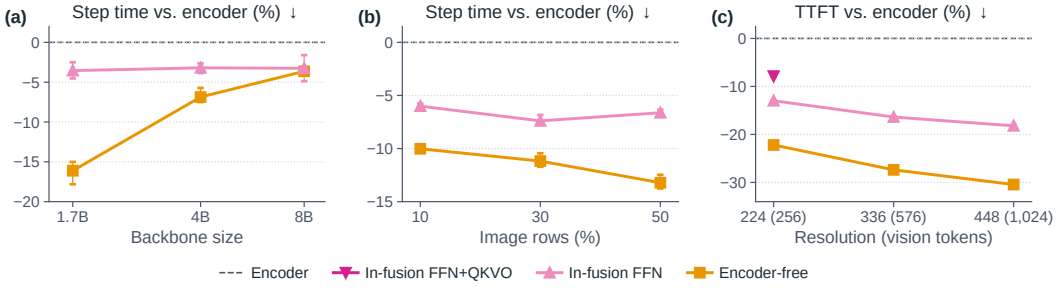

Figure 15: **Encoder-free processing and in-fusion FFN shorten training steps and time to first token relative to the same-backbone encoder.** As in Figure 6, values are time relative to the encoder (dashed line), so negative is faster. Panels (b) and (c) use the 1.7B backbone, and panel (c) reports TTFT at concurrency 16, with in-fusion FFN+QKVO measured at 224 pixels only. Training intervals are paired 95% bootstrap intervals; inference intervals are full ranges over four runs.

F.3 Inference latency

Removing the separate encoder shortens TTFT at every backbone size, most for smaller backbones (Figure 6). At 224 pixels and concurrency 16, in-fusion FFN reduces TTFT by 13.0%, 6.4%, and 3.9% for the 1.7B, 4B, and 8B backbones, respectively; encoder-free processing yields larger reductions of 22.2%, 11.5%, and 7.8%. In-fusion FFN+QKVO reduces TTFT by 7.8%, 3.2%, and 2.0% at the same scales. We did not train an 8B in-fusion FFN+QKVO model, so its 8B TTFT is measured on an untrained configuration with the same approximately 0.4B vision-only parameter budget. All reductions shrink as the backbone grows, because the encoder becomes a smaller share of the total cost. At 1.7B, they grow with input resolution (Figure 15c): from 13.0% at 224 pixels to 18.2% at 448 pixels for in-fusion FFN, and from 22.2% to 30.4% for encoder-free processing. With its vision-only FFNs skipped during text-only decoding (Section F.1), in-fusion FFN changes concurrency-16 output throughput by +0.5%, +1.0%, and +0.2% at 1.7B, 4B, and 8B; every range includes zero, so in-fusion FFN does not measurably change decoding speed. When the same image recurs, the encoder-based model can reuse cached encoder outputs instead of rerunning the encoder. In a 1.7B

diagnostic with encoder-output caching (Section F.1), reusing every image reduces its TTFT by 38.9% at concurrency 1 and 12.1% at concurrency 16 relative to no reuse. This diagnostic runs on vLLM rather than the native implementation, so these values are not comparable with the TTFT reductions above.

F.4 Encoder-side optimizations

The encoder in Table 33 and Figure 15 is an unoptimized reference implementation, and encoder-specific optimizations can narrow its gap.

A smaller compiled encoder. MoonViT-Base is the Base-size MoonViT encoder (Kimi Team, 2026) of Section A.2, with 18 transformer layers of width 768 and 152M parameters. We compile its whole tower and compare it at 4B with an in-fusion FFN model that also has 152M vision-only parameters (FFN width 2,480 in the first eight backbone layers); both use 224-pixel inputs with 256 vision tokens. MoonViT-Base takes only 0.2% longer to train than in-fusion FFN at equal lifetime FLOPs (Table 34; paired 95% interval $[0.1, 0.4]\%$) and uses 4.3 GiB more active GPU memory on the first GPU. MoonViT-Base leads Avg. VQA by 0.8 points and has slightly lower text perplexity, while in-fusion FFN leads Avg. Text, the frozen ImageNet probe, and caption perplexity. The two designs also differ in visual operators and compiled regions, so this comparison does not isolate where the visual capacity is placed.

Architecture	**Training cost**			**Text (%) ↑**	**VQA (%) ↑**	**Diagnostics**		
Variant	Time Δ (%)	95% CI	Memory Δ (GiB)	Avg	Avg	ImageNet top-1 ↑	Caption PPL_3-GM ↓	PPL_5-GM ↓
MoonViT-B	+0.2	[+0.1, +0.4]	+4.3	53.8	42.1	62.51	46.59	14.70
In-fusion FFN	Reference			54.1	41.3	63.10	46.44	14.76

Table 34: **At matched 152M visual capacity, a compiled MoonViT-Base trains almost as fast as in-fusion FFN and trades quality metrics with it.** MoonViT-B is MoonViT-Base with whole-tower compilation; both designs use the 4B backbone and 152M vision-only parameters. Training-cost columns are relative to in-fusion FFN: training time is projected at equal lifetime FLOPs with a paired 95% confidence interval (CI), and memory is active GPU memory on the first GPU. Quality columns report results at 5.2 ZFLOPs (200B token-equivalents).

Scheduling the full-size encoder. Overlapping or pipelining the full-size encoder does not shorten end-to-end training. At 4B, a one-microbatch lookahead that overlaps encoder computation with the backbone hides 20–36% of the encoder's step-time cost in steady-state timing windows, yet end-to-end training takes 0.4% longer than serial execution. Pipeline-parallel training of the full encoder-based model reduces throughput by 48.7% relative to fully sharded data parallelism on the same 32 GPUs. Standard FSDP tuning of the encoder alone changes steady-state step time by less than 0.7%.

G Limitations

To keep comparisons controlled, our pretrained encoders see the same image–text data as multimodal training, so our results do not rule out a benefit from released encoders that bring new data. We also study only dense models of up to 8B parameters on text and visual understanding, leaving larger scales and other architectures and modalities to future work.