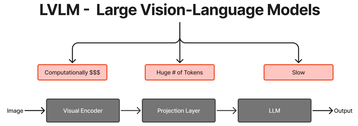
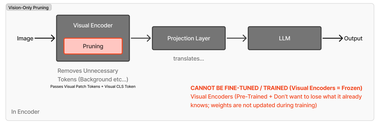
Overview + Proposed Solution

LVLM - Large Vision-Language Models





How Significance of Each Token is Calculated

Each Token has Attention Score in Relation to the CLS Token (Summary Token)

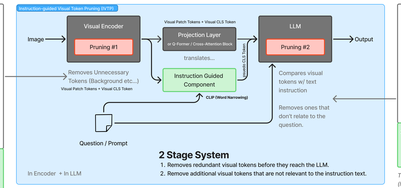
Patch Contributes **HEAVILY** to CLS Token → Keep
Patch Contributes **SLIGHTLY** → Prune

Group-wise Token Pruning (GTP)
Instead of looking at just one attention layer (noisy), **GTP looks across multiple transformer layers**, grouping them together.

It aggregates attention scores from these grouped layers to find which tokens are consistently important to the visual CLS token.

GTP uses the visual CLS token to decide which visual tokens are relevant.

Irrelevant → Prune



CLIP - Contrastive Language-Image Pretraining

CLIP's converts each word in instruction into text embeddings in the same space as CLIP's vision embedding.

Similarity of each word (token) to the **visual CLS**
High → Relevant
Low → Irrelevant

Keeps relevant word (tokens) → **psuedo CLS** (text summary)

GTP uses the pseudo CLS token to decide which visual tokens are relevant.

Irrelevant → Prune

The word "dog" will pay attention to "brown" and "collar" (high score), but not "sky" or "grass" (low score).

LVLM's

LVLM's

BLIP-2
Doesn't retrain Visual Encoder or LLM

Trains Q-Former

*Keeps them frozen.**

LLaVA
Trained using GPT-4 Question-Answer Pairs about images.

Visual Instruction Tuning
Response wide range of prompts. (Involves text and images.)

QWen-VL
Trains Visual Encoder (uses Higher Quality Images)

Improves Performance

Costs MORE compute.

Bigger Models
Higher Resolution Images
=
Better Performance
+
Higher Computational Cost
(Memory, Processing Time, Power)

Solution: IVTP
Cuts out wasteful tokens and saves computation.
(Works with existing)

USES BOTH
visual CLS + text CLS
(Image summary + question summary)

Prior Token Pruning / Token Merging Methods

Prior Token Pruning / Token Merging Methods

DynamicViT
Small neural module that learns to predict how important ea ch image patch is.

ToMe
Combining similar tokens together. (Average token)
=
Shrink Sequence Length

Number of tokens dynamic per/image.

Complex = More Tokens
Simple = Less Tokens

Image Distortion
• Blur details
• fine-grained features are lost
• accuracy affected

Merge discarded tokens into one "summary" token. (Instead of deleting completely.)

Two similar tokens have almost the same score.

Kept → Redundant
Removed → Information Loss

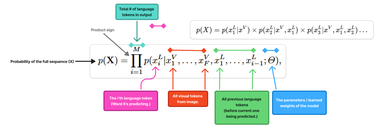
Methodology - Group-Wise Token Pruning (GTP) - 1

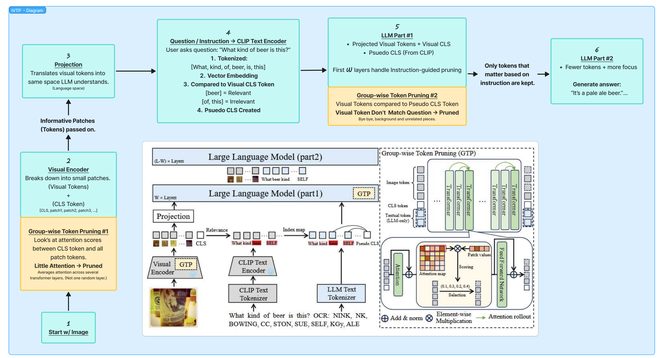
Instruction-guided Visual Token Pruning

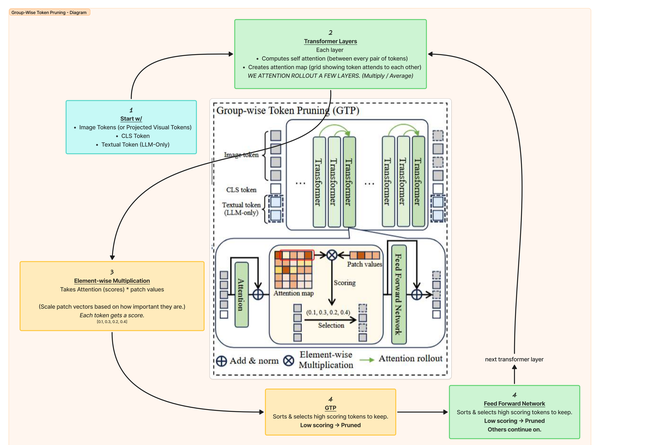
Performance (Accuracy) | **Speed** | **Cost** (Computational Resources) | **Number of Visual Tokens** (Patches kept.)

Goal: Prune efficiently without killing accuracy!

☆1☆ Response Generation







Group-Wise Token Pruning

Group-wise Token Pruning (GTP)

Attention Rollout
Combines attention maps from multiple layers to determine which tokens are important.

Tokens are evaluated not per/layer but overall (many layers) via Attention Rollout.

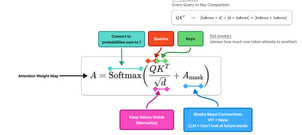
Past Approaches:
• Pruned layer by layer *(A token may appear unimportant to one layer but become important later.)*
• *Past attempts at retraining Vision Encoder (ViT) but LVLMs are frozen (not retrained).*
• Each layer removes a few

New Approach:
• Over many layers
• Gradually Shrinks Total # of Tokens

Attention Rollout works well (NLP transformers) in text models. (Why not vision pruning as well?)

Suppose the ViT has 12 layers.
They make groups of 3 layers → (4 groups total).
Each group runs independently and does attention rollout inside itself.

☆2☆ Attention Weights



☆3☆ Attention Rollout

☆4☆ Importance Score

Tell Im **Final Worth**

Higher Score → More Important Visual Token is to CLS Token
(Contributes more to model's understanding of image.)

Patch i important if CLS Token pays alot of attention to it and has strong Feature Vector.

How much the CLS token to patch i after attention rollout

Feature strength

Methodology - Group-Wise Token Pruning (GTP) - 2

Instruction-guided Token Pruning

(Inside the LLM)

Image → Visual Encoder + Pruned based on Visual CLS Token → Translated for LLM → LLM + Pruned based on Pseudo CLS Token → Output

CLIP's Text Encoder

Images ↔ Text

Alignment

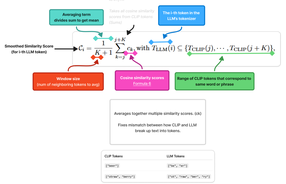
☆5☆ CLIP's Text Encoder

☆6☆ Cosine Similarity

Compute how similar each visual CLS token is to meaning of text. (Text tokens)

−1 to 1
High → More Related
Low → Less Related

☆7☆ Align CLIP Tokens to LLM Tokens



☆8☆ Pseudo CLS Token Construction

Experiments

Experimental Setup

Experimental Setup

Same datasets and settings as LLaVA-1.5

Pretraining the model:
558,000 image-text pairs from LAION-CC-SBU
(each image has a BLIP-generated caption)

Fine tuning (instruction tuned):
665,000 examples
(teams to answer instructional prompts)

Tested on 12 benchmark tasks, took the average accuracy as their main score for performance comparison.

Results normalized to make all scores comparable on the same scale. (MME benchmark differences)

Same training hyperparameters (batch size, learning rate, etc.) as LLaVA-1.5

LLaVA model, the image encoder outputs **576 tokens per image.**

They test pruning down to just **64 tokens using IVTP**

massive reduction (~89% fewer tokens)

prune every 3 transformer layers

All visual encoder layers participate in Stage 1 pruning (GTP)

Only the first 12 LLM layers perform Stage 2 pruning (instruction-guided)

Threshold details:
τ = 0.2
Meaning CS < 0.2 = Pruned

Group 1 = basic pruning:
• Randomly drop tokens.
• Keep top-k tokens most similar to the CLS token.
• Or drop spatially redundant patches.

Group 2 = vision-only pruning:
• Methods that prune tokens only based on visual information, like EVIT (Elastic ViT) and ToMe (Token Merging).

Group 3 = recent LVLM-specific pruning methods
they integrate visual-language reasoning in pruning, like:
• HoneyBee: uses abstractor modules.
• LLaMA-VID: uses context attention.
• Qwen-VL: uses visual adapter layers.

Experimental Results

Method	VQA^{v2} [↑]	CQA [↑]	VisWiz [↑]	SQA^{I} [↑]	VQA^{T} [↑]	POPE [↑]	MME [↑]	MMB [↑]	MMB^{CN} [↑]	SEED [↑]	LLaVA^{W} [↑]	MM-Vet [↑]	Avg. ↑	TFLOPs ↓
LLaVA-1.5-7B [ii]	78.5	62.0	50.0	66.8	58.2	85.9	75.5	64.3	58.3	58.6	63.4	30.5	62.7	15.4
LLaVA-1.5-7B* [ii]	79.1	62.7	49.0	67.8	58.6	86.3	72.8	66.2	58.3	58.5	63.7	31.7	63.0	15.4
Random sampling	69.0	57.1	37.9	67.2	48.5	82.5	65.6	55.4	48.0	51.0	55.8	23.6	55.1 (-7.9)	8.0 (-48.1%)
TopK	72.4	58.1	47.0	66.9	52.5	83.8	67.1	63.3	55.2	54.5	59.2	26.5	58.9 (-4.1)	8.0 (-48.1%)
Spatial pooling	73.9	59.6	46.5	67.7	52.5	82.3	68.5	63.3	56.6	54.9	59.7	28.3	59.5 (-3.5)	8.1(-47.4%)
EVIT [ii]	74.1	59.4	47.0	67.7	54.7	82.8	69.2	63.5	57.8	55.4	60.0	27.3	59.9 (-3.1)	8.0 (-48.1%)
ToMe [i]	75.1	60.0	47.1	67.5	55.3	82.4	70.4	63.9	56.5	55.2	60.5	26.6	60.0 (-3.0)	8.0 (-48.1%)
HoneyBee [i]	74.8	59.0	47.2	67.8	50.9	84.0	68.7	61.6	57.8	55.2	58.4	27.1	58.5 (-5.5)	8.1 (-47.4%)
LLaMA-VID [iii]	74.3	59.2	46.8	67.9	51.4	83.1	69.7	63.5	57.0	55.4	58.9	29.7	59.7 (-3.3)	8.2 (-46.8%)
Qwen-VL [i]	74.9	58.9	47.3	68.1	54.4	83.4	69.4	63.2	57.4	55.0	59.2	27.2	59.9 (-3.1)	8.1 (-47.4%)
IVTP (Ours)	77.8	60.4	47.9	67.8	58.2	85.7	72.6	66.1	57.4	56.4	62.8	30.5	62.0 (-1.0)	8.2 (-46.8%)

IVTP reduces computation by 46.8%. (Keeping performance identical -0.7 , 1)
Outperforms by 2-3%

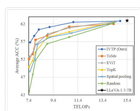
Method	VQA^{v2} [↑]	CQA [↑]	VisWiz [↑]	SQA^{I} [↑]	VQA^{T} [↑]	POPE [↑]	MME [↑]	MMB [↑]	MMB^{CN} [↑]	SEED [↑]	LLaVA^{W} [↑]	MM-Vet [↑]	Avg. ↑	TFLOPs ↓
LLaVA-1.5-13B [ii]	80.0	63.3	53.6	71.6	61.3	85.9	76.6	67.7	63.6	61.6	70.7	35.4	65.9	29.4
LLaVA-1.5-13B* [ii]	80.0	63.4	54.5	70.4	60.0	86.4	78.4	68.3	63.1	60.8	69.4	36.8	66.0	29.4
Random sampling	72.3	56.7	46.6	68.0	51.5	83.3	64.9	58.0	54.8	53.0	58.8	24.6	57.7 (-8.3)	15.4 (47.5%)
TopK	74.7	58.5	50.8	69.3	54.2	85.4	68.0	64.5	59.6	54.5	62.8	26.6	60.7 (-5.3)	15.4 (47.5%)
Spatial pooling	75.1	58.7	51.1	69.9	55.0	84.8	71.6	64.2	60.2	54.9	63.3	27.4	61.4 (-4.6)	15.6 (46.9%)
EVIT [ii]	77.2	60.2	53.4	70.1	57.9	84.6	73.6	65.3	60.1	55.4	64.9	28.6	62.6 (-3.4)	15.4 (47.5%)
ToMe [i]	76.9	61.4	53.9	70.1	57.6	85.5	73.1	65.0	61.2	56.0	65.9	32.6	63.3 (-2.7)	15.4 (47.5%)
HoneyBee [i]	76.2	61.2	52.1	70.5	59.7	83.6	73.5	63.2	61.2	55.7	66.5	32.0	63.0 (-3.0)	15.4 (47.5%)
LLaMA-VID [iii]	76.5	61.7	52.9	70.4	57.2	83.3	74.4	64.2	60.5	55.2	66.0	32.7	62.9 (-3.1)	15.5 (47.3%)
Qwen-VL [i]	77.3	61.1	52.1	70.8	56.4	84.0	71.7	65.9	61.7	56.3	66.7	31.5	63.0 (-3.0)	15.4 (47.5%)
IVTP (Ours)	78.4	62.3	54.1	70.1	60.0	85.4	77.1	67.7	63.3	59.3	68.6	35.5	65.2 (-0.8)	15.6 (46.9%)

IVTP keeps nearly identical accuracy (~0.7 difference)
Cutting computation (47%)

Only 1% accuracy loss → negligible.
Almost half the FLOPs saved → huge inference speedup.

Because IVTP also prunes inside the LLM (Stage 2), it's slightly costlier than simple pruning but achieves much better accuracy retention.

Methods	128	256	512	1024
LLaVA-1.5-7B [ii]	9.9	11.7	15.4	22.9
TopK	2.7 (-72.7%)	4.5 (-61.5%)	8.0 (-48.1%)	15.2 (-33.6%)
ToMe [i]	2.7 (-72.7%)	4.5 (-61.5%)	8.0 (-48.1%)	15.2 (-33.6%)
HoneyBee [i]	2.9 (-70.7%)	4.6 (-60.7%)	8.1 (-47.4%)	15.4 (-32.8%)
Qwen-VL [i]	2.9 (-70.7%)	4.6 (-60.7%)	8.1 (-47.4%)	15.3 (-33.2%)
IVTP (Ours)	3.0 (-69.7%)	4.7 (-59.8%)	8.2 (-46.8%)	15.5 (-32.3%)



Average Accuracy (above all others)

• As text token count increases, pruning saves less relative computation (since text processing dominates).
• But for small text sequences, IVTP saves up to 70% FLOPs.

Accuracy	Only 1% drop even with 88.9% token reduction
Efficiency	46–70% lower FLOPs (faster inference)
Fairness	Tested under same setup (LLaVA-1.5)
Advantage	+2–5% better than competing pruning methods
Integration	Plug-and-play: no retraining needed

Experimentation (Ablation and Analysis)

Ablation and Analysis

Removing parts of IVTP (like attention rollout, instruction guidance) to see how much each part matters.

All tests use LLaVA-1.5-7B and report average accuracy across 12 datasets in:
• IN = Pure Inference Mode (no retraining)
• Avg. Acc = Retrained performance

Attention Rollout

Model	Avg. Acc (IN)	Avg. Acc	TFLOPs
Group-wise	55.1 (-7.6)	56.1 (-5.6)	8.1 (-46.4%)
Layer-wise	56.2 (-6.4)	58.5 (-3.2)	8.1 (-46.4%)
Rollout	59.6 (-2.5)	62.0 (-1.0)	8.2 (-46.8%)
Residual	**59.6 (-2.5)**	**62.0 (-1.0)**	**8.2 (-46.8%)**

• Group-wise: Combines attention within a group of layers.
• Layer-wise: Uses each layer's CLS attention individually. [Noisy]
• Rollout: Combines attention maps across layers → smoother information flow.

Attention rollout performs best, gaining +4% over group-wise.

Attention Aggregation Strategy

Model	Avg. Acc (IN)	Avg. Acc
mean	56.1 (-6.0)	60.1 (-2.9)
max	55.9 (-6.2)	60.7 (-2.3)
multiply	56.7 (-5.4)	60.5 (-2.5)
Residual	**59.6 (-2.5)**	**62.0 (-1.0)**

• Mean / Max / Multiply: Simple mathematical aggregation.
• Residual: Combines attention weights using the residual connection (more balanced and stable).

Residual aggregation gives the best accuracy. (better captures hidden dependencies.)

Instruction-Guided Selection

OTT	TR	RT	Avg. Acc (IN)	Avg. ACC
			56.7 (-5.4)	59.2 (-3.8)
✓			58.3 (-3.7)	60.2 (-2.8)
✓	✓		57.3 (-4.7)	59.9 (-3.1)
✓	✓	✓	**59.6 (-2.5)**	**62.0 (-1.0)**

• OTT: Original textual tokens from the LLM.
• TE: CLIP text encoder (cross-modal alignment).
• RT: Relevance threshold (τ filtering).

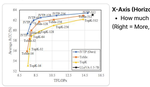
Combining all three components (text tokens + CLIP encoder + thresholding) yields the best accuracy. Each alone helps slightly, but together they guide pruning most effectively.

Pruning in LLM

Model	Avg. Acc (IN)	Avg. Acc
TopK	54.4 (-8.9)	56.9 (-6.1)
TopK^{+}	54.9 (-8.1)	58.7 (-4.3)
ToMe [i]	57.5 (-5.5)	60.0 (-3.0)
ToMe^{+}	57.7 (-5.3)	60.1 (-2.9)
IVTP V	56.5 (-6.5)	60.4 (-2.5)
IVTP	**59.6 (-2.5)**	**62.0 (-1.0)**

Extending pruning directly into the LLM doesn't help much; performance mainly comes from Stage 1 (ViT) and instruction guidance in early LLM layers.

Scaling Token Pruning



X-Axis (Horizontal) → TFLOPS
• How much computation is needed.
(Right = More, Left = Less)

• At 256–512 tokens, IVTP matches or exceeds full model accuracy.
• Below 64 tokens, other methods collapse, but IVTP still maintains 4–5% better performance.
• Achieves 10–30% TFLOP reduction with no accuracy loss.

IVTP finds an optimal range (~256 tokens) balancing efficiency and accuracy, adapting pruning to instruction relevance.

Group Layers

Layers Avg.	Acc (IN)	Avg. Acc
ALL	56.7 (-5.4)	59.2 (-3.8)
2	58.9 (-3.2)	60.2 (-2.8)
3	**59.6 (-2.5)**	**62.0 (-1.0)**
4	59.1 (-2.8)	61.5 (-1.5)
6	58.4 (-3.4)	59.9 (-3.1)

Grouping 3 layers gives best results.
More layers → over-smoothing
Fewer layers → lose inter-layer context

Computational Complexity

Methods	VIT	LLM	extra Text
LLaVA-1.5-7B [i]	2.3 B	15.6B	15.6B
TopK	0.196 T	7.7T	7.9T
ToMe [i]	0.196 T	7.7T	7.9T
Qwen-VL [i]	0.390 T	7.7T	8.08T
IVTP (ours)	0.392 T	3.8B	3.9B

IVTP's total cost is only 2% higher than ToMe / TopK, but delivers much better accuracy.

Visualization

Original Image → TopK Pruning → IVTP Pruning
IVTP better understands which image patches matter for the given instruction.