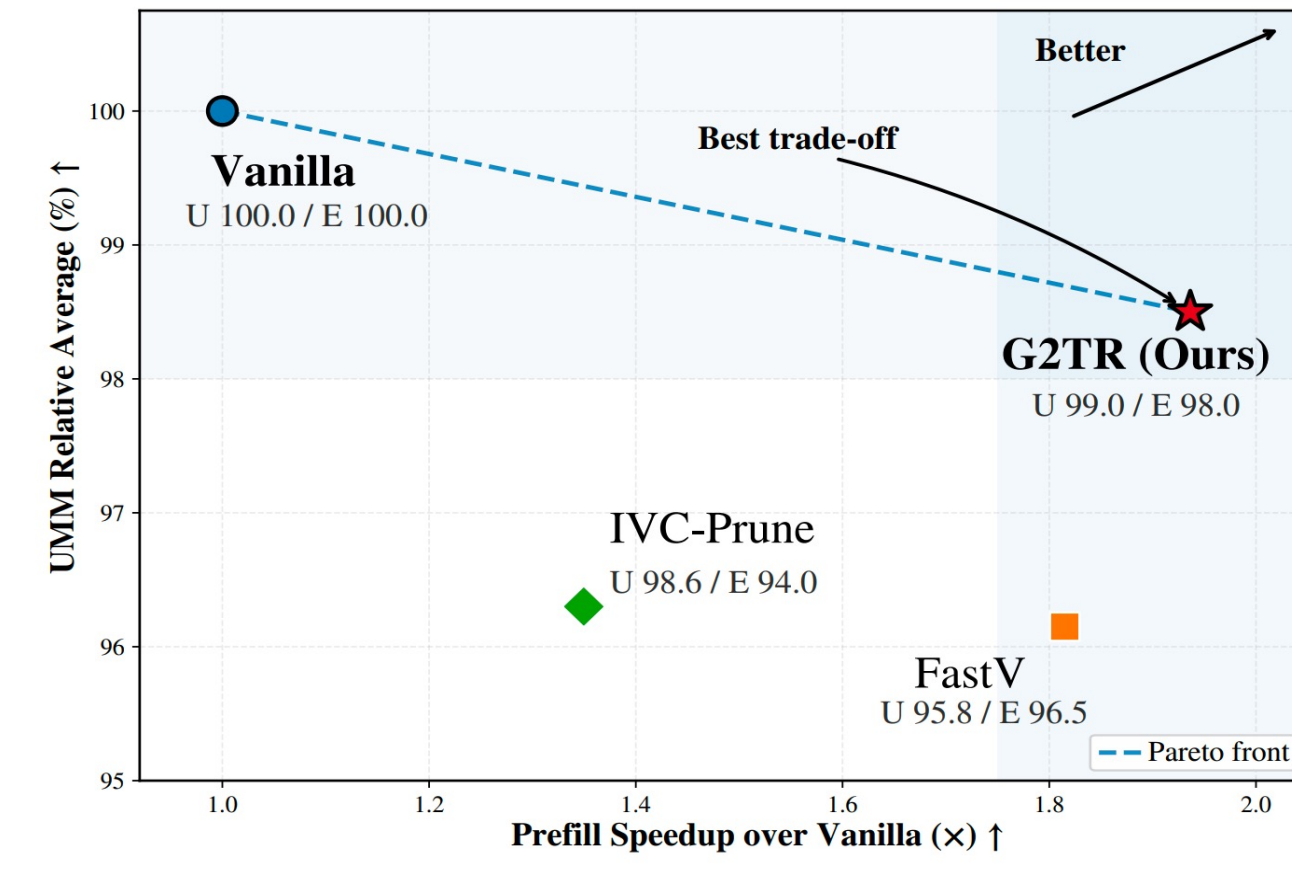
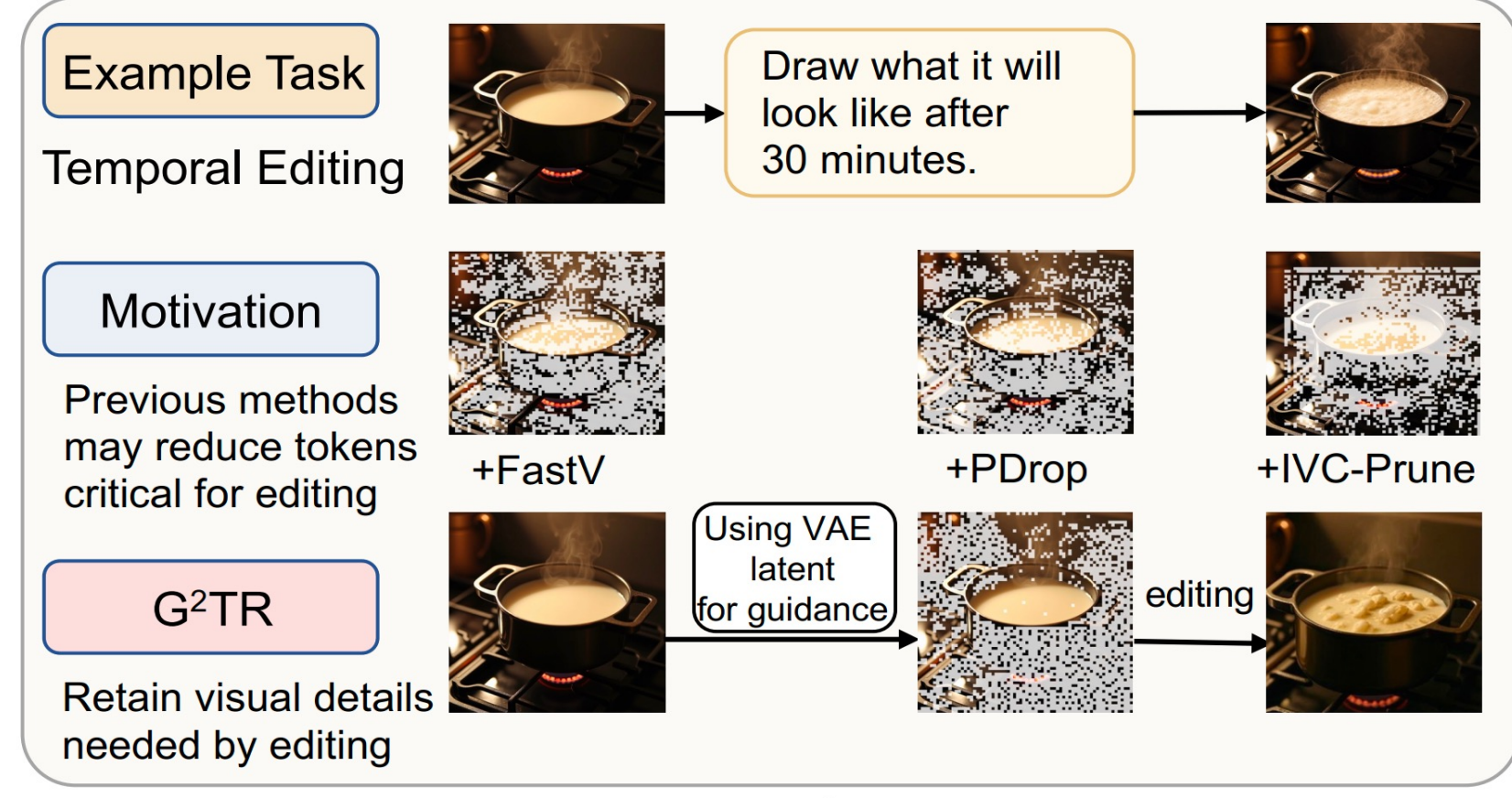


Motivation

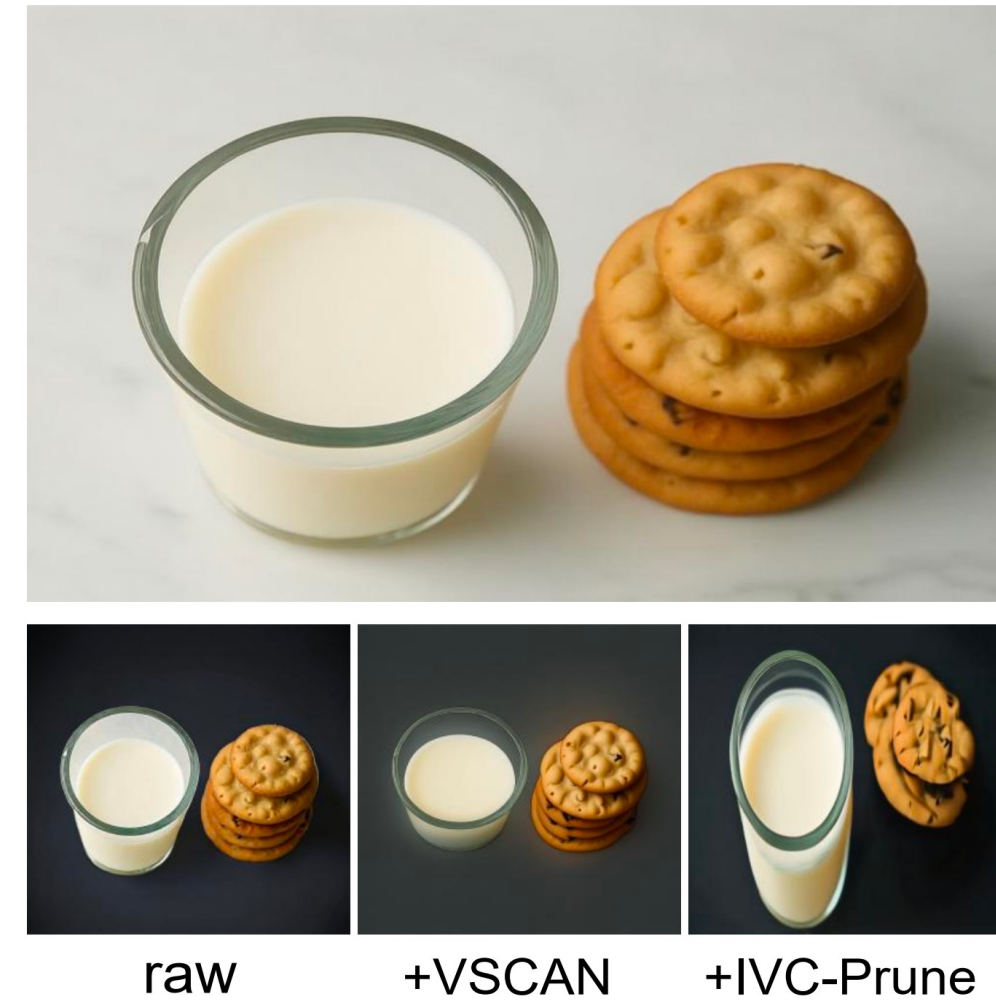
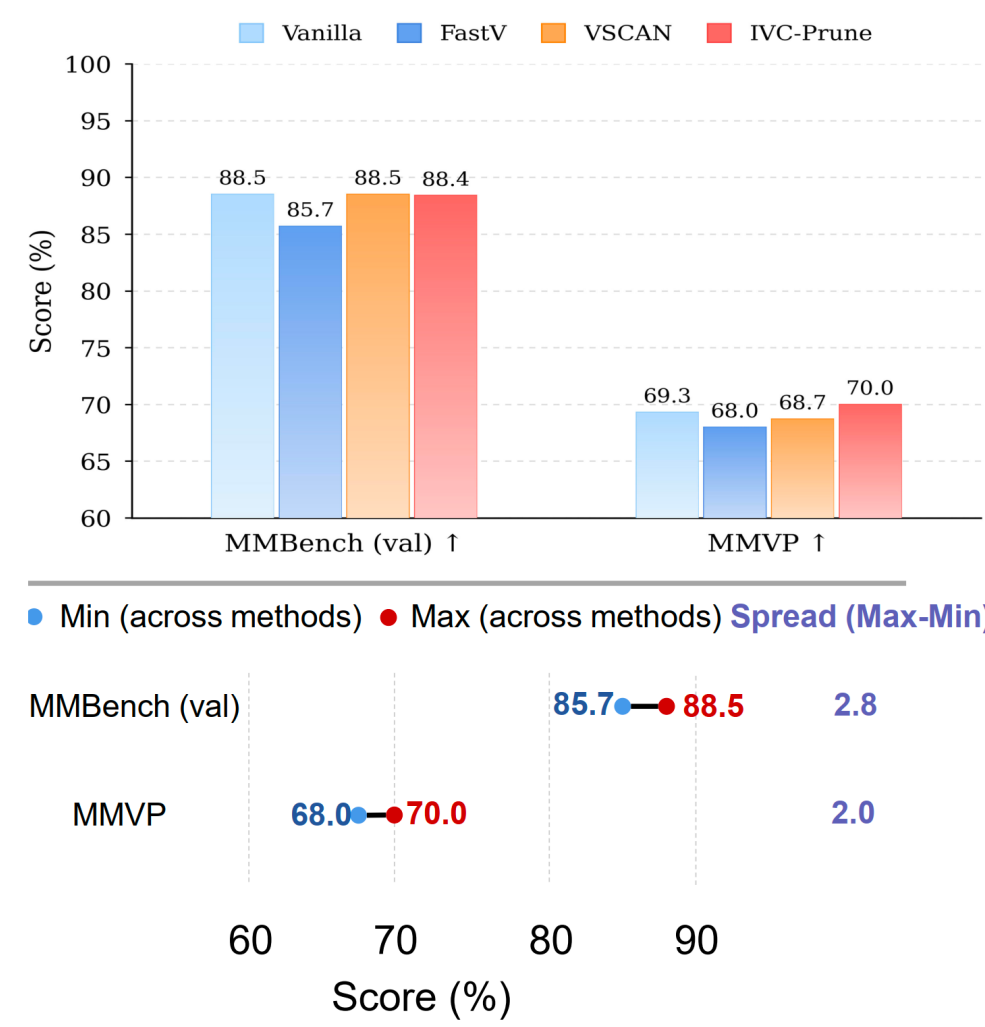


Our contributions:

- First systematically considering visual token reduction in UMMs
- A generation-guided visual token reduction method, G²TR
- Good performance on both image understanding and editing tasks.

Observations and Research Questions

Name	Visual Token Selection Signal
FastV [3]	inner LLM attention score
W-FastV [34]	window mechanism + inner LLM attention score
PDrop [40]	inner LLM attention score
VSCAN [42]	ViT attention score + inner LLM image-text similarity
IVC-Prune [28]	RoPE + image-text similarity

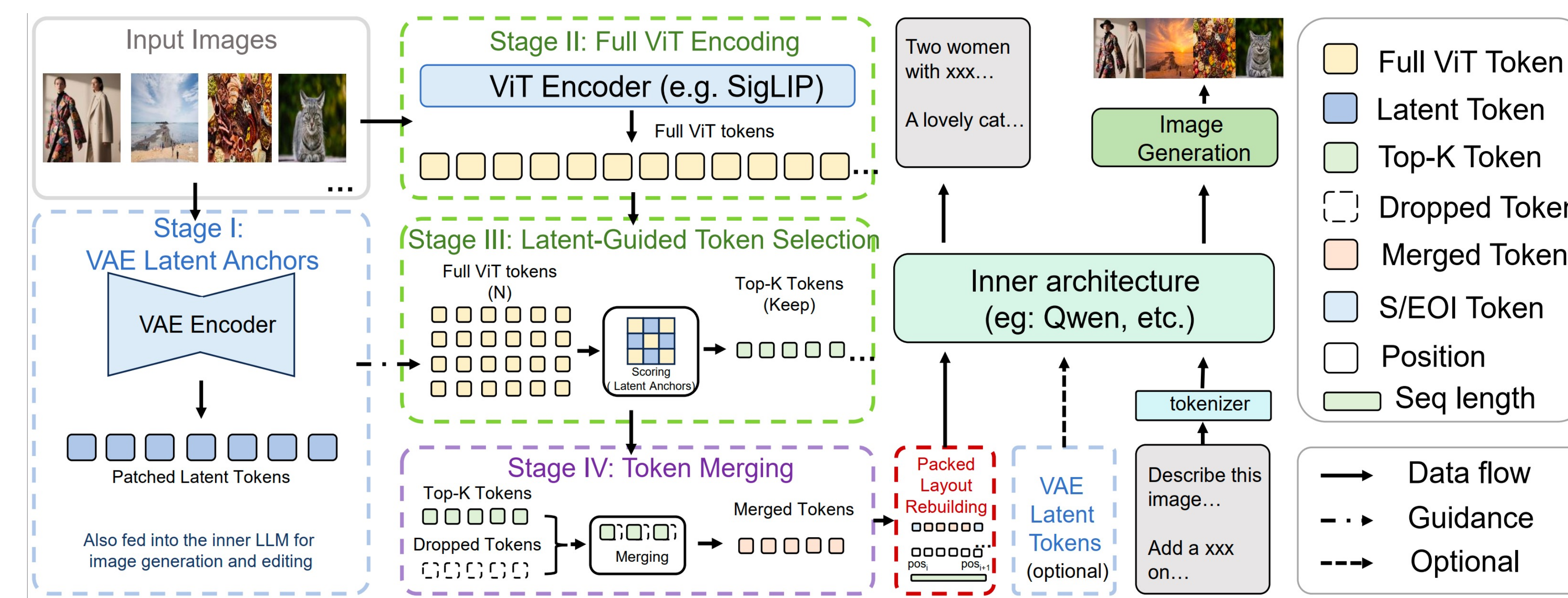


Understanding is relatively insensitive to the pruning rule, while editing should also be considered in UMM visual token reduction.

- Concentrating on the **ViT tokens** following previous works
- Different visual token reduction methods lead to **limited difference** on understanding capabilities
- Visual tokens chosen by previous methods are **not consistent** with image editing importance

Can we employ visual token reduction on UMMs to improve the efficiency of them?

G²TR: Generation-guided Token Reduction



Two-stage selecting-and-merging is proposed to reduce the understanding-side visual tokens of UMMs.

- **Latent-guided Importance Estimation** to select which tokens should be kept.
- **Balanced sampling** is applied to reduce the problem of token over-concentration.
- Redundant tokens are **merged** into the nearest cells.
- Implementation friendly for **Flash-Attention**

Algorithm Details

1. Match each ViT token to its corresponding VAE latent anchor according to the 2D spatial grids.

$$c(i) = \left(\left\lfloor \frac{p_i H_g}{H_u} \right\rfloor, \left\lfloor \frac{q_i W_g}{W_u} \right\rfloor \right)$$

2. Score each ViT token by its consistency with the matched generation-side latent:

$$s_i = \cos(u_i, z_{c(i)}) = \frac{u_i^\top z_{c(i)}}{\|u_i\|_2 \|z_{c(i)}\|_2}$$

3. Select high-score tokens while maintaining spatial coverage across latent regions.

$$b_c = \arg \max_{i:c(i)=c} s_i \quad B = \{b_c\}, \quad S = \text{TopK}(B, K)$$

4. Nearest token merging.

$$\tilde{u}_j = \frac{\lambda u_j + \sum_{i \in \bar{S}: n(i)=j} u_i}{\lambda + |\{i \in \bar{S}: n(i) = j\}|}$$

The generation branch is already there, used as a **free supervisory signal** for token importance.

Main Result

G²TR performs well in both understanding and editing tasks.



Models	Methods	Avg. Tokens	MME	MMBench (dev)	MMVP	RWQA	Rel. Avg.
InternVL-U	Vanilla	100%	2057	80.5	46.0	48.8	100%
	FastV [3]	54%	2037	77.9	38.0	45.5	92.9%
	W-FastV [34]	54%	1926	78.6	39.3	45.2	92.3%
	PDrop [40]	61%	1953	77.7	38.7	45.6	92.0%
	VSCAN [42]	50%	1968	78.1	40.0	43.5	92.2%
	IVC-Prune [28]	50%	2020	78.4	39.0	46.0	93.7%
	G ² TR (ours)	50%	2034	78.8	40.0	46.8	94.9%
BAGEL-7B	Vanilla	100%	2388	88.5	69.3	60.2	100%
	FastV [3]	54%	2306	85.7	68.0	55.2	95.8%
	W-FastV [34]	54%	2287	85.9	67.3	55.4	95.6%
	PDrop [40]	61%	2312	84.0	66.7	53.8	94.1%
	VSCAN [42]	50%	2339	88.5	68.7	48.4	94.4%
	IVC-Prune [28]	50%	2332	88.4	70.0	57.8	98.6%
	G ² TR (ours)	50%	2318	88.5	70.0	59.0	99.0%

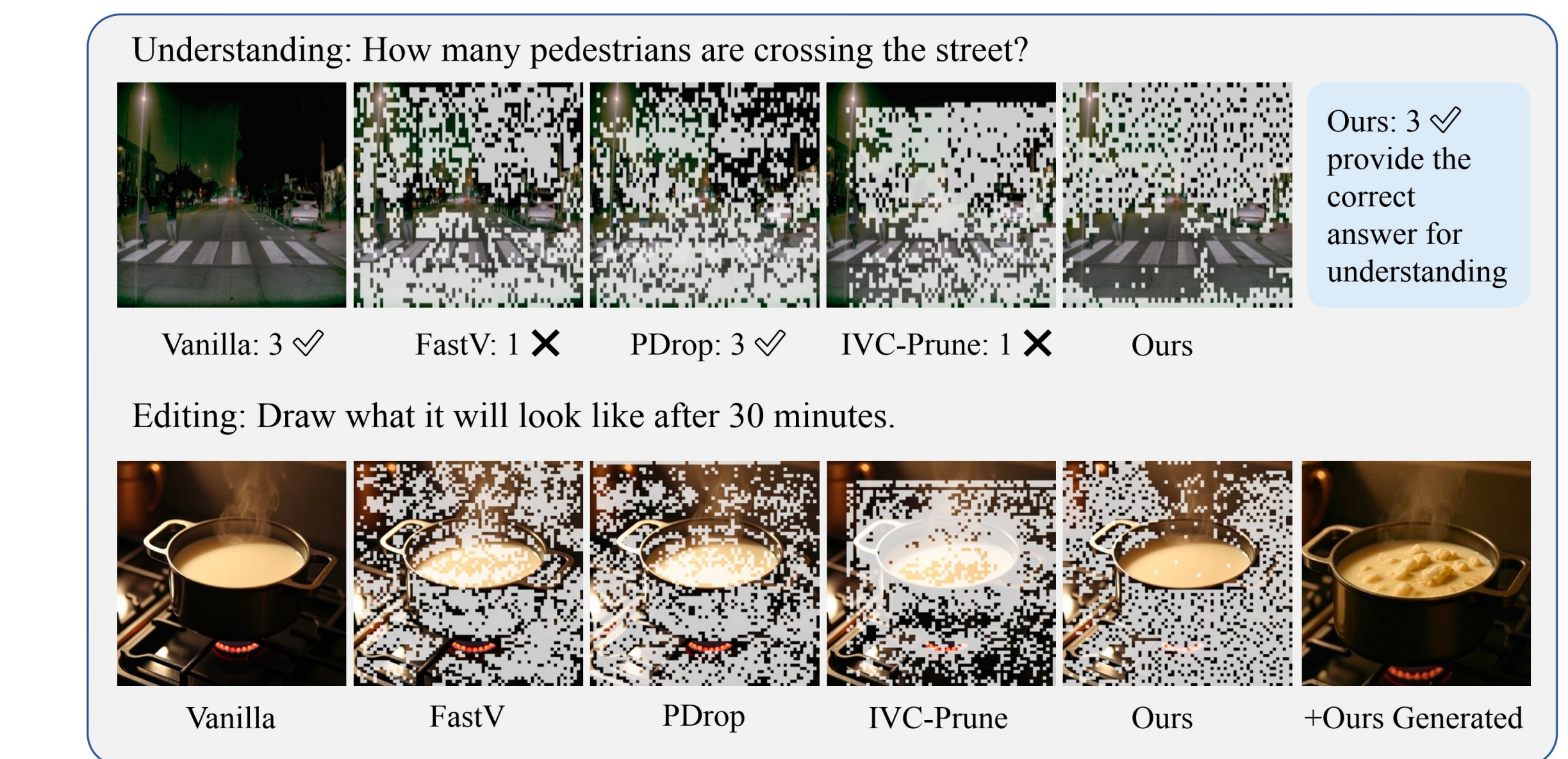
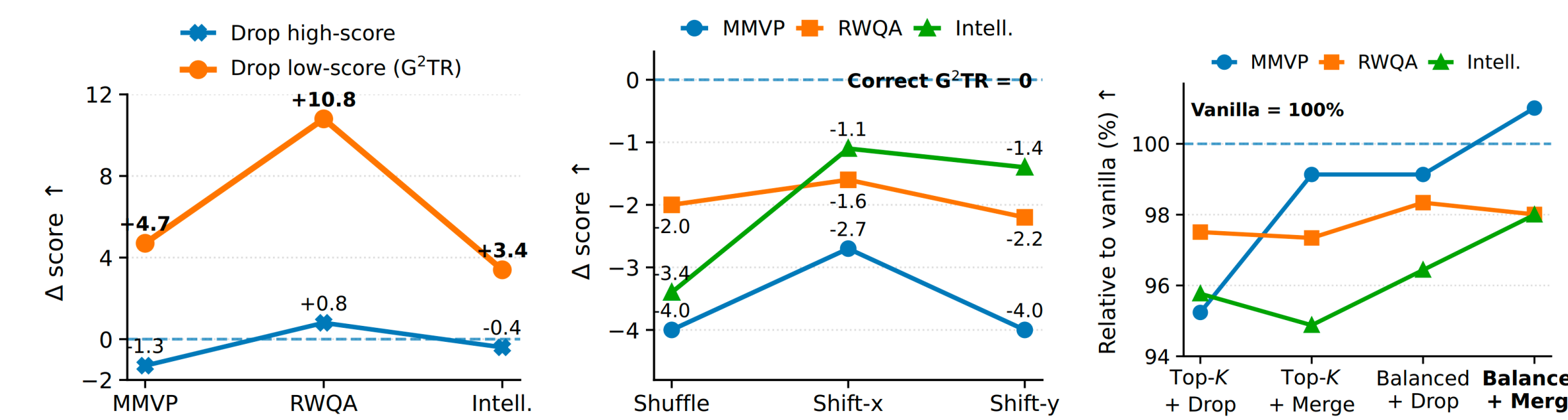
Efficiency

G²TR speeds up the prefilling stage effectively.

Model	Method	KV Cache (MB) ↓	Avg. Tokens (%) ↓	FLOPs Ratio ↓	Prefill FLOPs (T) ↓	Decode Latency (ms/token) ↓
BAGEL-7B	Vanilla	42.02 (1.00×)	100% (1.00×)	100.00% (1.00×)	4.1325 (1.00×)	49.60 (1.00×)
	FastV	22.04 (1.91×)	54% (1.85×)	55.10% (1.85×)	2.2769 (1.82×)	48.04 (1.03×)
	IVC-Prune	22.06 (1.90×)	50% (2.00×)	74.09% (1.37×)	3.0619 (1.35×)	49.30 (1.01×)
	G ² TR (Ours)	22.06 (1.90×)	50% (2.00×)	51.64% (1.94×)	2.1341 (1.94×)	47.88 (1.04×)

Quantitative results, ablations

Models	Methods	GEdit (EN) G_SC G_PQ G_O	Intell.	Rel. Avg.
BAGEL-7B	Vanilla	7.36 6.83 6.52 44.9	100%	
	FastV	7.09 6.58 6.35 43.0	96.5%	
	VSCAN	7.03 6.65 6.32 42.6	96.2%	
	IVC-Prune	6.82 6.52 6.27 41.1	94.0%	
	G ² TR (ours)	7.18 6.67 6.42 44.0	98.0%	



Generation-understanding agreement can be an important factor for UMMs!