

Peer Review

Review of: "iLLaVA: An Image is Worth Fewer Than 1/3 Input Tokens in Large Multimodal Models"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

The authors propose iLLaVA, a training-free way to cut the number of visual tokens inside a large vision-language model at two places rather than one. Inside the vision encoder, tokens are reduced at four intermediate blocks (§4.1: $r_v = 92$ tokens at each of Layers 3, 4, 5, and 6); inside the language model, a single reduction of $r_t = 108$ tokens happens at Layer $K = 2$. Importance is read from attention maps the model already computes (Eq. 1 for the encoder, Eq. 2 for the language model), and the low-attention tokens are not simply thrown away: they are folded into one new token through a fixed weight list $W_v = \{r_v+1, r_v, \dots, 1\}$, which the authors present as a one-shot replacement for greedy merging. The token budget is worked out in §3.2 and §4.1: $729 - 4 \times 92 = 361$ tokens in the encoder's later layers, then $361 - 108 = 253$ in the language model. I checked both subtractions; they are correct.

The headline claim, in the abstract and §1, is roughly $2\times$ throughput and about half the memory for a 0.2%–0.5% performance drop, across 7B, 13B, and 34B models and across single-image, multi-image, and video tasks. Evidence comes from Tables 1 and 2 (LLaVA-OneVision 7B and Qwen2-VL 7B on 18 benchmarks), Table 3 (comparison with FastV, ToMe, LLaVA-PruMerge, and Free-VideoLLM on MMMU), six ablation tables (Tables 4–9), and an honest limitations table on document and chart tasks (Table 10).

2. Whether the conclusion holds on the data

I want to begin with what I found genuinely useful. Reducing tokens inside the vision encoder as well as the language model is a sensible idea that most LLM-side pruning work ignores, and Fig. 2(a) motivates it with a measured time breakdown rather than an assertion. The ablations are unusually complete for a

short paper: Table 4 compares the merging rule against greedy merging, bipartite matching, k-means, and two CLIP-based croppers, and reports wall-clock time next to accuracy. That pairing is exactly what a reader needs. I also appreciate Table 10. Reporting drops of 8.0 (DocVQA), 12.0 (InfoVQA), and 10.2 (ChartQA) points is the kind of negative result that saves practitioners time, and the authors did not hide it.

The video results support the headline well. In Table 2, row "iLLaVA 7B" under LLaVA-OneVision, I recomputed the drops from the two rows of raw scores on the eight accuracy columns (ActivityNet-QA, Egoscchema, MLVU, NextQA, PerceptionTest, SeedBench video, VideoMME without and with subtitles): 0.2, -0.1 (a gain), 0.3, 0.4, 0.3, 0.4, 0.0, 0.1. Their mean is $1.6 / 8 = 0.2$ points. So for video, "0.2%–0.5%" is fair.

Single-image tasks tell a different story. Table 1, row "iLLaVA 7B" (LLaVA-OneVision), shows -2.8 on AI2D test, -2.7 on MMStar test, and -1.8 on SeedBench image. Averaging the six single-image accuracy columns of that row (MME left out because it is reported as two raw scores): $(2.8 + 0.9 + 0.4 + 2.7 + 0.3 + 1.8) / 6 = 8.9 / 6 \approx 1.48$ points. The two multi-image columns of the same row (MuirBench 0.7, Q-Bench 0.3) average 0.5. For Qwen2-VL, MMStar drops by 1.4 (Table 1, second "iLLaVA 7B" row). I would rephrase the abstract so the 0.2%–0.5% range is attached to video and multi-image tasks, and give the single-image average separately. The deltas in parentheses are also absolute points, not percentages. AI2D going from 81.4 to 78.6 is a 2.8-point drop, which is 3.4% in relative terms ($2.8 / 81.4$). Labeling them as "pp" or "points" would remove the ambiguity.

Three of the parenthetical deltas in Table 2 do not match the raw numbers next to them. Row "iLLaVA 7B" (LLaVA-OneVision), NextQA mc: $79.4 \rightarrow 79.0$ is -0.4, but the table prints -0.5. Same row, PerceptionTest val: $57.1 \rightarrow 56.8$ is -0.3, printed -0.1. Row "iLLaVA 7B" (Qwen2-VL), Egoscchema: $66.7 \rightarrow 66.3$ is -0.4, printed -0.3. These are small. Two of the three err in the paper's favor, so I would correct them.

There are a few places where the ablation text and the ablation tables disagree, and I think they matter more than the arithmetic slips.

- §4.6, paragraph on Table 6, says the combination {3,4,5,6} "performs slightly better than other combinations", and §4.1 uses Layers 3–6 as the default. Table 6 has no {3,4,5,6} row. Its best row is {5,6,7,8} at 57.6.
- §4.6, paragraph on Table 9, says Layer 2 is adopted "for its better performance". In Table 9, {2} gives 57.4, while {5} gives 57.5 and {8} gives 57.6.

- The default configuration (361 tokens after the encoder, 253 in the language model, merging at Layer 2) scores 57.4 in Table 8, row 253, and 57.4 in Table 9, row {2}. Yet the "iLLaVA" row in Table 4 and in Table 7 reports 57.6 at 4h10min for what reads as the same default.
- Tables 4–9 report the average over MMMU, MMBench, and MuirBench (§4.6). From Table 1, the base model gives $(48.8 + 80.8 + 41.8) / 3 = 57.13$ and iLLaVA gives $(48.4 + 79.9 + 41.1) / 3 = 56.47$. Neither equals 57.6, the value in Table 5, row "729 (Original)". Perhaps different splits or a different evaluation pass were used; I could not tell from the text.

Table 5 also puzzles me on timing. Going from 729 to 576 kept tokens cuts inference time from 7h40min to 4h24min, a 43% reduction from removing 21% of tokens, while going on to 361 saves only another 14 minutes. And 576 and 425 report the same 4h24min. A short explanation (fixed overheads, batch padding, I/O) would help readers trust the efficiency numbers.

On the comparison with prior methods, Table 3 is the only head-to-head, and it covers one benchmark (MMMU). iLLaVA reaches 48.4 against FastV's 47.6, a 0.8-point difference from single runs with no interval. Throughput is 4.22 for iLLaVA, 4.22 for ToMe, and 4.20 for FastV; memory is 19.1 for three of the five methods. So on this table, the gain is in accuracy, not in efficiency, and the efficiency advantage stated in the abstract (encoder plus language model reduction) is not visible relative to these baselines. I would add the unmodified base model as a row in Table 3 and extend the comparison to at least one video benchmark, where the encoder's share of time is largest according to Fig. 2(a).

Two smaller observations. The §4.6 sentence on Table 4 says merging approaches beat pruning approaches, but attention-based pruning (56.6) outscores bipartite matching (55.2), k-means (55.7), CLIP (53.2), and CLIPSeg (53.2). And many ablation differences are 0.1–0.3 points (Tables 5, 6, 8, 9), which I suspect sit inside run-to-run variation; without repeated runs, I would soften the ranking language.

Finally, the 2× throughput and 1.7×–2.0× memory claims (§4.2) and the 34B results rest on Figs. 1, 4, and 5. As rendered on Qeios, these figures are images, and their values are not legible in the text, so I could not verify the numbers. A table with throughput, latency, and peak memory for each model scale would make the central efficiency claim checkable.

3. Reproducibility

The code is released (GitHub link in the abstract), which is the most important thing, and the default hyperparameters are given concretely in §4.1. Several details still stand between a reader and the reported numbers.

Hardware and timing setup. I could not find the GPU type, number of GPUs, batch size, numeric precision, or the attention implementation. The last point is not cosmetic. iLLaVA reads full attention maps (Eq. 1, Eq. 2), and fused kernels such as FlashAttention or SDPA do not materialize them. If the baseline runs with a fused kernel and iLLaVA with eager attention at the merging layers, or vice versa, the throughput comparison changes. Please state which kernels were used at each layer for both the baseline and iLLaVA.

Model versions. The checkpoints are named (LLaVA-OneVision 7B, Qwen2-VL 7B and 2B, LLaVA-NeXT 7B/13B/34B) but without exact repository identifiers or revisions. Decoding parameters (greedy or sampled, maximum new tokens) are not reported, and neither is the evaluation toolkit and its version.

Token counts beyond one 729-token crop. LLaVA-OneVision encodes a single image as several crops plus a base view, and video as many frames; Qwen2-VL uses dynamic resolution, so its visual token count varies per image. §4.1 describes a fixed 729-token input. I could not find how r_v and r_t are applied per crop, per frame, or for variable-length Qwen2-VL inputs, nor how many frames were sampled per video. For a method whose whole point is token counts, these numbers matter.

Seeds and variance. Every table reports a single number per cell. No seeds or repeated runs are reported.

4. What specifically to fix

1. **Title and §4.1, last sentence.** $253 / 729 = 34.7\%$, which is slightly more than one third ($729 / 3 = 243$). The title says "fewer than $1/3$ ". Either adjust the title or the default r_t , or explain which count the title refers to.
2. **Eq. 1 and Eq. 2.** The summand A_i^v does not depend on k . I assume you mean $A_{k,i}^v$ (attention received by token i). Also, the quantity is an importance score, not a similarity between two tokens; calling it "similarity" confused me on first reading.
3. **§3.2, "Token merging".** Please write out the merge rule fully. Is W_v normalized? Without normalization, the merged token is a weighted sum with total weight $(r_v+1)(r_v+2)/2$, which would change its norm a lot. " $P_{\text{new}} \in \mathbb{R}^1$ " should carry the hidden dimension, and the cost of one weighted sum over r_v+1 vectors is $O(r_v d)$, not $O(1)$.
4. **§4.1 notation.** " $N_v = 4$ " clashes with N_v as the number of image tokens in Eq. 1; I think L_v is meant. In §3.2, "Fig. 2(a) visualizes the merging process" should probably point to Fig. 3(a).

5. **Table 2.** Correct the three deltas: NextQA -0.4 (not -0.5), PerceptionTest -0.3 (not -0.1) for LLaVA-OneVision, and Egoschema -0.4 (not -0.3) for Qwen2-VL.
6. **Tables 1 and 2, parenthetical deltas.** Relabel them as points rather than percent.
7. **Abstract and §1.** Split the "0.2%–0.5%" claim by task type; the single-image average drop on LLaVA-OneVision, by my arithmetic from the six single-image columns of Table 1, is about 1.48 points.
8. **Table 6 vs. §4.1.** Add the {3,4,5,6} row that the text describes, or align the default with the table.
9. **Tables 4, 7, 8, 9.** Reconcile the 57.6 reported for the default iLLaVA in Tables 4 and 7 with the 57.4 in Table 8 (row 253) and Table 9 (row {2}). Also explain why the base-model average in Table 5 (57.6) differs from the 57.13 that Table 1 implies.
10. **Table 3.** Name the units (throughput in samples/s? memory in GB?) instead of "%". Add the base model and a video benchmark.
11. **§4.2, §4.4, Figs. 1, 4, 5.** Give the plotted values as a table, including the 34B and 13B results and the "15% and 16%" gains over Qwen2-VL 2B, so they can be checked.
12. Report hardware, precision, batch size, and attention kernel (see §3 above).
13. **§4.6, Table 4 paragraph.** Qualify the claim that merging beats pruning; attention-based pruning (56.6) is above bipartite matching, k-means, and both CLIP-based croppers.
14. A short timing breakdown explaining the 7h40min → 4h24min jump in Table 5 would be welcome.

5. Recommendation

Minor revisions. The method is simple, practical, and released as code, and the video results in Table 2 support the main claim well. What needs work is bookkeeping: a handful of mismatched deltas, ablation text that disagrees with its tables, a title fraction that the default configuration does not quite meet, and efficiency numbers that live only inside figures. None of this requires new ideas. A careful pass over the tables, plus one table of measured throughput and memory, would, I think, make the paper considerably easier to trust and to build on.

Declarations

Potential competing interests: No potential competing interests to declare.