DYNAMIC TOKEN MERGING FOR EFFICIENT VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) represent each image with hundreds of visual tokens, which can make inference costly even when a prompt depends on only part of the image. **Dynamic Token Merging (DTM)**, introduced in MrT5 for byte-level text encoders, learns which tokens to remove: a lightweight delete gate scores contextualized hidden states, acts as a differentiable attention bias during training, and physically removes tokens at inference. We argue that DTM is a general token deletion framework rather than a byte-level technique, and extend it to the vision modality and to decoder-only models. Applying DTM to T5Gemma 2 and Gemma 3 yields V-MrT5Gemma2, which deletes visual tokens inside a bidirectional multimodal encoder, and V-MrGemma3, which compacts the prompt and its key–value caches during decoder-only prefill. Across ten benchmarks, DTM's advantage over attention-based and random deletion grows with the deletion rate and is largest on document, chart, and scene-text reading. At 80% visual token deletion (≈60% fewer FLOPs), DTM raises accuracy over FastV from 51.8 to 63.0 on ChartQA and from 63.3 to 67.8 on TextVQA with V-MrT5Gemma2, as well as from 62.2 to 68.8 and from 71.2 to 75.1 with V-MrGemma3, retaining 90–96% of unpruned performance on both tasks with significant efficiency gains. These results support DTM as a general framework for efficient transformers.

1 INTRODUCTION

Long token sequences are a major source of inference cost in transformer models, and this problem arises across many input modalities. Vision-language models (VLMs), for example, may represent a single image with hundreds of visual tokens, even when a prompt depends on only part of the image: reading one value from a bar chart may depend on only a few labels and bar ends. Byte-level language models encounter a similar problem, since they process much longer sequences than conventional subword models, even though a word rarely needs one representation per byte.

Despite these differences, both settings produce long token sequences that must be processed repeatedly across transformer layers. One way to improve efficiency is to shorten the sequence after the first few layers, so that later layers operate on fewer tokens. The challenge is determining which representations can be removed while preserving the information needed for the end task. In VLMs, this choice is often made by a fixed rule computed from the model's own attention, such as **FastV** (Chen et al., 2024b); the rule itself does not adapt when the model is fine-tuned for a downstream task, and simple selectors, including random selection, can be competitive with it (Wen et al., 2025).

MrT5 introduced **Dynamic Token Merging** (DTM; Kallini et al. 2025) as a learned mechanism for reducing sequence lengths in byte-level text encoders. DTM applies a scalar *delete gate* to contextual hidden states to identify tokens that can be removed, with the gate trained jointly with the model: during training, its scores act as soft attention biases, and at inference, low-scoring tokens are physically removed. Because token selection operates on contextualized representations rather than the original input tokens, information from tokens selected for deletion is already encoded in the representations that remain, which is the sense in which DTM *merges* tokens. In MrT5, DTM shortened byte-level sequences by up to 75% with little loss in downstream accuracy. Although DTM was originally developed for byte-level text encoders, its mechanism is not inherently tied to text: the gate reads only hidden states, and soft and hard deletion act only through attention.

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.

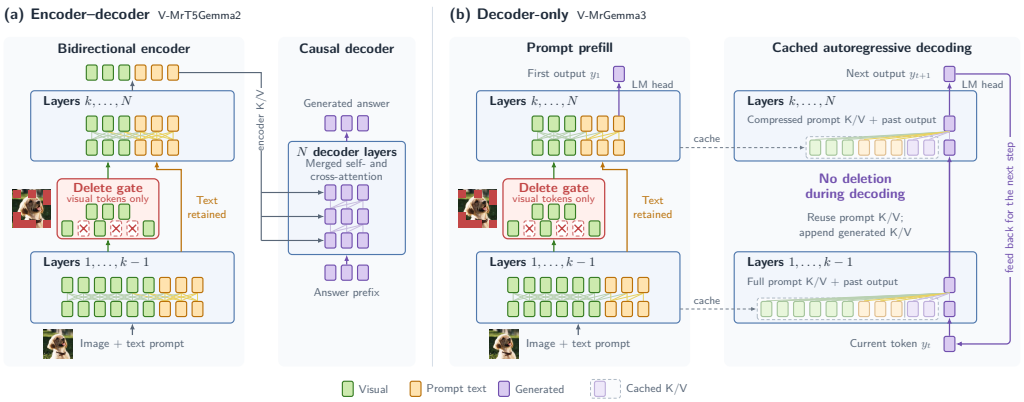


Figure 1: **Dynamic Token Merging (DTM) in encoder–decoder and decoder-only VLMs. (a)** V-MrT5Gemma2 deletes selected visual tokens before encoder layer k, retaining all text tokens. The shortened encoder output feeds the decoder's merged self- and cross-attention. **(b)** V-MrGemma3's prefill uses bidirectional attention within images and causal attention for text, caching the full prompt before the gate and the compacted prompt after it; decoding reuses these caches and appends generated K/V without further deletion. Both diagrams show hard deletion at inference. The gate and deletion rule are the same in both architectures; only where the gate takes effect differs.

In this paper, we broaden DTM beyond its original setting and position it as a general framework for learned token deletion in transformers. We characterize DTM by three components (a learned delete gate, a soft-to-hard deletion paradigm, and a deletion-rate controller) and extend the framework along two dimensions: from text to visual-token sequences, and from bidirectional encoder architectures to decoder-only models. We instantiate these extensions in two general settings. First, V-MrT5Gemma2 applies DTM to visual tokens within the bidirectional multimodal encoder of the encoder–decoder T5Gemma 2 (Zhang et al., 2025a), the setting closest to MrT5. Second, V-MrGemma3 extends the framework to visual-token compression in the decoder-only Gemma 3 (Gemma Team et al., 2025), where deletion occurs during prompt prefill and compacts the key–value caches reused during generation (Figure 1). Together with MrT5, these models form a broader DTM family that applies the same compression method across distinct modalities and architectures.

We fine-tune both model architectures with LoRA (Hu et al., 2022) on ten benchmarks, from document and chart reading to counting, hallucination detection, and grounding, and compare DTM with random deletion and two attention-based selectors that receive the same fine-tuning. **DTM shows particularly strong gains under aggressive compression on tasks requiring fine-grained visual reading, including document, chart, and scene-text understanding.** At matched 80% visual token deletion, which removes 58–63% of language-model FLOPs on document, chart, and scene-text tasks, the encoder–decoder V-MrT5Gemma2 achieves 57.5 ANLS on DocVQA and 63.0 relaxed accuracy on ChartQA, compared with 49.5 and 51.8 for our adapted FastV, and the decoder-only V-MrGemma3 extends these gains even though its gate cannot see the question.

Collectively, our contributions are: **(1) DTM for the vision modality.** We apply DTM to visual tokens while protecting text, and show on ten benchmarks and three backbones that its advantage over attention-based and random deletion grows with the deletion rate, both when each deployment budget has its own fine-tuned checkpoint and when one checkpoint serves budgets set at inference. **(2) DTM for decoder-only models.** We extend the delete gate from encoders to decoder-only models by compacting the prompt and its per-layer caches during prefill, and show that the gains carry over to Gemma 3. **(3) DTM as a general framework.** We identify the components that define DTM independently of its byte-level origin and show that they transfer across modality and architecture, with analyses that characterize where the framework is most effective.

2 Related Work

Non-Learned Visual Compression. Non-learned approaches specify the compression rule without training a separate selector. These rules can operate before the multimodal language backbone

or within it. VisionZip (Yang et al., 2025) selects and merges visual features before language-model processing using attention and feature similarity. Within the backbone, **FastV** (Chen et al., 2024b) ranks visual tokens using received attention, while SparseVLM (Zhang et al., 2025b) uses selected text tokens to guide pruning and recycles information from discarded tokens. PyramidDrop (Xing et al., 2025) follows a similar attention-based pruning principle, distributing token removal across multiple depths with a progressively shrinking visual token budget during training and inference. FEATHER (Endo et al., 2025) addresses positional effects through RoPE-free scoring, spatial coverage, and two pruning stages. LLaVolta (Chen et al., 2024a) instead compresses visual context through nonparametric average pooling inside the language model and studies schedules that change compression over training, whereas explicit merging methods such as Token Merging (ToMe; Bolya et al. 2023) combine token representations directly. These methods use fixed compression rules, although their surrounding models may be fine-tuned.

Learned Visual Compression. Learned approaches optimize the selection function through training, with precedents in both vision and language. DynamicViT (Rao et al., 2021) trains importance predictors with differentiable attention masking and progressively sparsifies vision-transformer tokens. VisionSelector (Zhu et al., 2026) trains a scorer before the language model using differentiable top-k selection. LearnPruner (Takezoe et al., 2026) combines a learned visual scorer after the vision encoder with text-guided selection at an intermediate language-model layer. In language, MrT5 (Kallini et al., 2025) introduced **DTM** for byte-level models (Section 3.2), which we extend to visual tokens and decoder-only models. Unlike selectors placed before the language model, **DTM** deletes tokens inside the backbone once, after attention has allowed retained states to absorb information from tokens that are later removed.

3 Background

3.1 T5Gemma 2 and Gemma 3

Gemma 3 is a decoder-only transformer (Gemma Team et al., 2025). To create T5Gemma 2, Zhang et al. (2025a) initialize both the encoder and decoder stacks from pretrained Gemma 3 checkpoints and train the resulting encoder–decoder model with UL2 (Tay et al., 2023). Both architectures process images with a SigLIP vision encoder (Zhai et al., 2023), then pool and project its representations into 256 visual tokens in the language model's input embedding space. In T5Gemma 2, these tokens enter a bidirectional encoder together with the text prompt, and each decoder layer attends jointly to its preceding decoder states and to all final encoder states under a single normalization, which Zhang et al. (2025a) call *merged attention*. In Gemma 3, the visual tokens are part of the prompt of a single causal stack; attention is bidirectional within an image but causal elsewhere, so image tokens cannot attend to a question that follows the image. Both models interleave five sliding-window (local) attention layers with one global-attention layer; indexing layers from 0, with layer 0 being the input states, layers $6, 12, 18, \ldots$ are global. Appendix B provides configurations, attention windows, and architecture diagrams for all three models.

3.2 MrT5 and Dynamic Token Merging

MrT5 (Kallini et al., 2025) introduced **DTM** to shorten the long input sequences of the byte-level encoder–decoder model ByT5 (Xue et al., 2022). A delete gate is placed before a fixed encoder layer k: layers $1, \ldots, k-1$ contextualize the full sequence, and layers k onward process only the tokens that the gate retains. Let $\mathbf{H} \in \mathbb{R}^{n \times d}$ contain the n hidden states entering layer k. The gate maps them to scores

$$\mathbf{g} = c\,\sigma\big[-\big(\mathrm{RMSNorm}(\mathbf{H})\,\mathbf{w} + b\mathbf{1}_n\big)\big] \in (c, 0)^n, \qquad c < 0, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are learned, RMSNorm (Zhang & Sennrich, 2019) has its own learned scale, σ is the sigmoid function, and c is a fixed constant. We negate the argument of σ so that a larger projection favors retention; this is equivalent to MrT5's parameterization up to the signs of $\mathbf{w}$ and b. The gate adds only $2d + 1$ parameters and negligible FLOPs (Appendix G.4.2). A score g_j near zero preserves token j, while a score near c suppresses it.

During training, **DTM** deletes tokens *softly*: each score is added to the attention logits of the corresponding key in every attention layer after the gate, including the decoder's attention to encoder

Table 1: **Three instances of the DTM framework.** All share the delete gate, soft-to-hard deletion, and controlled deletion objective (Section 3.2). MrT5 (Kallini et al., 2025) uses PI control with full fine-tuning; our instances use PID control with LoRA. Only the tokens eligible for deletion and where the shortened sequence takes effect differ across instances.

Instance	Backbone	Deletes	Gate after	Sees question	Hard deletion shortens
MrT5	ByT5 (enc.–dec.)	Bytes	Encoder layer 3	—	Encoder, cross-attn.
V-MrT5Gemma2	T5Gemma 2 (enc.–dec.)	Visual	Encoder layer 5	Yes	Encoder, merged attn.
V-MrGemma3	Gemma 3 (dec.-only)	Visual	Layer 5 (prefill)	No	Prompt, KV cache

states. For query i and key j in one attention head,

$$\widetilde{s}_{ij} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_h}} + M_{ij} + g_j, \qquad A_{ij} = \frac{\exp(\widetilde{s}_{ij})}{1 + \sum_u \exp(\widetilde{s}_{iu})}, \tag{2}$$

where $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^{d_h}$ are the query and key after normalization and positional transforms, and $M_{ij} \in \{0, -\infty\}$ is the host model's attention mask (causal or bidirectional, local or global). Soft deletion emulates removal while keeping the sequence intact, so the task loss trains the gate jointly with the model. Following MrT5, attention uses $\mathrm{softmax}_1$ (Miller, 2023), whose additional 1 in the denominator lets attention mass dissipate when every key is suppressed; a standard softmax would be invariant to a uniform bias. At inference, **DTM** deletes tokens *hard*: states with $g_j \leq \tau$, for a threshold $c < \tau < 0$, are physically removed, so all later layers process a shorter sequence (Figure 5 in Appendix A.1 illustrates both).

The gate is encouraged to delete by the regularizer $\mathcal{L}_G = \frac{1}{n}\sum_j g_j$, giving the objective $\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \alpha_t \mathcal{L}_G$. To reach a target deletion rate δ, MrT5 adjusts α_t with a controller resembling a proportional–integral (PI) controller from classical control theory, driven by the error $e_t = \delta - r_t$ between the target and the observed rate $r_t = \frac{1}{n}\sum_j \mathbf{1}_{[g_j \leq \tau]}$ (Equation 3 without its D term). MrT5's diagnostic tasks show that the gate learns to merge relevant context into retained tokens, not only to discard irrelevant ones, which gives the method its name.

4 Extending the DTM Framework

MrT5 is one instance of a more general recipe. We define a **DTM model** as a transformer with three components: **(i)** a learned delete gate that scores contextualized hidden states at a fixed layer boundary (Equation 1); **(ii)** soft deletion through attention biases during training and hard deletion at inference (Equation 2); and **(iii)** a mechanism for targeting a specific deletion rate, typically through a regularizer (Equation 3). None of these components refers to bytes or requires an encoder. What differs across instances is which tokens may be deleted and the layer at which sequences are shortened (Table 1); we instantiate **DTM** for visual tokens and decoder-only models below.

4.1 Application to the Vision Modality

To apply **DTM** to images, we restrict deletion to visual tokens. The gate scores every visual hidden state entering layer k; text tokens and image-boundary markers receive $g_j = 0$ and are never deleted, so the prompt reaches all later layers intact. We compute the regularizer and the observed deletion rate specifically over the *visual tokens* $\mathcal{V}_t$ of each batch, $\mathcal{L}_G = |\mathcal{V}_t|^{-1} \sum_{j \in \mathcal{V}_t} g_j$, so the target δ represents the fraction of visual tokens removed, rather than text and vision tokens together. In V-MrT5Gemma2, the gate sits in the bidirectional encoder, where visual states can attend to the question before they are scored. As in MrT5, hard deletion shortens both the remaining encoder layers and the encoder keys and values that every decoder layer reads through merged attention (Figure 1a). Retained tokens keep their original position indices.

4.2 Application to Decoder-Only Models

MrT5's gate operates in an encoder, whose output is computed once and stays fixed during generation. A decoder-only model instead processes the prompt and the answer in one causal stack,

reusing cached keys and values at every generation step. We therefore apply **DTM** during *prompt prefill* (Figure 1b). Layers $1, \ldots, k-1$ process and cache the full prompt; the gate then scores the visual prompt states, and hard deletion removes the selected states so that layers $k, \ldots, L$ compute and cache only the compacted prompt. Generated tokens are never deleted. Deletion thus reduces prefill computation in every later layer, along with the cache memory and attention cost of each decoding step. Retained tokens keep their original positions, and masks are evaluated in the original position space, so each unpruned pair's causal and local-window visibility is unchanged. During training, soft deletion biases later layers' attention for both prompt and answer queries.

The more fundamental difference lies in what the gate can see. Gemma 3's image tokens attend bidirectionally to one another but cannot attend to a question that follows the image, so the gate in V-MrGemma3 scores visual states contextualized by the image alone, whereas the gate in V-MrT5Gemma2 scores question-conditioned states. This setting therefore tests whether **DTM**'s gate depends on the prompt or on task-trained properties of the visual states themselves (Section 6.1).

4.3 Other Improvements to DTM

Derivative control. Task fine-tuning favors retaining visual information, pushing the deletion rate away from its target. We add a derivative (D) term, which responds to recent changes in the error, to MrT5's proportional (P) and integral (I) terms:

$$\alpha_{t+1} = \max\Big(0,\ \underbrace{k_p \bar{e}_t}_{\text{P}} + \underbrace{k_i \textstyle\sum_{u=1}^{t} e_u}_{\text{I}} + \underbrace{k_d (e_t - e_{t-\omega})}_{\text{D}}\Big), \qquad \bar{e}_t = \rho\, \bar{e}_{t-1} + (1-\rho)\, e_t, \tag{3}$$

where t counts controller updates and ω is a lookback. Thus, when the model is deleting too few tokens and the error is increasing, the derivative term increases α_t more quickly. As in MrT5, $\rho = 0.9$ and the non-negative clamp prevents the regularizer from adding pressure toward retention. We use $k_p = 1$, $k_i = 10^{-5}$, $k_d = 0.5$, $\omega = 10$, and $\alpha_0 = 0.1$ for all backbones. In an ablation across five tasks and four deletion targets, PID kept the observed deletion rate closer to its target than the PI controller in 19 of 20 task–target pairs. This tighter control improved task performance most at the 80% target, where PID outperformed PI on all five tasks by 0.81 points on average; at 20–60%, task-score differences were small and did not consistently favor PID (Appendix A.3).

Threshold-shifted biases for answer queries. After hard deletion, the gate continues to reweight the visual tokens that remain. During input processing (encoder attention in V-MrT5Gemma2 and prompt attention in V-MrGemma3), we retain the original gate bias g_j on visual keys. During answer generation, we instead use $\min(g_j - \tau, 0)$ for visual keys, while text keys receive no gate bias. Thus, visual keys with $g_j \geq \tau$ incur no gate penalty when the model generates its answer, whereas any retained key with $g_j < \tau$, which can occur under fixed-budget deletion, remains downweighted.

Initial deletion-rate calibration. The original **DTM** implementation initializes the gate bias to $b = 1$ (Kallini et al., 2025). We instead apply a one-time calibration of b on the first batch so that a chosen fraction δ_{init} of visual tokens starts below the threshold; we found that this calibration improves accuracy at 60% and 80% targets (Appendix A.2).

Other settings. We adapt the backbone with LoRA rather than MrT5's full fine-tuning, and omit MrT5's attention-score regularizer, which did not improve accuracy in our setting (Appendix A.4). We use $c = -22$ and $\tau = -7$ throughout.

5 Experiments

Models and tasks. We use the encoder–decoder T5Gemma 2 1B–1B and the decoder-only Gemma 3 4B as our primary backbones and T5Gemma 2 270M–270M as a smaller encoder–decoder model. We evaluate on ten benchmarks, covering document and infographic reading (DocVQA; Mathew et al. 2021; InfoVQA; Mathew et al. 2022), chart understanding (ChartQA; Masry et al. 2022), scene-text reading (TextVQA; Singh et al. 2019), general and assistive visual question answering (VQAv2; Goyal et al. 2017; GQA; Hudson & Manning 2019; VizWiz; Gurari et al. 2018), counting (TallyQA; Acharya et al. 2019), object hallucination detection (POPE; Li et al. 2023), and

referring-expression grounding (RefCOCO+; Yu et al. 2016). We report each task's own metric on a 0–100 scale. POPE evaluates checkpoints trained on VQAv2; the other benchmarks use task-specific fine-tuning for each pruning method. Every method, including the unpruned references, is fine-tuned with LoRA on the attention and MLP projections at learning rate 10^{-4}, with the vision encoder frozen; the **DTM** gate is trained jointly with the adapters. Training steps vary by task, so the results are not equalized by training compute; Appendix C lists the evaluation splits, task metrics, training budgets, hyperparameters, and evidence region sources.

Baselines. We compare **DTM** with **random** deletion and two attention-based selectors, **FastV** (Chen et al., 2024b) and **φ-R** from FEATHER (Endo et al., 2025). All methods delete visual tokens after five layers (L5, i.e., $k = 6$) and receive the same task fine-tuning. The adapted **FastV** and **φ-R** learn backbone parameters while their ranking rules remain fixed, whereas **DTM** also learns the ranking function. **Random** deletion controls for adaptation to shorter visual sequences without informed selection, and unpruned fine-tuned backbones serve as *Base* references. Our **FastV** ranks visual keys by received attention averaged over heads and all valid input queries (prompt queries for Gemma 3). Our **φ-R** applies FEATHER's RoPE-free scoring, which counters the positional bias that RoPE's distance-dependent decay induces in **FastV**, with the same query aggregation but without FEATHER's uniform sampling or two-stage procedure (Endo et al., 2025). Appendix D.6 compares all-query scoring with text-query variants on all three backbones, including the criterion of Learn-Pruner's (Takezoe et al., 2026) second stage; all-query scoring wins in most tested comparisons.

Hyperparameter tuning across all models. We fine-tune every method on DocVQA with deletion after each of layers 3–7 (Appendix D.7). L5 is the best layer for **FastV** and **φ-R** on T5Gemma 2 1B–1B, and both drop sharply at L6, the first global-attention layer; **DTM**, which reads hidden states rather than attention weights, leads at every layer on all three backbones at 40–80% deletion. We evaluate all models at L5 on the basis of our T5Gemma 2 layer sweep results.

Two evaluation settings. The two settings correspond to two ways a fine-tuned model can be deployed. In the *matched-rate* setting, a model provider serves several deployment targets, such as device classes or latency tiers, whose compute budgets are fixed in advance, and fine-tunes a separate checkpoint for each; we therefore train separate models toward 20%, 40%, 60%, and 80% visual token deletion and evaluate each at its own training target, measuring the best accuracy each method reaches at a given budget. In the *fixed-checkpoint* setting, a single fine-tuned model serves budgets that are set at inference, for example per request or as load changes; we evaluate each checkpoint at deletion rates from 10% to 90% in 10-point increments, focusing on models trained at 60%. At inference, every method enforces the requested budget by ranking visual tokens and deleting $\lfloor \delta \cdot n_v \rfloor$ of the n_v visual tokens, which for **DTM** separates the budget from the gate's natural threshold-crossing rate (Appendix A.5).

Computational cost. We compute analytical forward-pass FLOPs of the language backbone and vocabulary projection from the recorded sequence lengths before and after deletion, excluding the vision encoder and image projector (Appendix G). Because each method deletes the same number of tokens after the same layer, all four methods share the same backbone cost at a given deletion rate and differ only in how they select tokens, which costs little: the **DTM** gate adds one d-dimensional projection per visual token, **FastV** reuses the preceding layer's attention weights, and **φ-R** adds one RoPE-free score computation in that layer (Appendix G.4). Averaged over the ten benchmarks, deleting 40% and 80% of visual tokens saves 29.4% and 58.4% of FLOPs on T5Gemma 2 1B–1B, 30.9% and 61.4% on Gemma 3 4B, and 27.4% and 53.1% on T5Gemma 2 270M–270M (Table 2; Appendix F.1); savings are smaller on RefCOCO+, whose text prompts are longer.

6 Results and Analysis

6.1 Matched-rate Evaluation Setting

Table 2 compares the methods at 40% and 80% deletion, which save about 30% and 60% of FLOPs. At 40%, **DTM** achieves small gains: V-MrT5Gemma2 leads on five of ten benchmarks and is within 1.1 points of the strongest baseline on every benchmark. On DocVQA, **DTM** leads by 2.5. At 80%, **DTM** leads on all ten benchmarks by 3.76 points on average. **The largest gains under aggressive**

Table 2: **Matched-rate evaluation at 40% and 80% visual token deletion for T5Gemma 2 (1B–1B) and Gemma 3 (4B), deletion after layer 5.** Each score comes from a model trained and evaluated at the given deletion rate δ; Base is the unpruned model fine-tuned in the same way. Bold marks the best method per column within each model; each task uses its own metric (Section 5). The gray row gives the percentage of language-model FLOPs saved, which is the same for every method. Figures 2 and 3 include 60%, and Figures 7–9 (appendix) show 20–80%. **DTM**'s lead grows with the deletion rate on both architectures and is largest on DocVQA, ChartQA, and TextVQA.

T5Gemma 2	DocVQA		ChartQA		InfoVQA		TextVQA		VQAv2		VizWiz		GQA		TallyQA		POPE		RefCOCO+	
δ	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%	40%	80%
Base	82.0		69.7		47.2		72.0		77.5		71.4		71.6		74.9		84.8		79.9	
DTM	**78.4**	**57.5**	66.2	**63.0**	43.0	**34.1**	70.8	**67.8**	**77.3**	**75.6**	69.8	**69.6**	**71.2**	**70.0**	**73.9**	**72.0**	84.7	**84.0**	**79.6**	**75.3**
FastV	75.9	49.5	**67.3**	51.8	**43.2**	31.4	**71.3**	63.3	76.6	72.4	**70.4**	66.9	71.0	66.5	73.2	69.2	**85.2**	82.3	77.9	73.3
φ-R	74.6	47.5	66.5	55.5	42.9	30.7	70.1	58.4	76.3	71.6	70.3	64.0	70.9	67.4	72.4	69.2	84.8	81.9	78.2	71.5
Random	66.8	37.2	65.9	51.0	37.1	27.7	68.9	55.1	76.0	72.4	70.1	67.1	69.9	66.9	71.6	68.7	84.6	82.3	78.7	70.0
FLOPs saved	*29.6*	*58.9*	*29.4*	*58.3*	*29.5*	*58.7*	*30.2*	*60.0*	*30.5*	*60.6*	*30.2*	*59.9*	*30.3*	*60.1*	*30.5*	*60.6*	*30.5*	*60.5*	*23.5*	*46.6*
Gemma 3																				
Base	84.5		76.5		56.7		78.7		78.5		73.7		69.2		73.5		84.5		78.2	
DTM	**80.6**	**58.8**	**77.2**	**68.8**	**51.7**	**40.0**	**77.5**	**75.1**	77.7	**76.7**	72.7	70.8	**69.3**	**68.2**	72.9	**71.6**	**85.4**	**84.0**	**78.3**	**72.7**
FastV	78.1	48.9	74.8	62.2	51.4	36.4	76.6	71.2	**77.8**	73.8	**73.6**	**72.0**	68.9	64.6	73.6	69.9	84.8	80.4	78.2	69.5
φ-R	77.6	47.7	75.2	61.9	49.1	34.2	75.8	69.3	77.6	74.1	73.4	70.3	68.8	64.6	**73.7**	70.6	84.2	81.4	77.7	69.4
Random	68.9	42.1	70.8	58.4	47.6	34.3	73.3	65.3	77.0	73.6	73.0	69.7	68.4	64.4	72.7	69.6	83.8	81.7	76.1	69.6
FLOPs saved	*31.1*	*62.0*	*30.8*	*61.3*	*31.0*	*61.7*	*31.7*	*63.0*	*32.0*	*63.7*	*31.6*	*63.0*	*31.7*	*63.1*	*32.0*	*63.6*	*31.9*	*63.6*	*24.8*	*49.5*

compression occur on tasks that require reading fine visual detail: V-MrT5Gemma2 scores 57.5 on DocVQA, 63.0 on ChartQA, and 67.8 on TextVQA, 8.0, 7.5, and 4.5 points above the strongest baseline. From 40% to 80%, **DTM** loses only 3.3 points on ChartQA and 3.0 on TextVQA, whereas **FastV** loses 15.5 and 7.9. On VQAv2, VizWiz, GQA, TallyQA, and POPE, the attention-based selectors are at most 0.6 points above **random** at 80%, consistent with Wen et al. (2025), while **DTM** leads by 1.7–3.2.

The same pattern holds for the decoder-only V-MrGemma3, whose gate cannot see the question (Table 2 bottom). It leads on seven of ten benchmarks at 40%, trailing by at most 0.9 points, and on nine at 80%, where it averages 3.55 points above the strongest baseline and exceeds **FastV** by 9.9, 6.6, and 3.9 points on DocVQA, ChartQA, and TextVQA; **FastV** remains stronger on VizWiz (72.0 vs. 70.8). The two instances differ in where deletion acts, in a bidirectional encoder whose visual states have read the question or in a causal prefill whose visual states have not, yet **DTM**'s advantage has the same shape in both, consistent with the gate not requiring question-conditioned states. At 60%, **DTM** leads on all ten T5Gemma 2 benchmarks and six Gemma 3 benchmarks (Figures 2 and 3); at 20%, it is within 1.9 points of the strongest baseline on both backbones (Appendix D.1). On T5Gemma 2 270M–270M, the advantage emerges later: **DTM** leads on three, seven, and eight benchmarks at 40%, 60%, and 80% (Appendix D.3). Nonetheless, these results demonstrate that **DTM is the best choice among the evaluated methods for most tasks when a targeted computation budget is known and fixed.**

6.2 Fixed-checkpoint Evaluation Setting

Figure 2 shows that **DTM** retains its comparative advantage when the inference budget becomes tighter than the training target. For T5Gemma 2 1B–1B, **DTM** leads all baselines on every benchmark at each tested deletion rate from 60% through 90%. At 80% inference deletion, the 60%-trained models score 56.5 on DocVQA, 61.4 on ChartQA, and 67.2 on TextVQA, compared with **FastV**'s 46.5, 53.1, and 62.6. This advantage extends to Gemma 3, where **DTM** leads on every benchmark at 80% and 90% deletion (Figure 3). On ChartQA at 90%, it scores 59.0, compared with 46.6 for **random**, the strongest baseline, and 31.8 for **FastV**. The 270M–270M model also leads on all ten benchmarks at 80% (Appendix D.3). Across training targets (Appendix D.2), the decoder-only model transfers best: a Gemma 3 **DTM** checkpoint trained at only 20% loses 2.5 points on average at 80% deletion, compared with the checkpoint trained at 80%, against 5.4–7.1 for the baselines, whereas on T5Gemma 2 1B–1B, **DTM**'s loss (7.4) is similar to the attention-based baselines (5.6–7.4).

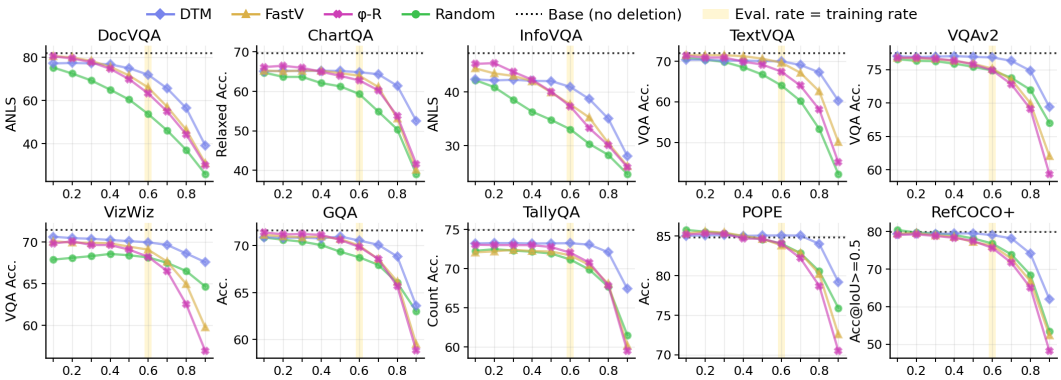


Figure 2: **One 60%-trained T5Gemma 2 1B–1B checkpoint (V-MrT5Gemma2) across inference budgets, deletion after layer 5.** Each curve holds one task- and method-specific checkpoint fixed and evaluates it at deletion rates from 10% to 90%; the training rate, 60%, is highlighted. DTM's advantage widens as the inference budget tightens beyond the training target.

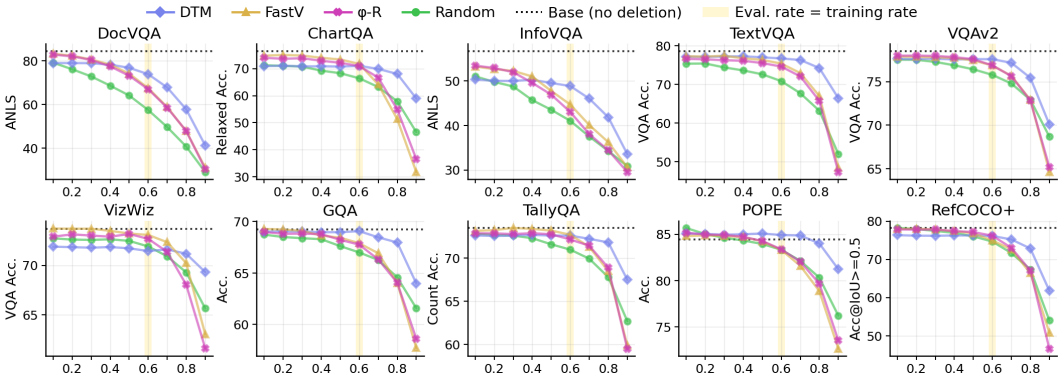


Figure 3: **One 60%-trained Gemma 3 4B checkpoint (V-MrGemma3) decoder-only across inference budgets, deletion after five decoder layers during prefill.** Each curve holds one checkpoint fixed across deletion rates from 10% to 90%, with 60% highlighted. V-MrGemma3 shows the same widening advantage at 80–90% deletion.

Increasing the token budget does not necessarily recover the performance of a model adapted to that budget: on DocVQA, the 60%-trained T5Gemma 2 DTM model scores 77.4 at 20% inference deletion, compared with 81.0 for the model trained at 20%. Conversely, using a 60%-trained checkpoint at 80% deletion costs only 0.6, 0.2, and 0.2 points on average relative to training at 80% for T5Gemma 2 1B–1B, Gemma 3 4B, and T5Gemma 2 270M–270M, respectively (Figure 16, appendix). On Gemma 3 ChartQA, raising the training target from 20% to 80% improves performance at 90% inference deletion from 49.3 to 63.0, but lowers it at 20% deletion from 77.0 to 69.2. A fixed checkpoint is thus useful across budgets, but its training target still shapes the accuracy–cost trade-off; over the full 10–90% range, mean normalized AUC is highest for 60% training in both T5Gemma 2 sizes and 40% in Gemma 3 (Appendix D.4).

To interpret the fixed-checkpoint curves, we identify the largest tested deletion rate that preserves at least 90% of each task's unpruned Base score (Table 12). For both larger backbones trained at 60%, DTM reaches 50% deletion on DocVQA and 70% on ChartQA, compared with FastV's 40% and 60%. On TextVQA, it reaches 80% on both backbones, ten percentage points beyond FastV, and on POPE it meets this criterion even at 90%. However, neither DTM model reaches 90% of Base at any tested rate on InfoVQA, even though both become the strongest methods at high deletion; FastV and φ-R meet that threshold through 30%. **The results show that DTM is the best choice on most tasks when a single fine-tuned checkpoint must generalize to variable inference budgets**, although it can be a worse choice on certain benchmarks when preserving unpruned performance is the priority.

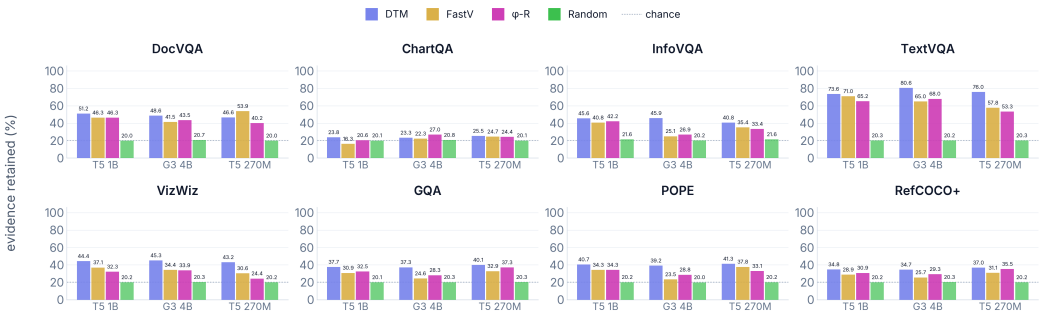


Figure 4: **Spatial evidence retention at matched 80% visual token deletion after layer 5.** Bars show the mean percentage of evidence patches retained on the same eligible questions for each model–dataset pair; dashed lines mark the expected retention of random deletion. T5 1B, T5 270M, and G3 4B denote T5Gemma 2 1B–1B, T5Gemma 2 270M–270M, and Gemma 3 4B. **DTM** retains the most evidence in 22 of 24 comparisons.

6.3 Spatial Evidence Retention

We measure how much spatial evidence each method retains—the fraction of image patches overlapping task-relevant regions, identified from spatial annotations or grounding proxies (Appendix E.1)—and report results on the subset of questions for which such evidence regions are available. Figure 4 shows that **DTM** achieves the highest mean evidence retention in 22 of the 24 model–dataset comparisons at 80% deletion. Across 20–80% deletion, absolute retention decreases as deletion increases while **DTM**'s advantage grows: weighting model–dataset pairs equally, its lead over **FastV** increases from 2.8 to 4.6, 6.7, and 7.7 percentage points, and its lead over **φ-R** from 2.2 to 7.7 points. However, spatial retention does not fully predict task performance. At 80% deletion on Gemma 3 4B ChartQA, **φ-R** retains more evidence than **DTM** (27.0% vs. 23.3%) yet achieves lower relaxed accuracy on the question subset (70.6% vs. 80.5%); on T5Gemma 2 270M–270M DocVQA, **FastV** retains more evidence (53.9% vs. 46.6%) but obtains lower ANLS (41.2 vs. 41.9). Consistent with **DTM**'s premise, retained tokens may already carry context from deleted ones, so spatial retention diagnoses token selection without establishing which information the model actually uses.

Across 15 randomly sampled ChartQA and 15 TextVQA questions at 80% deletion (Appendix E.3), **DTM** concentrates its budget on chart titles, tick labels, and bar ends, and on the full ChartQA sets it keeps 57–67% of text-label patches, against 39–61% for the attention-based selectors (chance: 20%). This is not a general preference for text, since the attention-based selectors cover more OCR text on DocVQA, but for answer-bearing content (Figure 4), which is associated with correctness: where **DTM** keeps more evidence than **FastV**, it alone answers correctly on 13.5% of questions, against 6.2% for **FastV** alone, although this comparison is correlational (Appendix E.2).

7 Conclusion

MrT5 introduced Dynamic Token Merging to make byte-level language models more efficient. We have argued that **DTM** is more general: a learned delete gate, soft-to-hard deletion, and a controlled deletion objective define a framework for token deletion in transformers, of which MrT5 is one instance. Extended with small modifications to visual tokens in both an encoder–decoder and a decoder-only VLM, these components give **DTM** its strongest gains over attention-based and random deletion when visual token budgets are tight, particularly on document, chart, and scene-text tasks, and a model trained at one deletion rate remains effective across inference budgets. The decoder-only gate succeeds without seeing the question, and **DTM**, whose gate reads hidden states rather than attention weights, keeps more task-relevant evidence and is less sensitive than attention-based selectors to whether the preceding layer uses local or global attention. Its gains are smaller under mild deletion and on some general VQA tasks. We hope that treating **DTM** as a general framework encourages its application to further modalities and architectures, such as video, audio, and multi-image inputs, and its combination with other compression techniques.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods: coding assistants wrote much of the code for our model implementations, baselines, training and evaluation pipelines, and analysis scripts, and helped execute experiments. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims or proofs, or propose hypotheses; the research ideas and analyses were conceived by the authors. Additionally, we used generative AI tools to help rewrite and polish prose, draft and edit parts of the paper, create and edit code for figures and tables, check the analyses reported in the paper and write them up in full, locate research literature, and format references. We have reviewed all AI-assisted work: submitted code was manually inspected by multiple authors, all sources identified by AI were manually reviewed, and all AI-created or edited text was checked by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Training hyperparameters and evaluation settings are described in Section 5 and Appendix C, model configurations in Appendix B, and FLOPs calculations in Appendix G. In the supplementary materials, we have included annotated code for **DTM**, including V-MrT5Gemma2 and V-MrGemma3, and for our **FastV**, **φ-R**, and **random** baselines, as well as dataset preprocessing, training and evaluation setups, task metrics, and FLOPs calculations; comments in the code refer to the corresponding equations and sections of the paper. Upon publication, we will also provide model checkpoints.

REFERENCES

Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 8076–8084, 2019.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Chongyan Chen, Samreen Anjum, and Danna Gurari. Grounding answers for visual questions asked by visually impaired people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19098–19107, June 2022.

Jieneng Chen, Luoxin Ye, Ju He, Zhao-Yang Wang, Daniel Khashabi, and Alan Yuille. Efficient large multi-modal models via visual context compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=5ujp72CiYB.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 19–35, Cham, 2024b. Springer Nature Switzerland. ISBN 978-3-031-73004-7.

Mark Endo, Xiaohan Wang, and Serena Yeung-Levy. Feather the throttle: Revisiting visual token pruning for vision-language model acceleration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22826–22835, October 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi,

Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

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 *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6325–6334, 2017. doi: 10.1109/CVPR.2017.670.

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

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://openreview.net/forum?id=nZeVKeeFYf9.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pp. 6700–6709, 2019. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Hudson_GQA_A_New_Dataset_for_Real-World_Visual_Reasoning_and_Compositional_CVPR_2019_paper.html.

Julie Kallini, Shikhar Murty, Christopher D Manning, Christopher Potts, and Róbert Csordás. Mrt5: Dynamic token merging for efficient byte-level language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYWBMq1L7H.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In Houda Bouamor, Juan Pino, and Kalika Bali

(eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20/.

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 *Computer Vision – ECCV 2014*, pp. 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48.

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 Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2200–2209, January 2021.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1697–1706, January 2022.

Evan Miller. Attention is off by one, Jul 2023. URL https://www.evanmiller.org/attention-is-off-by-one.html.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/747d3443e319a22747fbb873e8b2f9f2-Paper.pdf.

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 (CVPR)*, June 2019.

Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. *arXiv preprint arXiv:2105.05486*, 2021. URL https://arxiv.org/abs/2105.05486.

Rinyoichi Takezoe, Yaqian Li, Zi-Hao Bo, Anzhou Hou, Mo Guang, and Kaiwen Long. Learnpruner: Rethinking attention-based token pruning in vision language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Dxb6gBJHby.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6ruVLB727MC.

Rubèn Tito, Dimosthenis Karatzas, and Ernest Valveny. Hierarchical multimodal transformers for multipage docvqa. *Pattern Recognition*, 144:109834, 2023.

Alexander Vogel, Omar Moured, Yufan Chen, Jiaming Zhang, and Rainer Stiefelhagen. RefChartQA: Grounding visual answer on chart images through instruction tuning. In *International Conference on Document Analysis and Recognition*, pp. 523–537. Springer, 2025.

Zichen Wen, Yifeng Gao, Weijia Li, Conghui He, and Linfeng Zhang. Token pruning in multimodal large language models: Are we solving the right problem? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15537–15549, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.802. URL https://aclanthology.org/2025.findings-acl.802/.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2025. URL https://arxiv.org/abs/2410.17247.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi: 10.1162/tacl_a_00461. URL https://aclanthology.org/2022.tacl-1.17/.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19792–19802, June 2025.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European conference on computer vision*, pp. 69–85. Springer, 2016.

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 (ICCV)*, pp. 11975–11986, October 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Biao Zhang, Paul Suganthan, Gaël Liu, Ilya Philippov, Sahil Dua, Ben Hora, Kat Black, Gus Martins, Omar Sanseviero, Shreya Pathak, Cassidy Hardin, Francesco Visin, Jiageng Zhang, Kathleen Kenealy, Qin Yin, Xiaodan Song, Olivier Lacombe, Armand Joulin, Tris Warkentin, and Adam Roberts. T5Gemma 2: Seeing, reading, and understanding longer. *arXiv preprint arXiv:2512.14856*, 2025a. URL https://arxiv.org/abs/2512.14856.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=80faIPZ67S.

Jiaying Zhu, Yurui Zhu, Xin Lu, Wenrui Yan, Dong Li, Kunlin Liu, Xueyang Fu, and Zheng-Jun Zha. Visionselector: End-to-end learnable visual token compression for efficient multimodal llms. *arXiv preprint arXiv:2510.16598*, 2026. URL https://arxiv.org/abs/2510.16598.

A DTM MECHANISMS

A.1 SOFT AND HARD DELETION

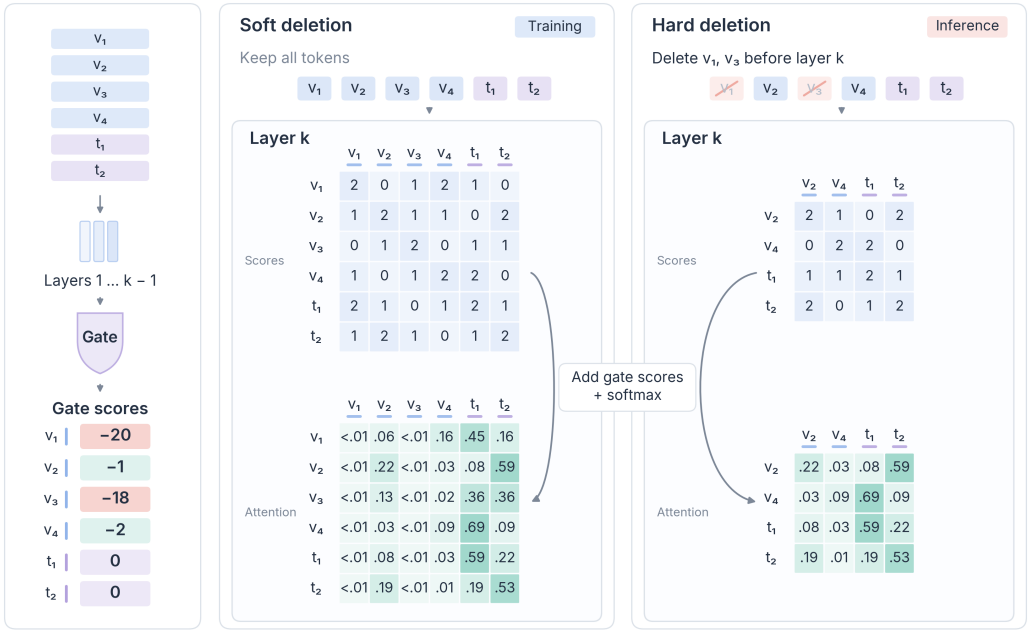


Figure 5: **Soft and hard visual token deletion in DTM.** Vision and text tokens, denoted as s_{v_i} and s_{t_i}, are processed normally from layers 1 through $k-1$. The hidden states before layer k are entered into the deletion gate, producing a score for each token. During hard deletion, hidden states with gate scores less than τ are pruned; the gate scores are applied as biases to only the remaining pre-softmax attention scores. Soft deletion adds gate scores to key columns; hard deletion removes s_{v_1} and s_{v_3} and retains the text tokens.

A.2 INITIAL DELETION-RATE CALIBRATION

Our goal is to initialize the gate so that its measured deletion rate is close to a chosen value $\delta_{\text{init}} \in (0, 1)$. We achieve this by adjusting the scalar bias b while holding the initialized gate weights fixed.

Let $\mathcal{V}_0$ index the visual tokens used for calibration during the first forward pass. For each token $j \in \mathcal{V}_0$, let $\mathbf{x}_j \in \mathbb{R}^d$ be its normalized hidden state and define its scalar projection score as $z_j = \mathbf{w}^\top \mathbf{x}_j$, where $\mathbf{w}$ is the initialized gate weight vector. From Equation 1, the gate score is

$$g_j = c\,\sigma[-(z_j + b)], \qquad c < \tau < 0. \tag{4}$$

The deletion criterion compares this scalar gate score with the threshold: $g_j \leq \tau$.

To determine how the bias affects this criterion, we rearrange it:

$$\begin{aligned} g_j \leq \tau &\iff \sigma[-(z_j + b)] \geq \tau/c \\ &\iff -(z_j + b) \geq \operatorname{logit}(\tau/c) \\ &\iff z_j \leq -\operatorname{logit}(\tau/c) - b, \end{aligned} \tag{5}$$

where $\operatorname{logit}(p) = \log[p/(1-p)]$ is the inverse sigmoid. The first inequality reverses because $c < 0$. Thus, changing b shifts the cutoff applied to the projection scores z_j.

Let q_0 be the empirical δ_{init}-quantile of the projection scores $\{z_j : j \in \mathcal{V}_0\}$, so δ_{init} of these scores are at or below q_0. For example, $\delta_{\text{init}} = 0.6$ selects the 60th percentile. We initialize the bias as

$$b_0 = -\operatorname{logit}(\tau/c) - q_0. \tag{6}$$

With this initialization, $g_j \leq \tau$ exactly when $z_j \leq q_0$, so the deletion criterion is satisfied by the desired fraction of calibration tokens. Calibration is performed only once and sets the starting value of the trainable bias.

Effect of calibration. Table 3 compares calibration with the default constant intercept on DocVQA and VizWiz, at all four training targets and two thresholds ($\tau = -7$ and -11). The constant intercept starts training at 84.4–99.6% deletion regardless of the target, whereas calibration starts within 1.5 points of it in all 16 configurations. The effect on accuracy depends on the target: calibration improves the best validation score by 0.45 and 0.75 points on average at 60% and 80%, but has no consistent effect at 20% and 40% (-0.13 and -0.05). Its clearest benefit is therefore a predictable starting point, with accuracy gains only when the budget is tight.

Table 3: **Calibrated initial deletion rate versus the default constant gate intercept on T5Gemma 2 1B–1B.** Step-0 deletion is the validation deletion rate measured before the first optimizer step; best scores are best in-training validation scores at the trained rate (L5, $c = -22$). The default intercept starts training at 84–99.6% deletion regardless of the target, whereas calibration starts within 1.5 points of the target in all 16 configurations. Calibration improves the best score by 0.45 and 0.75 points on average at 60% and 80% targets but has no consistent effect at 20% and 40% (-0.13 and -0.05).

			Step-0 deletion (%)		Best score		
Task	Target	τ	Calibrated	Default	Calibrated	Default	Δ
DocVQA	20%	-7	21.1	98.5	81.06	81.46	-0.40
DocVQA	20%	-11	21.1	84.4	81.36	81.07	$+0.29$
VizWiz	20%	-7	20.3	99.6	70.24	70.48	-0.24
VizWiz	20%	-11	20.3	93.4	69.86	70.02	-0.16
DocVQA	40%	-7	41.5	98.5	78.25	78.15	$+0.10$
DocVQA	40%	-11	41.5	84.4	77.78	77.50	$+0.28$
VizWiz	40%	-7	40.4	99.6	69.94	70.79	-0.85
VizWiz	40%	-11	40.4	93.4	69.80	69.54	$+0.26$
DocVQA	60%	-7	61.4	98.5	71.74	72.47	-0.72
DocVQA	60%	-11	61.3	84.4	70.00	68.78	$+1.22$
VizWiz	60%	-7	60.8	99.6	69.74	69.51	$+0.23$
VizWiz	60%	-11	61.0	93.4	70.10	69.01	$+1.09$
DocVQA	80%	-7	81.3	98.5	57.90	56.18	$+1.73$
DocVQA	80%	-11	81.2	84.4	52.51	51.62	$+0.89$
VizWiz	80%	-7	81.0	99.6	69.63	69.17	$+0.46$
VizWiz	80%	-11	81.1	93.4	68.90	68.99	-0.09

Target	Mean Δ (calibrated$-$default)	Calibration wins
20%	-0.13	1/4
40%	-0.05	3/4
60%	$+0.45$	3/4
80%	$+0.75$	3/4
All 16	$+0.25$	10/16

A.3 PI VERSUS PID CONTROL

We isolate the derivative term by retraining one **DTM** arm per task and training target with $k_d = 0$ and the derivative disabled, changing nothing else (T5Gemma 2 1B–1B, L5, $c = -22$, $\tau = -7$), for five tasks at all four targets. Table 4 reports the best in-training validation score at the trained deletion rate, along with the controller's tracking error.

PID tracks its target more tightly at every target: it lowers the mean tracking error in 19 of 20 pairs (on the 20th, TextVQA at 40%, the two differ by 0.03 points) and the maximum overshoot in 19; on GQA at 60%, both controllers overshoot early by about 39 points, but PID recovers faster. The accuracy effect depends on the target. At 20–60%, the two controllers are within noise (mean Δ

Table 4: **PI versus PID deletion-rate control on T5Gemma 2 1B–1B at all four training targets.** Each DTM arm is retrained with the derivative term disabled ($k_d = 0$), changing nothing else (L5, $c = -22$, $\tau = -7$). Scores are best in-training validation scores at the trained deletion rate; tracking error $|e_t|$ and overshoot above the target are in percentage points of training-time deletion. The derivative term tightens tracking at every target (lower mean $|e_t|$ in 19 of 20 pairs, one tie) but improves accuracy only at the 80% target, where PID wins on all five tasks by 0.81 points on average.

Task	Target	PID	PI	Δ (PI−PID)	Mean $\lvert e_t\rvert$, PID → PI	Max overshoot, PID → PI
DocVQA	20%	81.06	81.54	+0.48	2.41 → 4.29	8.0 → 23.2
ChartQA	20%	69.38	68.54	−0.83	4.15 → 10.83	19.8 → 32.4
VizWiz	20%	70.24	69.91	−0.33	4.40 → 18.19	17.0 → 34.2
TextVQA	20%	71.37	71.69	+0.32	3.88 → 4.57	10.4 → 26.8
GQA	20%	71.53	71.18	−0.35	2.60 → 4.56	36.3 → 47.7
DocVQA	40%	78.25	78.59	+0.34	3.91 → 7.48	9.0 → 20.2
ChartQA	40%	65.89	67.92	+2.03	2.76 → 4.07	12.9 → 21.8
VizWiz	40%	69.94	69.65	−0.29	6.79 → 13.96	26.6 → 37.1
TextVQA	40%	70.96	71.15	+0.19	3.73 → 3.70	25.5 → 36.4
GQA	40%	71.05	71.31	+0.26	2.48 → 4.44	33.5 → 39.7
DocVQA	60%	71.74	71.77	+0.02	5.32 → 9.24	12.1 → 22.2
ChartQA	60%	64.58	65.21	+0.62	1.94 → 3.11	16.9 → 20.8
VizWiz	60%	69.74	70.21	+0.47	5.23 → 8.11	18.2 → 24.3
TextVQA	60%	69.94	69.94	+0.00	3.00 → 3.08	14.4 → 25.1
GQA	60%	70.32	70.72	+0.40	2.51 → 3.63	39.5 → 39.3
DocVQA	80%	57.90	55.51	−2.39	5.78 → 9.17	5.7 → 14.1
ChartQA	80%	63.07	62.86	−0.21	1.07 → 2.25	4.4 → 15.7
VizWiz	80%	69.63	68.88	−0.75	4.43 → 6.51	8.1 → 11.9
TextVQA	80%	67.89	67.72	−0.17	1.57 → 2.70	5.1 → 9.9
GQA	80%	69.98	69.46	−0.52	1.57 → 2.86	19.5 → 19.9

Target	Mean Δ (PI−PID)	PI wins	PID tighter, mean $\lvert e_t\rvert$	PID tighter, overshoot
20%	−0.14	2/5	5/5	5/5
40%	+0.51	4/5	4/5	5/5
60%	+0.30	5/5	5/5	4/5
80%	−0.81	0/5	5/5	5/5
All 20	−0.03	11/20	19/20	19/20

of -0.14, $+0.51$, and $+0.30$), whereas at 80%, PID wins on all five tasks (mean -0.81), with the largest gain on DocVQA (2.39 ANLS). Early overshoots of 20–48 points under PI aren't significant at 20–60% targets, but at an 80% target even an overshoot of 10–20 points trains the model at 90–100% deletion (about 94% on DocVQA), discarding information that it cannot recover within its training budget.

A.4 Attention-Score Regularization

Because soft deletion suppresses a key only by adding $g_j \geq c$ to its logits, the backbone could in principle learn to raise $\mathbf{q}_i^\top \mathbf{k}_j$ enough to cancel the bias. MrT5 counters this with an attention-score regularizer $\mathcal{L}_S$ (Kallini et al., 2025, App. D). On T5Gemma 2 1B–1B at 80% deletion (L5), we measured this failure directly and tested two defenses: $\mathcal{L}_S$ in MrT5's form ($\beta = 5$), and a logit soft-cap $v \tanh(s/v)$ with $v = 20 < |c|$, which makes cancellation arithmetically impossible. MrT5's knee $m = 5$ lies below the mean attention score of this backbone (6.1), so we set $m = |c| = 22$. Every arm is compared at an exactly enforced 79.7% deletion rate (Table 5).

The failure mode exists: on every task, scores on suppressed keys reach 38–44 logits against a bias of -22, and on DocVQA 3% of suppressed keys end up with a positive post-bias score. However, it matters only where soft deletion already overstates hard deletion, which is substantial only on DocVQA (8.0 points). Both defenses work mechanically: the cap bounds suppressed-key scores

Table 5: **Attention-score regularization for the delete gate.** The first column is the DTM score without regularization (best checkpoint), and the next two are changes relative to it. "Cap 20 without gate" is the cap's effect on a fine-tuned model with no deletion. The soft−hard gap is the score difference between soft and hard deletion without regularization. Arms within a row share a training recipe. Neither defense improves accuracy on average, so we use neither.

Task	No regularizer	$+\mathcal{L}_S$	+cap 20	Cap 20 without gate	Soft−hard gap
DocVQA	51.69	+0.24	+1.41	−1.09	8.03
ChartQA	64.27	−0.89	−2.03	−1.72	0.26
TextVQA	67.24	+0.68	−0.45	−0.63	−0.02
VQAv2	73.17	+0.04	+0.54	−0.37	0.28
Mean Δ	—	+0.02	−0.13	−0.95	—

at 19.3 and reduces leaked attention 3–9×, and a cancellation-targeted variant of $\mathcal{L}_S$ lowers the largest post-bias score on suppressed keys monotonically in β (from 25.3 to 8.1). Neither improves accuracy. MrT5's form of $\mathcal{L}_S$ changes the mean score by only $+0.02$, and the cancellation variant lowers DocVQA ANLS by 1.3, 2.1, and 4.9 points at $\beta = 1, 5, 25$. The cap helps only on DocVQA and costs accuracy on the ungated model on all four tasks. We therefore use neither defense.

A.5 Natural Thresholds and Per-Example Budgets

Table 6: **Natural thresholding versus a fixed per-example budget within the same checkpoint (T5Gemma 2 1B–1B, L5).** Each row uses the main-experiment DTM checkpoint for that task and training target. Natural deletion removes every visual token with $g_j \leq \tau$, so the per-example rate varies (mean and 5th–95th percentiles shown); forced deletion removes the same mean fraction from every example. The two passes' FLOPs differ by at most 0.61%. Metrics follow the main results, except that RefCOCO+ uses mean IoU. Natural thresholding scores higher in 16 of 18 pairs, including all nine at the 60% target, by 0.29 points on average.

Task	Target	Nat. del. (%)	Nat. p_5–p_{95} (%)	Natural	Forced	Δ (Nat.−Forced)
DocVQA	60%	59.98	40.2–82.4	73.22	71.95	+1.26
DocVQA	80%	80.43	68.0–91.8	56.87	57.04	−0.17
ChartQA	60%	61.69	43.8–79.7	65.10	65.05	+0.05
ChartQA	80%	80.47	67.6–90.6	63.18	62.40	+0.78
TextVQA	60%	63.93	31.2–88.3	70.10	69.79	+0.31
TextVQA	80%	77.64	53.5–94.5	68.60	67.92	+0.68
VizWiz	60%	59.84	27.7–91.8	70.11	69.96	+0.15
VizWiz	80%	80.32	58.2–97.7	69.64	69.73	−0.09
GQA	60%	60.79	43.8–78.1	70.69	70.58	+0.11
GQA	80%	79.91	71.5–87.5	70.09	69.85	+0.23
TallyQA	60%	67.10	48.8–82.8	73.38	73.17	+0.21
TallyQA	80%	83.31	70.7–93.4	72.27	71.82	+0.45
RefCOCO+	60%	60.47	46.9–75.0	70.39	70.06	+0.33
RefCOCO+	80%	79.46	71.5–87.1	65.55	65.48	+0.08
VQAv2	60%	59.93	44.9–75.0	77.26	77.00	+0.26
VQAv2	80%	79.40	70.3–87.5	76.31	76.07	+0.24
POPE	60%	57.50	42.6–72.7	85.19	85.11	+0.08
POPE	80%	79.80	71.5–87.1	84.56	84.31	+0.24

Our evaluations enforce an exact budget for every example, which separates the budget from the gate's natural threshold-crossing rate (Section 5). Table 6 compares two inference policies within the same main-experiment checkpoint in 18 T5Gemma 2 1B–1B task/target pairs: natural deletion of every visual token with $g_j \leq \tau$, and exact-count deletion of that policy's measured mean rate from every example. Natural deletion scores higher in 16 of 18 comparisons, including all nine at the 60% target, by 0.29 points on average, with the largest gain on DocVQA at 60% (+1.26 ANLS); the two exceptions are DocVQA and VizWiz at 80%. The natural policy allocates budgets unevenly: on VizWiz at the 60% target, per-example deletion ranges from 27.7% to 91.8% between the 5th and

95th percentiles. Natural thresholding therefore has a small measured advantage over a fixed per-example budget, but the comparison also changes the remaining tokens' weighting (naturally kept tokens receive no answer-query bias; Section 4.3), so it does not isolate the effect of token allocation. We report fixed budgets in the main results so that every method removes the same number of tokens from every example.

B Model Architectures

B.1 Model Sizes and Configurations

We use T5Gemma 2 1B–1B, T5Gemma 2 270M–270M, and Gemma 3 4B. Table 7 lists the configurations of the backbones used in our implementation. The T5Gemma 2 encoder and decoder are initialized from the pretrained Gemma 3 1B and 270M models, and subsequently adapted with UL2 (Zhang et al., 2025a). T5Gemma 2 shares its encoder input, decoder input, and vocabulary-projection weights. Gemma 3 also ties its input embeddings and vocabulary projection.

Table 7: **Configurations of the three model backbones.** T5Gemma 2 dimensions and head counts apply to both language stacks. Language parameter counts include the shared text embeddings once; totals additionally include the vision encoder and image projector. All three backbones receive the same 256 visual tokens per image, so a deletion rate removes the same number of visual tokens in each.

	T5Gemma 2		Gemma 3
Configuration	1B–1B	270M–270M	4B
Encoder / decoder blocks	26 / 26	18 / 18	— / 34
Hidden width d	1,152	640	2,560
Feed-forward width d_{ff}	6,912	2,048	10,240
Query heads	4	4	8
Key/value heads	1	1	4
Head width d_h	256	256	256
Vocabulary size	262,144	262,144	262,208
Local-attention window	512	512	1,024
Visual tokens per image	256	256	256
Language parameters (billions)	1.698	0.368	3.880
Total parameters (billions)	2.116	0.786	4.300

All three models use grouped-query attention, query/key normalization, rotary positional embeddings, and gated GELU feed-forward sublayers. Each attention and feed-forward sublayer applies RMSNorm (Zhang & Sennrich, 2019) before the operation and again before residual addition. For a sublayer function f, this order is

$$\mathbf{y} = \mathbf{x} + \mathrm{RMSNorm}_{\text{post}}(f(\mathrm{RMSNorm}_{\text{pre}}(\mathbf{x}))) \,. \tag{7}$$

The pre- and post-normalizations have separate parameters, and a final RMSNorm follows each complete language stack. The single key/value head in each T5Gemma 2 model is the multi-query special case of grouped-query attention. Figure 6 illustrates both backbone architectures and the normalization and residual connections within each block.

B.2 Vision Processing

All three models use the same approximately 417-million-parameter SigLIP (Zhai et al., 2023) vision encoder. Each 896×896 image is divided into 14×14 patches, producing a 64×64 grid of 4,096 patch representations. Spatial average pooling reduces the grid to 16×16, yielding 256 visual tokens per image. These representations are passed through a linear projection into the language model's hidden width. Visual tokens are inserted in the encoder input for T5Gemma 2 and in the decoder prompt for Gemma 3.

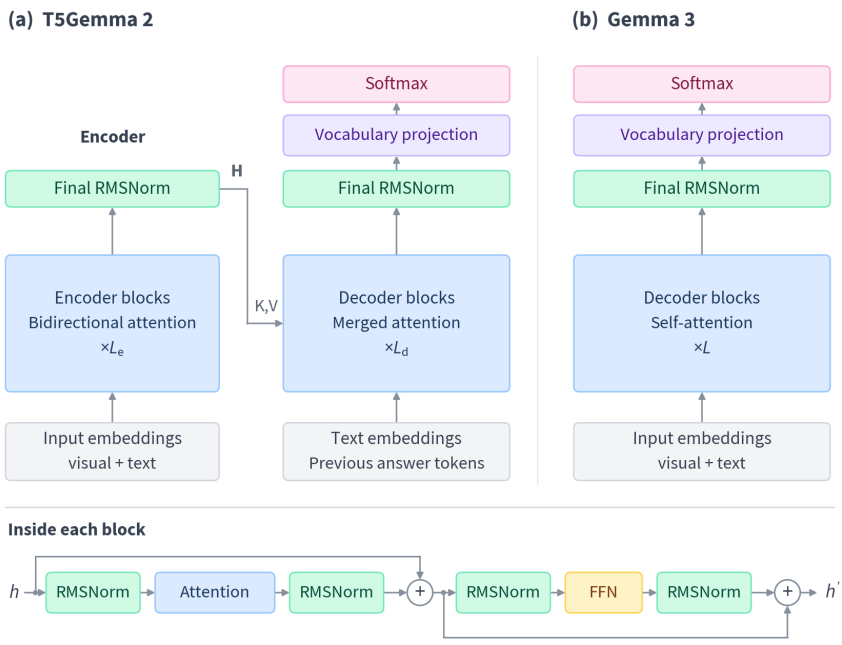


Figure 6: **T5Gemma 2 and Gemma 3 language-backbone architectures.** Visual tokens enter after SigLIP encoding, pooling, and image projection. (a) T5Gemma 2 uses a bidirectional encoder and a decoder with merged self- and cross-attention. Each decoder block projects the final normalized encoder states $\mathbf{H}$ into its own keys and values. (b) Gemma 3 uses a decoder-only backbone. Here L_{e} and L_{d} denote encoder and decoder depths: both are 26 for T5Gemma 2 1B–1B and 18 for 270M–270M; Gemma 3 4B has $L = 34$ decoder blocks.

B.3 Attention Visibility and Layer Indexing

As in Section 3.1, layer 0 denotes the input states and transformer layers are indexed from one. The global-attention layers are $6, 12, 18, 24$ in each T5Gemma 2 1B–1B stack, $6, 12, 18$ in each 270M–270M stack, and $6, 12, 18, 24, 30$ in Gemma 3 4B. All remaining layers use local attention, including the final layers when the stack ends before the next global layer.

T5Gemma 2's encoder uses a 512-token bidirectional local window. In our implementation, this includes up to 255 preceding positions, the current position, and 256 following positions. Its decoder uses a causal 512-token window over decoder keys in local layers; all valid encoder keys remain visible through merged attention in every decoder layer. Gemma 3 uses a causal 1,024-token local window for text queries, including the current position, and permits bidirectional attention within each image block. Thus, image-prefix states cannot attend to a later text question, although image patches can attend to one another.

As in Figure 1, k denotes the first layer that processes the shortened sequence, so layers $1, \ldots, k-1$ process the full sequence. We name each configuration by the number of full-sequence layers: our main setting, L5, deletes after five layers and before the sixth ($k = 6$). Attention-based selectors use scores from layer $k - 1$; consequently, L5 reads the local-attention layer 5, whereas L6 reads the first global-attention layer, layer 6.

Merged attention. Let $\mathbf{H} \in \mathbb{R}^{n \times d}$ and $\mathbf{X} \in \mathbb{R}^{m \times d}$ contain the final encoder states and the decoder states of T5Gemma 2, where n, m are their sequence lengths. For one query head and its associated key/value head, the learned projections $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d_h}$ and $\mathbf{W}_o \in \mathbb{R}^{d_h \times d}$ give

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = [\mathbf{X}; \mathbf{H}]\mathbf{W}_k, \quad \mathbf{V} = [\mathbf{X}; \mathbf{H}]\mathbf{W}_v, \quad \mathbf{O} = \mathbf{A}\mathbf{V}\mathbf{W}_o, \tag{8}$$

so $\mathbf{Q} \in \mathbb{R}^{m \times d_h}$, $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{(m+n) \times d_h}$, and the attention weights $\mathbf{A}$ have shape $m \times (m + n)$. We omit head indices, query/key normalization, and positional transforms. Decoder queries jointly

attend to visible decoder keys and all valid encoder keys, with causality applied to the decoder-key block. Under **DTM**, hard deletion removes rows of $\mathbf{H}$, shortening the encoder block of $\mathbf{K}$ and $\mathbf{V}$ in every decoder layer.

C Experimental Details

Table 8 lists each benchmark's metric, the number of evaluated examples, and the training steps; Table 9 lists the shared adaptation and gate recipe. All evaluations use zero-shot prompts. To keep splits consistent, we evaluate on validation splits wherever one exists: DocVQA, InfoVQA, TextVQA, and VQAv2 withhold their test answers for server-side scoring, and the evidence annotations for TextVQA, VizWiz, and GQA are released only for training and validation images. TallyQA and POPE have no validation split, so we use their test splits. DocVQA and InfoVQA use average normalized Levenshtein similarity (ANLS), ChartQA uses relaxed accuracy, TextVQA, VQAv2, and VizWiz use their VQA answer-scoring rules, and TallyQA uses exact-count accuracy; RefCOCO+ counts a predicted box as correct when its intersection-over-union with the reference is at least 0.5.

Table 8: **Evaluation populations and training steps (T5Gemma 2 1B–1B).** Every method and deletion rate is evaluated on the same examples. VQAv2 and GQA use fixed, uniformly random subsets of their validation splits (25,600 of 214,354 and 12,800 of 132,062 questions); every other benchmark uses its full split. POPE evaluates the VQAv2-trained checkpoints. Budgets differ across tasks, so we compare methods within each task rather than at equal training compute. The last column gives the source of the evidence regions in Figures 4 and 24, which are computed on the evaluated examples that have at least one evidence region ("With evidence"); OCR-word regions are the words that spell a gold answer.

Task	Split / metric	Evaluated examples	With evidence	Training steps (deletion methods)	Evidence regions
DocVQA	val. / ANLS	5,349	5,109	300	OCR words (Tito et al., 2023)
ChartQA	val. / relaxed acc.	1,920	1,142	1,250	Support boxes (Vogel et al., 2025)
InfoVQA	val. / ANLS	2,801	1,916	300	OCR words (Mathew et al., 2022)
TextVQA	val. / VQA acc.	5,000	4,239	1,250	OCR words (Singh et al., 2021)
VQAv2	val. / VQA acc.	25,600	—	1,500	—
VizWiz	val. / VQA acc.	4,319	1,131	160	Grounding polygons (Chen et al., 2022)
GQA	val. / accuracy	12,800	12,005	2,000	Scene-graph boxes (Hudson & Manning, 2019)
TallyQA	test / count acc.	38,589	—	1,250	—
POPE	test / accuracy	9,000	4,016	VQAv2	Object boxes (Lin et al., 2014)
RefCOCO+	val. / Acc@IoU$\geq$0.5	10,758	10,758	1,500	Referent box (Yu et al., 2016)

Table 9: **Shared adaptation and gate recipe.** Apart from batch size, every listed setting is identical across the three backbones and all plotted arms.

Setting	Value
Adapter learning rate	10^{-4}
Gate learning rate	10^{-4} (separate parameter group)
Learning-rate schedule	Cosine for adapters and gate; no warmup
Weight decay	10^{-4}
Batch size	256 (Gemma 3 4B: 128)
Gradient clipping	Maximum gradient norm 1
LoRA rank / scaling / dropout	32 / 64 / 0.05
LoRA targets	Attention Q/K/V/O and all MLP projections
Vision encoder	Frozen (no LoRA)
Precision	BF16 backbone
Gate scale / deletion threshold	$c = -22$ / $\tau = -7$
Controller P / I / D gains	$k_p = 1$ / $k_i = 10^{-5}$ / $k_d = 0.5$
Derivative lookback ω	10 controller updates
Gate training mode	Soft deletion throughout training
Inference budget	Exactly $\lfloor \delta n_v \rfloor$ visual tokens deleted per example

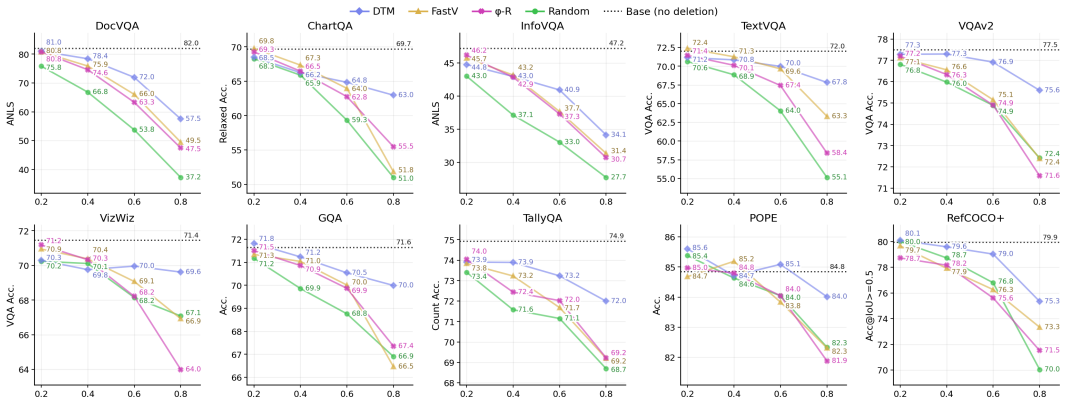


Figure 7: **Matched-rate evaluation on T5Gemma 2 1B–1B (V-MrT5Gemma2), deletion after layer 5.** Each point uses a model trained and evaluated at the deletion rate δ on the horizontal axis. **DTM** leads on all ten benchmarks at 60% and 80% deletion; at 20–40%, the methods are close.

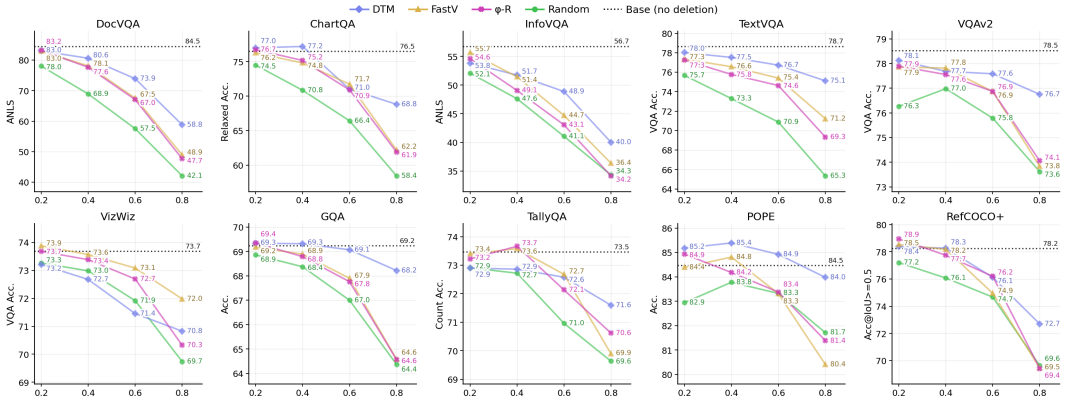


Figure 8: **Matched-rate evaluation on Gemma 3 4B (V-MrGemma3), deletion after five decoder layers during prompt prefill.** Each point uses a model trained and evaluated at the deletion rate δ on the horizontal axis. **DTM** leads on nine benchmarks at 80% and six at 60%; at 20–40%, the methods are close.

D Additional Results and Analysis

D.1 Matched-Rate Results on All Three Models

Figures 7, 8, and 9 show every matched-rate result behind Table 2, from 20% to 80% deletion, for T5Gemma 2 1B–1B, Gemma 3 4B, and T5Gemma 2 270M–270M. The three backbones share one pattern. At 20% deletion, **DTM** leads on 5, 4, and 1 of the ten benchmarks, respectively, and stays within 1.9 points of the strongest baseline on the two larger models; at 80%, it leads on 10, 9, and 8, with mean margins of 3.8, 3.6, and 1.9 points. Section 6.1 discusses the two primary backbones, and Appendix D.3 the smaller one.

D.2 Fixed Checkpoints at Every Training Rate

Figures 10–15 repeat the fixed-checkpoint evaluation of Figures 2 and 3 for checkpoints trained at 20%, 40%, 60%, and 80% deletion on all three backbones, comparing methods trained at the same target. Three patterns hold across training targets. First, **DTM**'s advantage appears when the inference budget is tight: at 80% inference deletion, **DTM** leads the strongest baseline on 5, 9, 10, and 10 of the T5Gemma 2 1B–1B benchmarks for training targets of 20%, 40%, 60%, and 80%, and on 10, 10, 10, and 9 of the Gemma 3 benchmarks. Second, under mild inference deletion (20%), the methods are close, and **DTM** leads on at most four benchmarks for any backbone and training

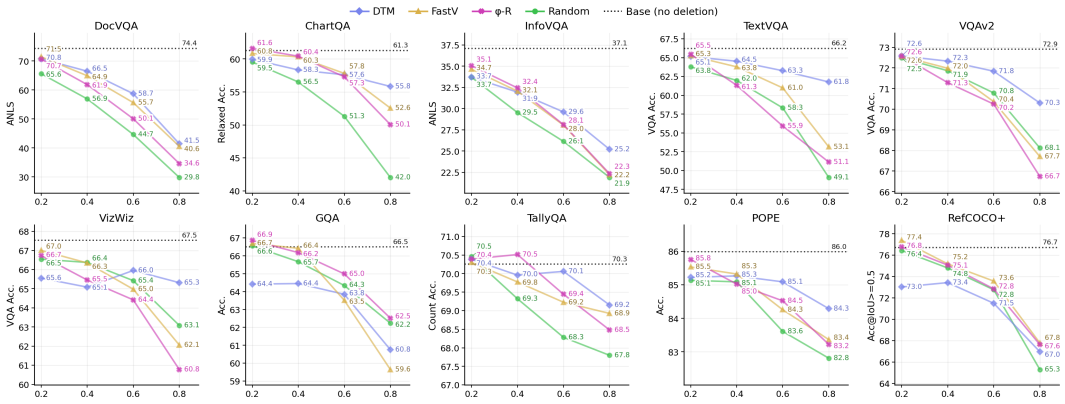


Figure 9: **Matched-rate evaluation on T5Gemma 2 270M–270M, deletion after layer 5.** Each point uses a model trained and evaluated at the deletion rate δ on the horizontal axis. The advantage emerges at higher deletion rates on the smallest backbone: **DTM** leads on 1, 3, 7, and 8 benchmarks at 20%, 40%, 60%, and 80%.

target. Third, the decoder-only Gemma 3 transfers best to tighter budgets: its **DTM** checkpoints trained at only 20% still lead on all ten benchmarks at 80% and 90% deletion, by 6.9 and 8.1 points on average. Used at 80% deletion, the 20%-trained Gemma 3 **DTM** checkpoint loses 2.5 points on average relative to training at 80%, against 5.4–7.1 for the baselines; on T5Gemma 2 1B–1B, **DTM**'s loss (7.4) is similar to that of the attention-based selectors (5.6–7.4) (Figure 16). The training target still shapes the trade-off (Appendices D.4 and D.5).

D.3 T5Gemma 2 270M–270M

We apply the same L5 deletion procedure to the smaller T5Gemma 2 backbone (Figures 9, 14, and 15). Across the ten benchmarks, **DTM**'s mean margin over the strongest baseline increases from 0.62 score points at matched 60% deletion to 1.87 at 80%. At 80%, it scores 41.47 ANLS on DocVQA vs. **FastV**'s 40.58, and 55.83 relaxed accuracy on ChartQA vs. 52.55. The gains are task-dependent. TextVQA improves by 8.63 points over **FastV**, while GQA trails **φ-R** by 1.76 and RefCOCO+ trails **FastV** by 0.82 (66.97 vs. 67.79). At 80% inference deletion, a 60%-trained checkpoint outperforms the strongest baseline on each of the ten tasks. This backbone also benefits most from deleting later (Appendix D.8).

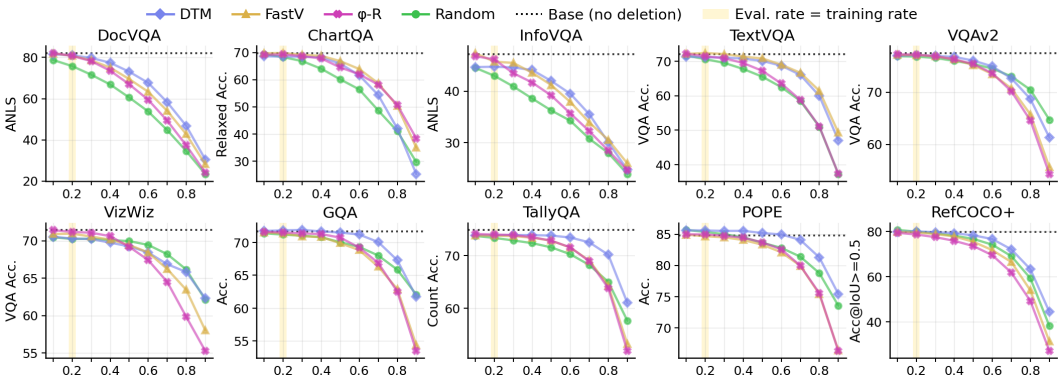
D.4 Fixed-Checkpoint Budget Summaries

For a fixed checkpoint, we summarize scores s_i at common inference deletion rates $r_i \in \{0.1, 0.2, \ldots, 0.9\}$ by normalized trapezoidal area:

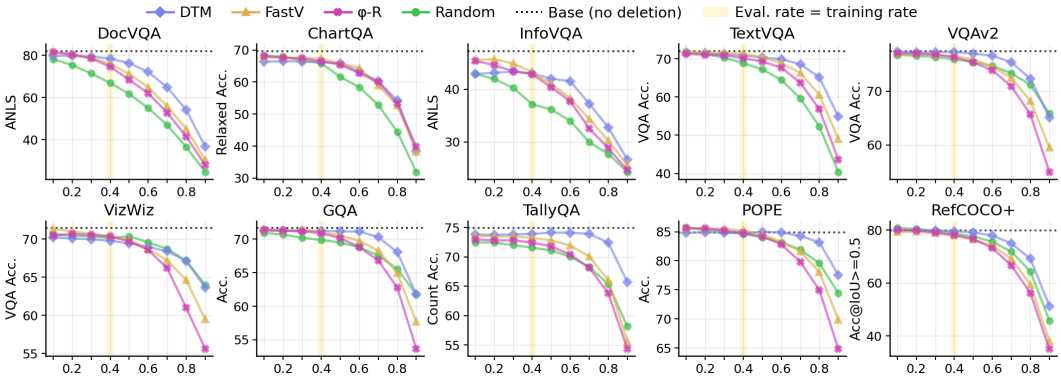
$$\overline{s} = \frac{1}{0.9 - 0.1} \sum_{i=1}^{8} \frac{s_i + s_{i+1}}{2} (r_{i+1} - r_i). \tag{9}$$

This averages performance over the deletion rate axis. Table 10 reports each benchmark separately because the metrics differ: **DTM** has the largest area on all ten T5Gemma 2 1B–1B benchmarks, nine Gemma 3 benchmarks, and eight T5Gemma 2 270M–270M benchmarks. Table 11 repeats the ten-benchmark mean for each training target: **DTM**'s mean area peaks at 60% training on both T5Gemma 2 sizes and at 40% on Gemma 3, and it is the highest of the four methods at every training target on all three backbones.

We also report the largest tested inference deletion rate whose score is at least 0.9 times the separately trained unpruned Base model in Table 12. "None" means that no tested rate meets the criterion, not that no possible configuration could meet it.



(a) Trained at 20%

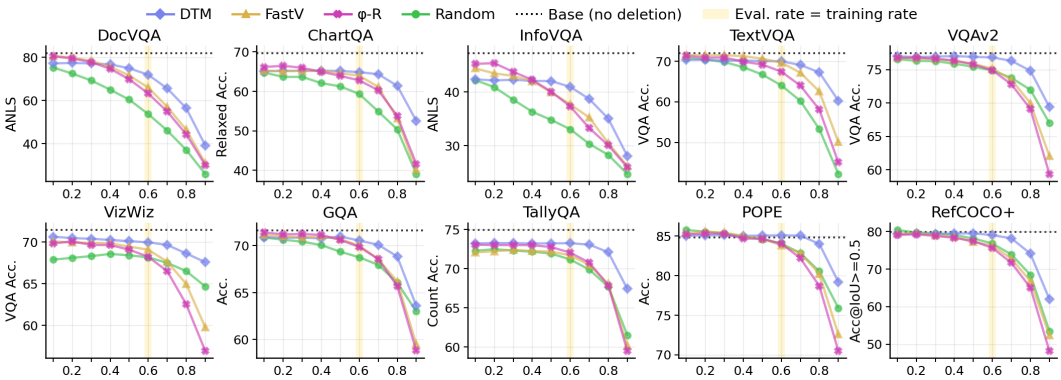


(b) Trained at 40%

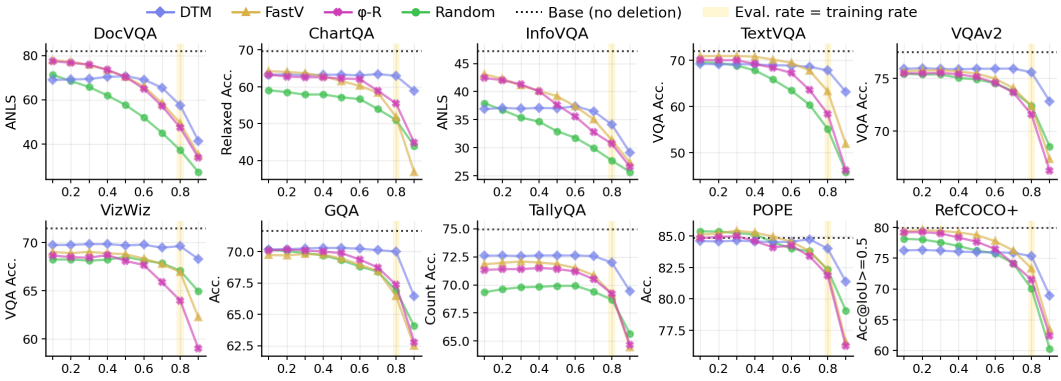
Figure 10: **Fixed checkpoints on T5Gemma 2 1B–1B trained at 20% (a) and 40% (b) deletion.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: **φ-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. Trained at 40%, **DTM** leads on nine benchmarks at 80% inference deletion; checkpoints trained at only 20% transfer less well, and **DTM** leads on half.

Table 10: **Normalized area over inference deletion rates 10–90%.** All checkpoints are trained at 60% deletion and evaluated at nine deletion rates; the area is the trapezoidal integral divided by the 0.8-wide interval, so values remain on each task's 0–100 metric scale. The final row weights tasks equally. Bold marks the strongest method within each model. **DTM** has the highest area on 10 of 10 T5Gemma 2 1B–1B benchmarks, 9 of 10 Gemma 3 4B benchmarks and 8 of 10 T5Gemma 2 270M–270M benchmarks, and the highest equal-weight mean on all three backbones.

	T5Gemma 2 1B–1B				Gemma 3 4B				T5Gemma 2 270M–270M			
Task	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**
DocVQA	**69.82**	66.39	64.98	56.88	**71.59**	68.35	67.93	60.51	**58.31**	56.60	52.81	47.61
ChartQA	**63.76**	61.37	61.49	58.39	**69.65**	67.19	67.52	65.70	**57.15**	57.01	56.20	51.30
InfoVQA	**39.78**	38.37	38.44	34.44	**47.33**	45.85	44.81	42.71	**29.41**	28.12	28.93	27.13
TextVQA	**69.03**	68.12	66.07	63.68	**75.98**	73.25	72.37	70.21	**62.07**	59.51	55.94	58.18
VQAv2	**76.12**	74.20	73.81	74.54	**76.80**	75.94	76.04	75.59	**70.95**	69.55	69.54	70.47
VizWiz	**69.82**	68.21	67.38	67.72	71.51	**72.27**	71.45	71.45	**65.74**	64.20	64.15	64.97
GQA	**70.02**	69.20	69.18	68.77	**68.50**	67.22	67.08	66.97	63.36	63.01	**64.52**	64.12
TallyQA	**72.69**	70.66	71.07	70.55	**72.12**	71.52	71.22	70.69	**69.15**	68.53	68.94	67.59
POPE	**84.58**	83.32	82.85	83.59	**84.63**	82.64	82.92	83.08	**84.41**	83.85	83.96	83.32
RefCOCO+	**77.47**	74.48	73.72	75.30	**74.77**	73.55	73.58	73.52	70.30	71.43	**71.53**	71.33
Mean benchmark score	**69.31**	67.43	66.90	65.39	**71.29**	69.78	69.49	68.04	**63.08**	62.18	61.65	60.60



(a) Trained at 60%

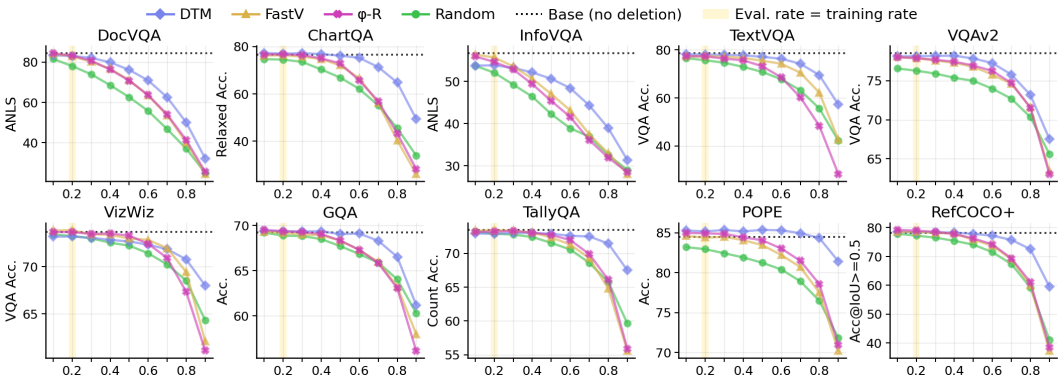


(b) Trained at 80%

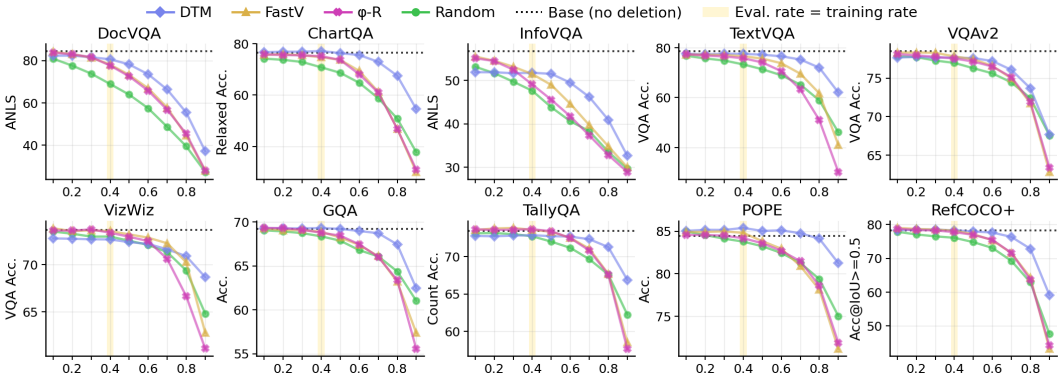
Figure 11: **Fixed checkpoints on T5Gemma 2 1B–1B trained at 60% (a) and 80% (b) deletion.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: **φ-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. Trained at 60% or 80%, **DTM** leads on all ten benchmarks at 80% and 90% inference deletion.

Table 11: **Mean normalized AUC over 10–90% inference deletion, by training target.** Each mean weights the ten benchmarks equally on their 0–100 score scales. Bold marks the strongest training target for each method within each backbone. The final row counts tasks where the 60%-trained method exceeds all three other methods in AUC. **DTM**'s mean area peaks at 60% training on T5Gemma 2 1B–1B, 40% on Gemma 3 4B and 60% on T5Gemma 2 270M–270M, and **DTM** has the highest mean area at every training target on all three backbones.

	T5Gemma 2 1B–1B				Gemma 3 4B				T5Gemma 2 270M–270M			
Training rate	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**
20%	67.67	66.50	65.25	64.93	71.58	68.40	67.99	66.47	62.01	61.17	60.86	60.07
40%	68.84	67.13	66.04	**65.40**	**71.92**	69.32	68.60	67.61	62.69	61.95	61.08	**60.65**
60%	**69.31**	**67.43**	**66.90**	65.39	71.29	**69.78**	**69.49**	**68.04**	**63.08**	**62.18**	**61.65**	60.60
80%	68.09	67.39	66.77	64.54	70.32	69.28	69.13	67.30	62.33	61.51	61.06	59.52
Tasks led at 60% training	10/10	0/10	0/10	0/10	9/10	1/10	0/10	0/10	8/10	0/10	2/10	0/10



(a) Trained at 20%

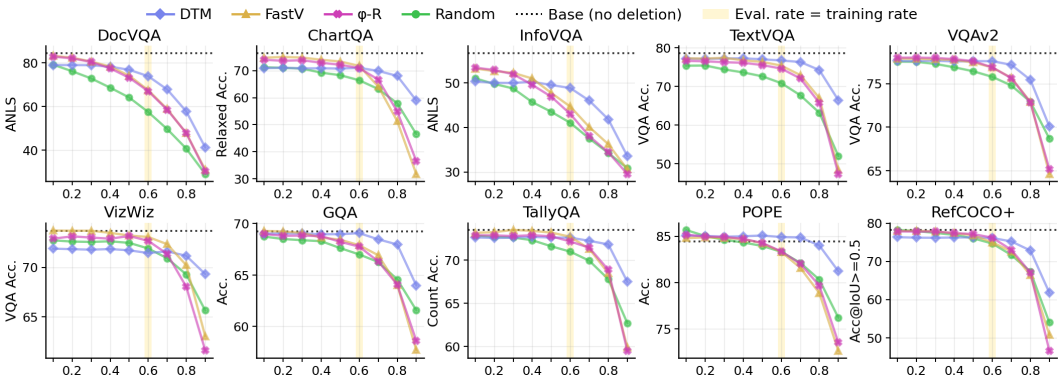


(b) Trained at 40%

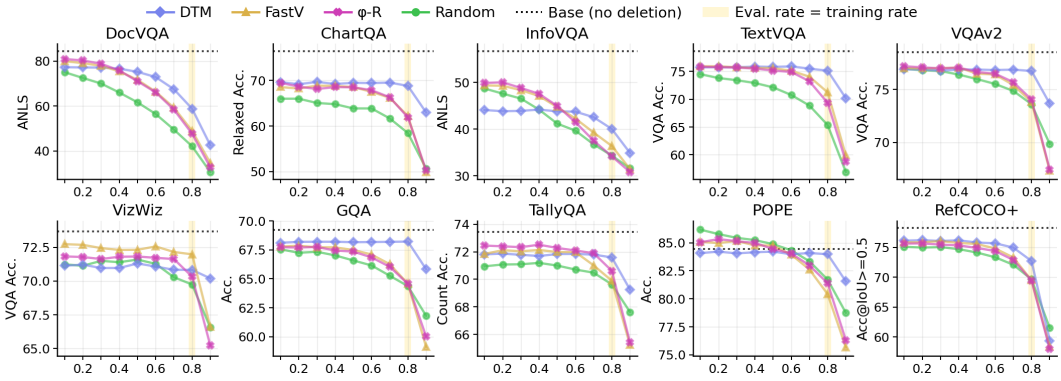
Figure 12: **Fixed checkpoints on Gemma 3 4B trained at 20% (a) and 40% (b) deletion, deletion during prefill.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: **φ-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. Even **DTM** checkpoints trained at 20% or 40% lead on all ten benchmarks at 80% and 90% inference deletion.

Table 12: **Largest tested deletion rate retaining at least 90% of Base.** All checkpoints are trained at 60% deletion and evaluated at 10–90% deletion in 10-point increments; Base is the separately fine-tuned unpruned model. None indicates that no tested rate meets this threshold. **DTM** reaches the largest budget (ties included) on 9 of 10, 9 of 10, 10 of 10 benchmarks for T5Gemma 2 1B–1B, Gemma 3 4B and T5Gemma 2 270M–270M, respectively, but no **DTM** checkpoint preserves 90% of Base on InfoVQA.

	T5Gemma 2 1B–1B				Gemma 3 4B				T5Gemma 2 270M–270M			
Task	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**	**DTM**	**FastV**	**φ-R**	**Random**
DocVQA	50%	40%	40%	10%	50%	40%	40%	20%	30%	20%	20%	None
ChartQA	70%	60%	60%	30%	70%	60%	60%	40%	80%	70%	60%	40%
InfoVQA	None	30%	30%	None	None	30%	30%	None	None	None	None	None
TextVQA	80%	70%	60%	50%	80%	70%	70%	60%	70%	60%	40%	50%
VQAv2	80%	80%	70%	80%	80%	80%	80%	80%	80%	80%	70%	80%
VizWiz	90%	80%	70%	90%	90%	80%	80%	80%	90%	80%	80%	90%
GQA	80%	80%	80%	80%	90%	80%	80%	80%	80%	70%	80%	80%
TallyQA	80%	80%	80%	80%	90%	80%	80%	80%	90%	80%	90%	80%
POPE	90%	80%	80%	80%	90%	80%	80%	90%	90%	80%	80%	80%
RefCOCO+	80%	70%	60%	70%	80%	70%	70%	70%	70%	70%	70%	70%

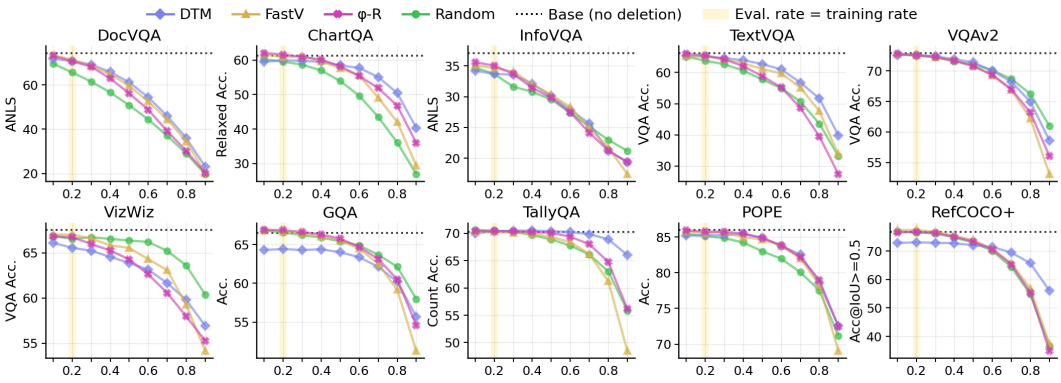


(a) Trained at 60%

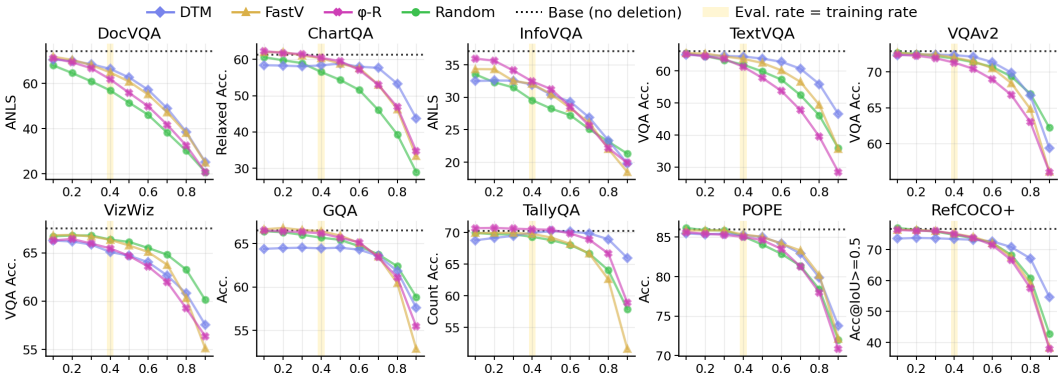


(b) Trained at 80%

Figure 13: **Fixed checkpoints on Gemma 3 4B trained at 60% (a) and 80% (b) deletion, deletion during prefill.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: φ**-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. Trained at 60%, **DTM** leads on all ten benchmarks at 80% and 90% inference deletion; trained at 80%, on nine.

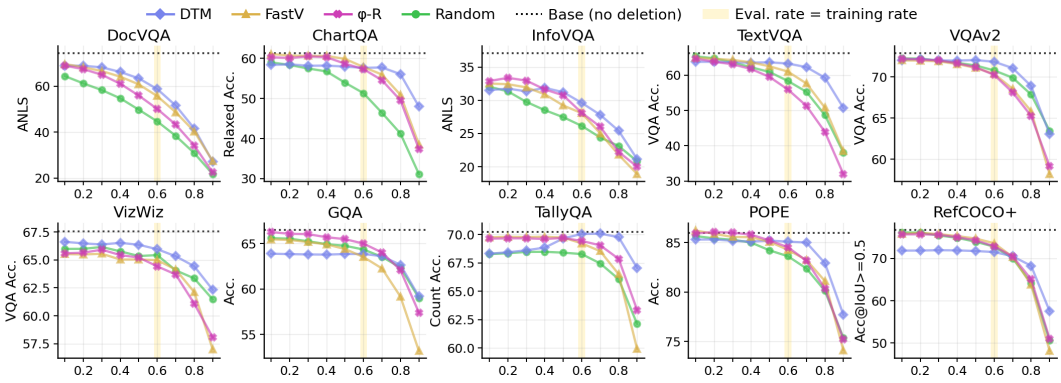


(a) Trained at 20%

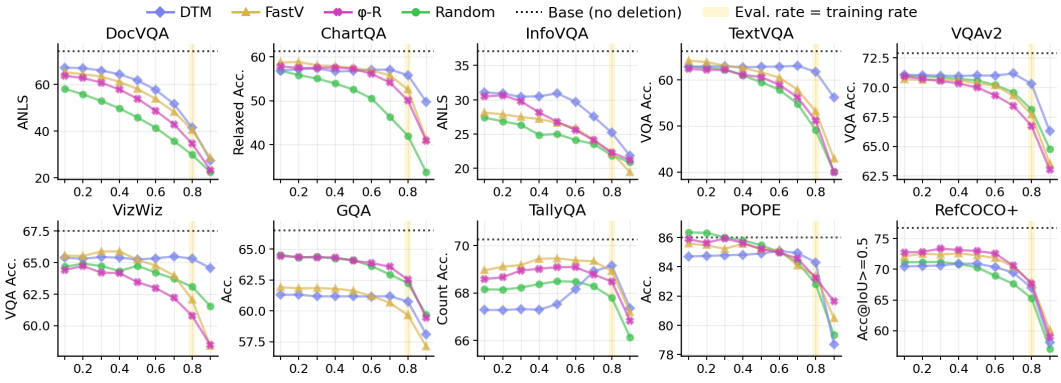


(b) Trained at 40%

Figure 14: **Fixed checkpoints on T5Gemma 2 270M–270M trained at 20% (a) and 40% (b) deletion.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: **φ-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. The smallest backbone transfers less well from low training targets: **DTM** leads on five or six benchmarks at 80% and 90% inference deletion.



(a) Trained at 60%



(b) Trained at 80%

Figure 15: **Fixed checkpoints on T5Gemma 2 270M–270M trained at 60% (a) and 80% (b) deletion.** Each curve evaluates one checkpoint at deletion rates from 10% to 90%; the training rate is highlighted. Blue diamonds: **DTM**; gold triangles: **FastV**; magenta crosses: **φ-R**; green circles: **Random**; dotted line: unpruned Base, fine-tuned the same way. Trained at 60%, **DTM** leads on all ten benchmarks at 80% inference deletion (eight at 90%); trained at 80%, on eight and six.

D.5 Checkpoint Reuse Across Deletion Budgets

For a given method and task, let $S_t(e)$ denote the score of a checkpoint trained at deletion target t and evaluated at rate e. We measure the effect of checkpoint reuse as $S_t(e) - S_e(e)$, holding the inference budget fixed.

Figure 16 separates the loss associated with a tighter budget from the additional cost of checkpoint reuse. For **DTM**, moving from matched 60% to matched 80% deletion lowers the mean score by 3.3, 3.5, and 3.6 points for T5Gemma 2 1B–1B, Gemma 3 4B, and T5Gemma 2 270M–270M, respectively. Reusing the 60%-trained checkpoint at 80% incurs only a further 0.6, 0.2, and 0.2 points. Matched training is not uniformly best: at 60% inference deletion, Gemma 3's 40%-trained **DTM** checkpoints average 0.7 points above their 60%-trained counterparts. These means summarize heterogeneous task metrics and are descriptive comparisons of the selected checkpoints.

T5Gemma 2 1B–1B

	EVALUATION DELETION															
	20%				40%				60%				80%			
TRAIN	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM
20%	71.5	**72.6**	72.5	72.4	0.0	-0.1	-0.2	+0.3	-0.6	-1.3	-2.1	-1.5	-3.8	-5.6	-7.4	-7.3
40%	-0.2	-0.3	-0.5	-0.6	69.0	71.2	70.7	**71.5**	0.0	-0.5	-0.8	-0.2	-2.4	-3.7	-5.3	-3.1
60%	-1.5	-1.1	-0.7	-1.3	-0.7	-0.5	-0.2	-0.5	65.4	68.3	67.5	**70.2**	-0.8	-1.8	-2.2	-0.6
80%	-3.5	-2.1	-2.4	-3.6	-2.2	-1.4	-1.3	-2.6	-0.9	-0.4	-0.2	-1.5	59.9	62.7	61.8	**66.9**

Gemma 3 4B

	EVALUATION DELETION															
	20%				40%				60%				80%			
TRAIN	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM
20%	73.2	74.9	**75.0**	74.9	-0.7	-0.5	-0.1	0.0	-2.0	-1.8	-2.0	+0.2	-5.4	-6.7	-7.1	-2.5
40%	+0.2	0.0	-0.3	-0.4	71.2	73.8	73.3	**74.3**	-0.7	-0.6	-1.1	+0.7	-3.0	-4.6	-5.6	-1.1
60%	-0.5	-0.5	-0.9	-2.0	-0.3	-0.2	-0.2	-1.4	68.0	70.8	70.5	**72.2**	-1.0	-2.6	-2.0	-0.2
80%	-2.5	-2.6	-2.5	-3.4	-1.8	-2.1	-1.6	-2.9	-0.7	-1.0	-0.9	-1.3	62.9	64.9	64.3	**68.7**

T5Gemma 2 270M–270M

	EVALUATION DELETION															
	20%				40%				60%				80%			
TRAIN	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM	Random	FastV	φ-R	DTM
20%	66.0	67.2	**67.2**	66.1	0.0	-0.3	+0.1	-0.1	-0.8	-1.1	-1.1	-1.5	-3.3	-5.5	-4.9	-4.4
40%	-0.1	-0.1	-0.3	-0.4	63.8	**65.6**	65.0	65.2	+0.1	-0.2	-0.6	-0.3	-1.8	-3.5	-4.1	-2.5
60%	-1.1	-1.1	-1.2	-1.0	-0.5	-0.5	-0.2	-0.4	60.6	62.8	61.8	**63.7**	-0.4	-1.6	-1.6	-0.2
80%	-3.3	-3.2	-3.2	-2.3	-2.4	-2.1	-1.8	-1.7	-1.1	-0.9	-0.6	-0.9	55.2	57.8	56.8	**60.1**

Figure 16: **Checkpoint reuse across deletion budgets at L5.** Rows specify training targets; grouped columns specify inference deletion. Gray diagonal cells show matched scores, with the strongest method in bold and underlined. Off-diagonal cells show $S_t(e) - S_e(e)$; negative values favor training at the evaluation rate. All cells average the same ten benchmarks equally. Red shading deepens with the size of the penalty. A 60%-trained **DTM** checkpoint used at 80% costs at most 0.6 points on average, and on Gemma 3 even a 20%-trained **DTM** checkpoint loses less at 80% than any baseline.

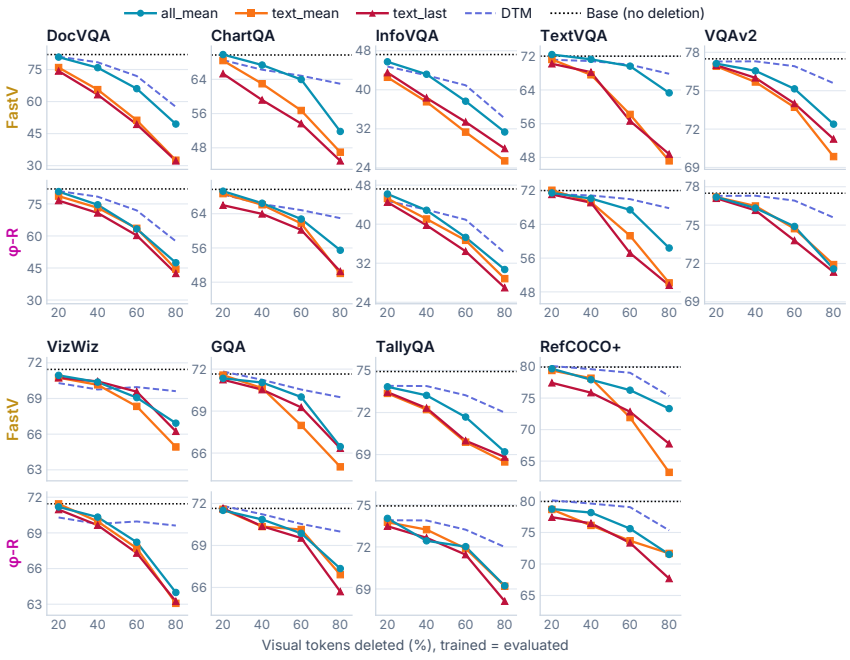


Figure 17: **Query aggregation for FastV and φ-R on T5Gemma 2 1B–1B, matched deletion after layer 5.** Each selector ranks visual tokens by attention averaged over all valid queries (`all_mean`, used in the main results), over text queries only (`text_mean`), or from the final text query (`text_last`); each variant is separately fine-tuned. Dashed: DTM; dotted: unpruned Base. All-query scoring is the strongest FastV variant almost everywhere, by up to 17 points on DocVQA.

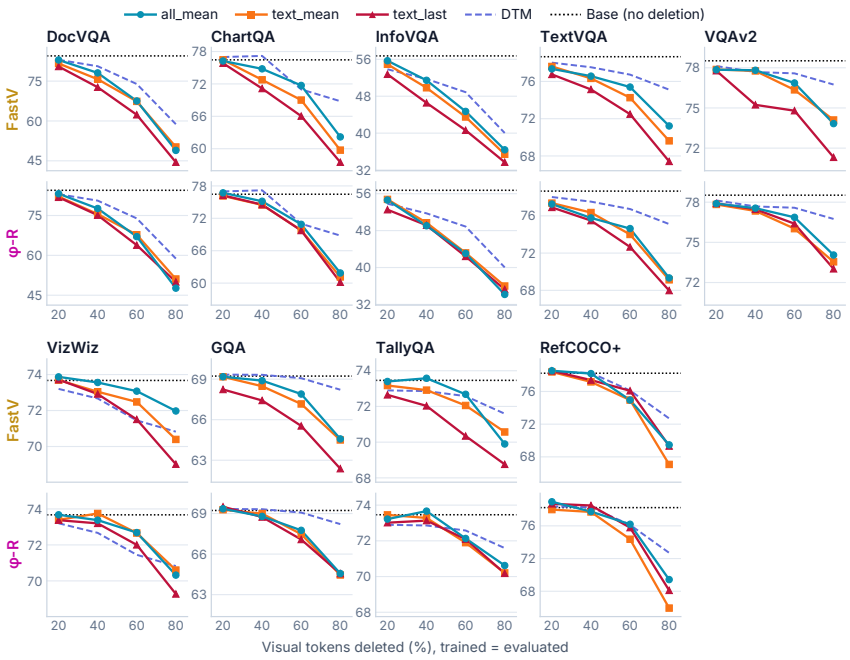


Figure 18: **Query aggregation for FastV and φ-R on Gemma 3 4B, matched deletion after five decoder layers during prefill.** Variants and line styles as in Figure 17. FastV again prefers all-query scoring (35 of 36 settings against `text_last`), whereas φ-R gains from text queries on DocVQA and InfoVQA at 80%.

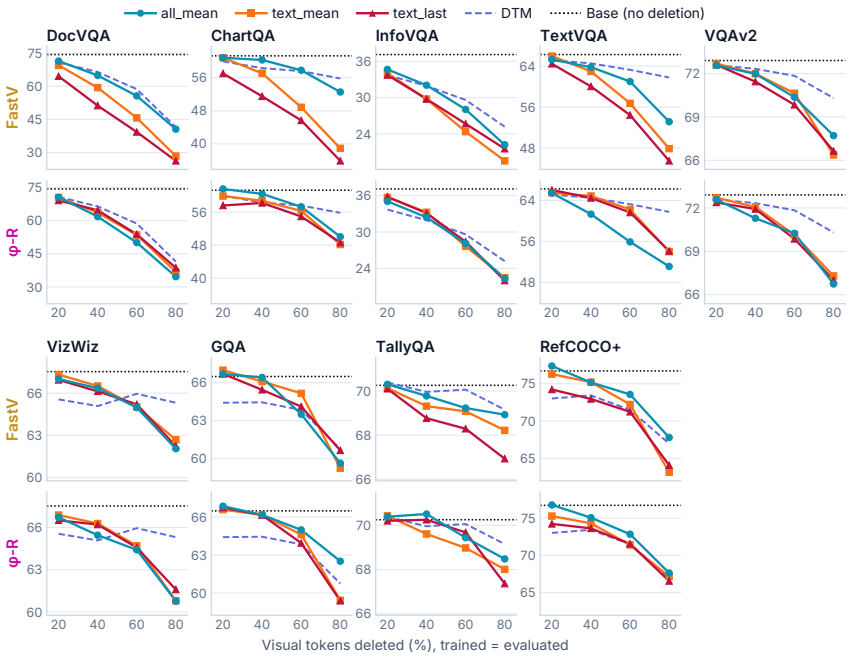


Figure 19: **Query aggregation for FastV and φ-R on T5Gemma 2 270M–270M, matched deletion after layer 5.** Variants and line styles as in Figure 17. On the smallest backbone, text queries help φ-R on DocVQA and TextVQA but not FastV; neither reaches DTM at 60–80% deletion on these tasks.

D.6 Attention-Score Aggregation

We compare `all_mean` (all valid queries), `text_mean` (text queries only), and `text_last` (the final text query), averaging heads in each case. Each variant is separately fine-tuned. `text_mean` matches the ranking criterion of LearnPruner's second stage (Takezoe et al., 2026); we evaluate it under our pruning placement and task fine-tuning, omitting LearnPruner's learned first stage, which operates before the language model. Figures 17–19 compare the three variants for FastV and φ-R at matched deletion rates on all three backbones, using the nine benchmarks evaluated with text queries on every backbone (all but POPE).

FastV favors `all_mean` in most settings: over `text_last` in 99 of 108 comparisons across the three backbones, and over `text_mean` in 87 of 108. Its advantage is particularly large for T5Gemma 2: at 80% deletion on DocVQA, the scores are 49.48, 32.54, and 32.24 for all, mean-text, and last-text queries, and on T5Gemma 2 270M–270M ChartQA, 52.55 against 35.89 with `text_last`. However, all-query aggregation is not universally best, particularly for φ-R, which it favors in 68 of 108 and 84 of 108 comparisons. On Gemma 3 DocVQA at 80%, φ-R reaches 51.11 with `text_mean`, vs. 47.66 with `all_mean`; DTM remains higher at 58.79. On T5Gemma 2 270M–270M, text queries raise φ-R by up to 6.3 points on TextVQA (62.24 vs. 55.91 at 60%), still below DTM's 63.27. At 80% deletion, only one text-query variant on any backbone scores above DTM (φ-R with `text_mean` on 270M–270M RefCOCO+, 67.19 vs. 66.97). Thus, the headline baselines use a strong common aggregation choice, while these additional variants expose where it leaves performance available, without closing the gap to DTM under aggressive deletion.

D.7 Choosing the Deletion Layer

DTM learns token importance from contextual hidden states rather than attention weights, allowing it to operate across sliding-window and global attention without changing its scoring rule. Figure 20 demonstrates this robustness on T5Gemma 2: at 80% deletion on the 1B–1B model, moving the gate from after layer 5 (L5), where attention-based selectors read a sliding-window layer, to after layer 6 (L6), where they read the first global-attention layer, reduces FastV from 49.5 to 37.2 ANLS

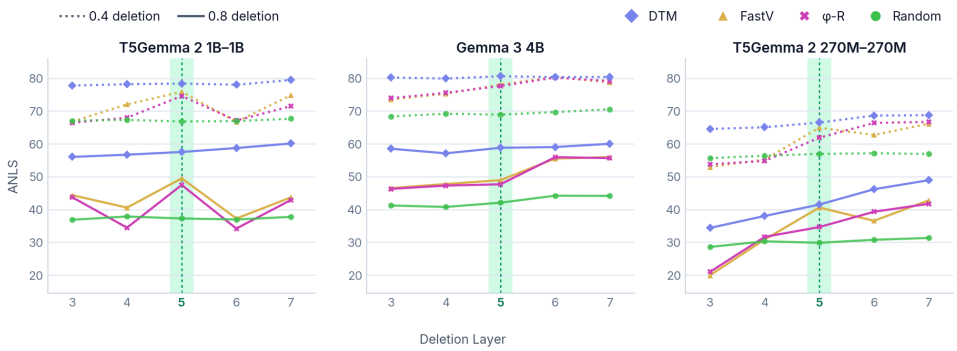


Figure 20: **Sensitivity to deletion depth on DocVQA.** Models are separately trained and evaluated at 40% (dotted) or 80% (solid) deletion with the gate after each layer. The horizontal axis gives the number of layers before deletion; shading marks L5, the setting used in our main experiments. On T5Gemma 2 1B–1B, the attention-based selectors drop when they read the first global-attention layer (L6), whereas **DTM** does not.

and **φ-R** from 47.5 to 34.2, while **DTM** improves from 57.5 to 58.7. This contrast is less apparent on Gemma 3, whose larger 1,024-token window, compared with T5Gemma 2's roughly 256-token reach in each direction, may narrow the difference in attention visibility between local and global layers for these inputs. Robustness across attention patterns lets us place the gate early, where more layers benefit from the shortened sequence; we use L5 as a shared quality–compute compromise, although later deletion can improve accuracy at additional cost: at 80% deletion on T5Gemma 2 1B–1B, deleting after layer 7 instead of layer 5 raises **DTM**'s ANLS by 2.6 but saves 53.4% rather than 58.9% of FLOPs (Appendix F.2). Appendix D.8 extends this analysis to 20–80% deletion.

D.8 Deletion Depth Across Backbones and Budgets

Figures 21–23 extend the depth analysis of Figure 20 to matched deletion rates of 20%, 40%, 60%, and 80%. Each point uses a separately fine-tuned checkpoint for its layer and deletion target, evaluated on the same 5,349 DocVQA validation examples.

Dependence on the deletion budget. Across L3–L7, **DTM** achieves the highest ANLS at every layer on all three backbones at 40%, 60%, and 80% deletion. Its advantage is less consistent at 20%, where attention-based selection can match or exceed it. For example, at L6 on Gemma 3, **FastV** scores 83.8 versus 82.9 for **DTM**; at L7 on T5Gemma 2 270M–270M, the corresponding scores are 71.9 and 71.7. Some higher-deletion comparisons are also close: at 40% deletion and L6 on Gemma 3, **DTM** and **FastV** differ by 0.02 ANLS. The overall ranking therefore includes settings where the methods perform nearly identically.

Model size and depth sensitivity. The smaller T5Gemma 2 backbone benefits increasingly from later deletion as the token budget tightens. Moving **DTM** from L3 to L7 improves ANLS by 1.7, 4.2, 8.5, and 14.5 points at the four respective deletion rates. In contrast, **DTM** on the two larger backbones varies little across depth at 20% deletion, and its L3-to-L7 gains remain smaller even at 80% (4.1 and 1.5 points). **Random** deletion is comparatively insensitive to depth: its largest best-to-worst ranges across the tested rates are 1.4 ANLS on T5Gemma 2 1B–1B, 2.7 on T5Gemma 2 270M–270M, and 3.7 on Gemma 3. These results show that the value of additional processing before deletion depends on both the backbone and the token-selection method.

Which layer favors attention-based selection? On T5Gemma 2 1B–1B, both **FastV** and **φ-R** attain their highest ANLS at L5 at all four deletion rates. On Gemma 3, **φ-R** instead peaks at L6 throughout, while **FastV** peaks at L6 for 20% and 40% deletion and at L7 for 60% and 80%. On T5Gemma 2 270M–270M, both attention-based methods peak at L7, although **FastV** drops from L5 to L6 before recovering. L6 reads the first global-attention layer, but the T5Gemma 2 1B–1B drop is not shared across backbones. These sweeps change the scoring layer and learned representations

together, so they do not isolate a causal effect of attention type. Deleting later also saves fewer FLOPs (Table 17).

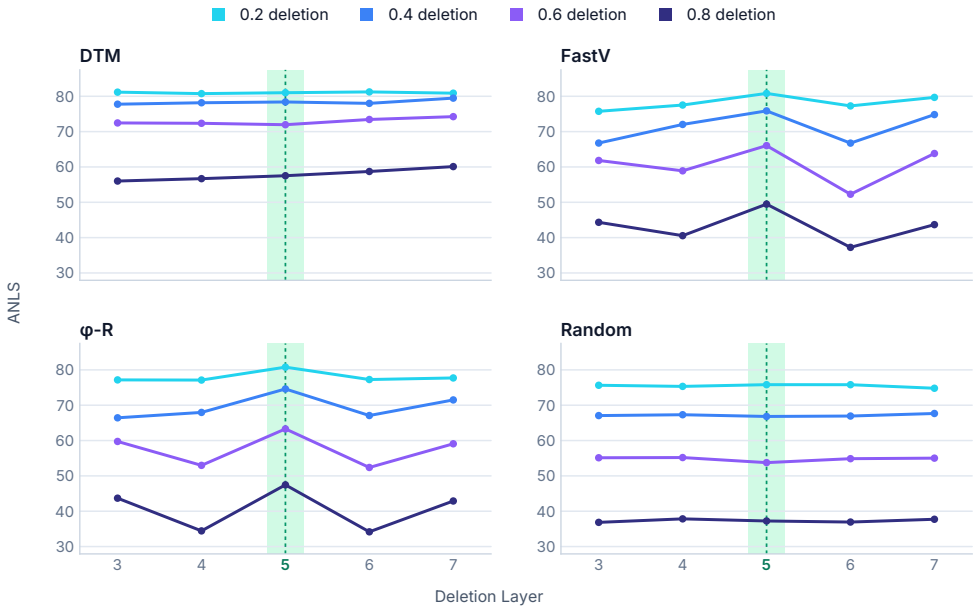


Figure 21: **Deletion-depth sensitivity across budgets for T5Gemma 2 1B–1B on DocVQA.** Panels show individual methods; line colors indicate matched training and evaluation deletion rates (20%, 40%, 60%, 80%). The horizontal axis gives the number of layers before deletion; shading marks L5. **DTM** is nearly flat across depth, whereas **FastV** and **φ-R** peak at L5 and fall at L6 at every rate.

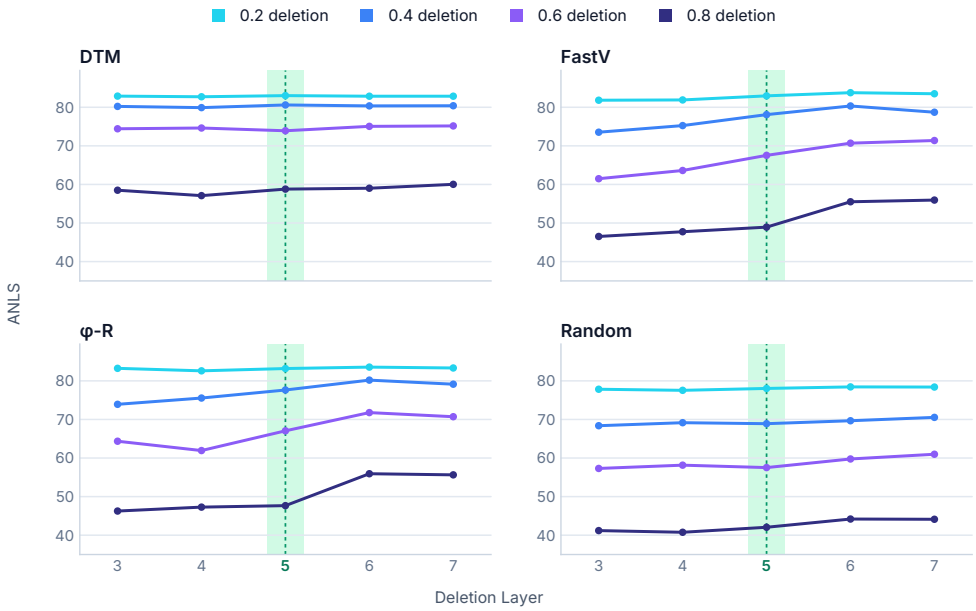


Figure 22: **Deletion-depth sensitivity across budgets for Gemma 3 4B on DocVQA.** Panels show individual methods; line colors indicate matched deletion rates (20%, 40%, 60%, 80%); shading marks L5. The attention-based selectors improve with later deletion on this backbone, while **DTM** stays nearly constant and is highest at 40–80%.

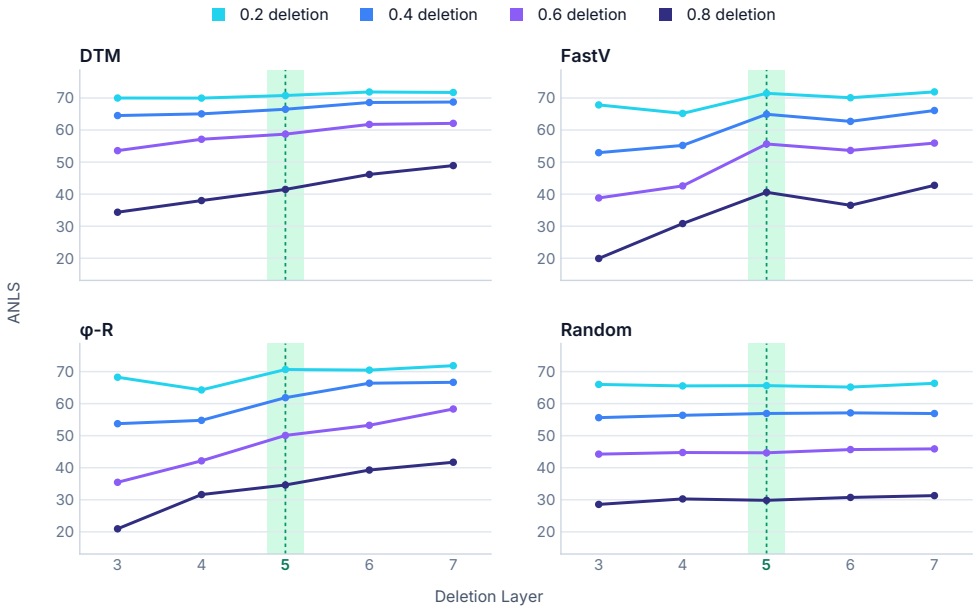


Figure 23: **Deletion-depth sensitivity across budgets for T5Gemma 2 270M–270M on DocVQA.** Panels show individual methods; line colors indicate matched deletion rates (20%, 40%, 60%, 80%); shading marks L5. The smallest backbone benefits most from later deletion, especially at 80%.

E What the Gate Keeps

E.1 Spatial Evidence Retention at Other Deletion Rates

Figure 24 repeats the analysis of Figure 4 at matched 20%, 40%, and 60% deletion. Evidence regions come from each benchmark's spatial annotations where available (e.g., answer-bearing OCR words, object boxes, or referent boxes) and from grounding proxies otherwise; a patch counts as evidence if it overlaps such a region. Methods are compared on the same eligible questions within each model–dataset pair.

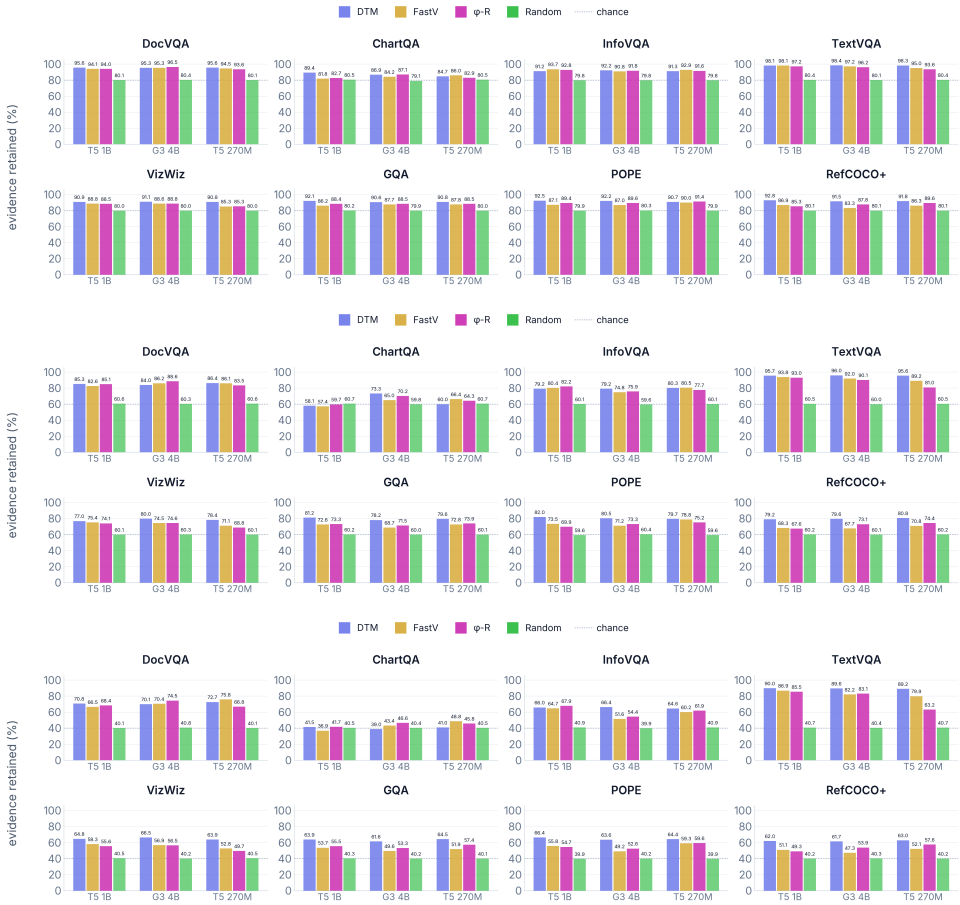


Figure 24: **Spatial evidence retention at matched 20% (top), 40% (middle), and 60% (bottom) visual token deletion after layer 5.** Bars show the mean percentage of evidence patches retained on the same eligible questions for each model–dataset pair; dashed lines mark chance. Blue: **DTM**; gold: **FastV**; magenta: **φ-R**; green: **Random**. **DTM** retains the most evidence in 17, 19, and 18 of 24 comparisons at 20%, 40%, and 60%, and its lead grows with the deletion rate.

E.2 Evidence, Correctness, and Selector Agreement

Figure 25 splits each model's 80% evaluations by whether **DTM** kept more or less of the evidence than **FastV** and shows how often only one of the two answers correctly. Where **DTM** keeps more evidence, it is the only correct method on 13.5% of questions, against 6.2% for **FastV** alone (pooled over 72,884 questions), and it leads in 23 of 24 model–benchmark pairs; the gap is largest on DocVQA (37.3% vs. 4.5% on T5Gemma 2 1B–1B). Where it keeps less, the ordering reverses (8.0% vs. 11.1%). Retaining evidence is thus associated with which method answers correctly, although this comparison is correlational.

Figure 26 shows one way the attention-based selectors spend budget that **DTM** does not. On Gemma 3, **FastV** and **φ-R** keep the four corner patches of the image at median rates of 0.50 and 0.42, above the uniform rate of 0.20 on nine of ten benchmarks, whereas **DTM** keeps them at 0.02–0.12 on every model and benchmark. The two attention-based selectors also agree with each other far more than either agrees with **DTM** (median Cohen's κ of 0.59, against 0.29 and 0.30), while **DTM**'s keep-sets agree across the three separately trained backbones (median κ 0.63, against 0.34 for **FastV** and 0.29 for **φ-R**). Gates trained on different models thus converge on similar selections.

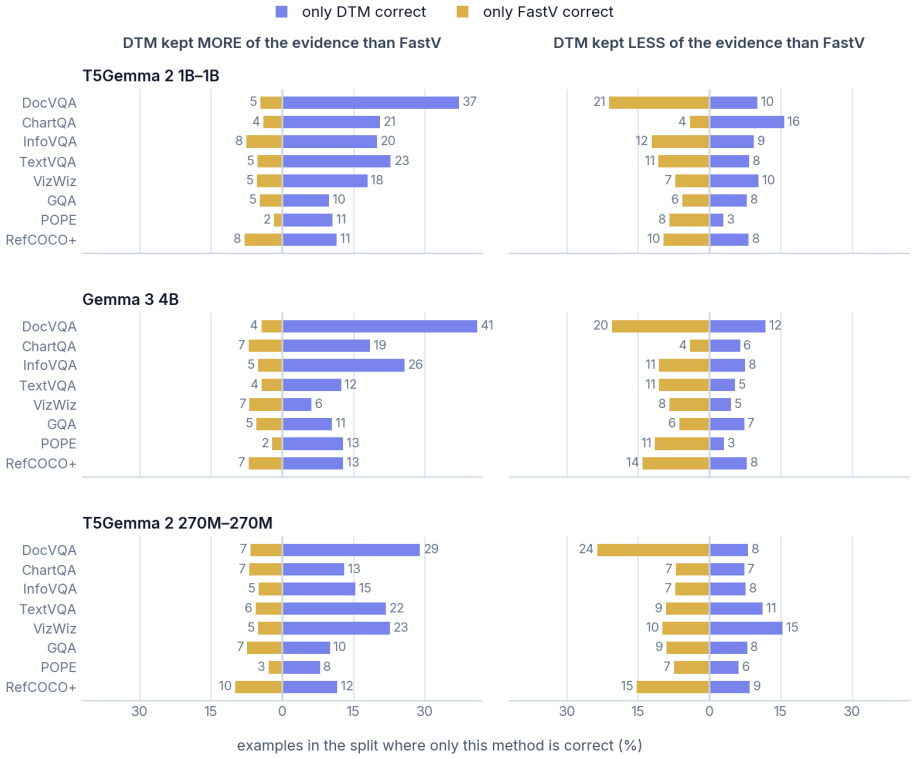


Figure 25: **Evidence retention and which method answers correctly, at matched 80% deletion.** For each model and benchmark, questions are split by whether **DTM** kept more (left) or less (right) of the evidence patches than **FastV**; bars give the percentage of questions in each split that only **DTM** (blue) or only **FastV** (gold) answers correctly. When **DTM** keeps more evidence, it is the sole correct method more than twice as often as **FastV** (13.5% vs. 6.2% of questions, pooled).

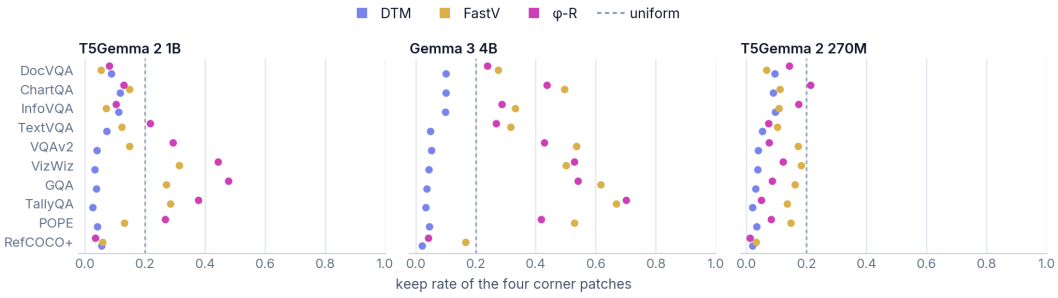


Figure 26: **Keep rate of the four corner patches at matched 80% deletion.** Blue: **DTM**; gold: **FastV**; magenta: **φ-R**; the dashed line is the rate under uniform selection (0.20). On Gemma 3, the attention-based selectors keep the corners far more often than uniform selection would, whereas **DTM** rarely does on any model.

E.3 Randomly Sampled Examples at 80% Deletion

Figures 27–30 show 15 ChartQA and 15 TextVQA validation questions drawn uniformly at random, without regard to correctness, from all questions scored by the four T5Gemma 2 1B–1B checkpoints trained and evaluated at 80% deletion; each method keeps 52 of 256 visual tokens. On these samples, **DTM** answers 8 of 15 ChartQA and 9 of 15 TextVQA questions correctly, compared with 4 and 11 for **FastV**, 6 and 8 for **φ-R**, and 7 and 7 for **Random**. On charts, **DTM** tends to keep titles, axis and tick labels, and the ends of bars and lines. Table 13 measures this on the full ChartQA validation set: **DTM** keeps more text-label and bar/line patches than both attention-based selectors on all three backbones, and more numeric tick labels on Gemma 3 and T5Gemma 2 270M–270M, slightly fewer than **FastV** on T5Gemma 2 1B–1B (83.2% vs. 84.1%). This is not a general preference for text: on DocVQA, the attention-based selectors cover more of the OCR text on every backbone, and on TextVQA **FastV** does so on T5Gemma 2 1B–1B. **DTM**'s advantage there lies in the answer-bearing words (Figure 4).

Table 13: **What each method keeps on ChartQA and OCR-text benchmarks at matched 80% deletion.** Recall is the percentage of patches in each category that a method keeps; uniform selection keeps 20.3%. ChartQA categories come from the dataset's element annotations (text labels include titles, axis, and legend text; bar and line marks are plotted elements). "Blank" is the share of the kept budget spent on blank patches, which make up 47% of ChartQA patches. OCR text covers every word detected in the image. On charts, **DTM** keeps more text labels and marks than both attention-based selectors on every backbone; on documents, the attention-based selectors cover more text.

		ChartQA				OCR-text recall	
Model	Method	Text labels	Numeric ticks	Bars/lines	Blank (% of kept)	TextVQA	DocVQA
T5Gemma 2 1B–1B	**DTM**	**66.8**	83.2	**24.3**	**20.2**	58.0	31.4
	FastV	60.7	**84.1**	17.4	26.9	**62.5**	**35.8**
	φ-R	51.2	66.7	21.0	27.3	58.4	34.7
Gemma 3 4B	**DTM**	**66.7**	**80.3**	**24.5**	20.0	**66.2**	28.6
	FastV	49.9	49.9	20.6	**17.7**	54.5	34.0
	φ-R	47.1	37.3	23.4	**17.7**	56.4	**34.1**
T5Gemma 2 270M–270M	**DTM**	**57.1**	**70.7**	**24.7**	26.6	**61.2**	30.4
	FastV	52.9	53.7	21.4	**24.0**	51.0	**41.2**
	φ-R	38.8	39.8	22.9	28.3	45.7	33.0

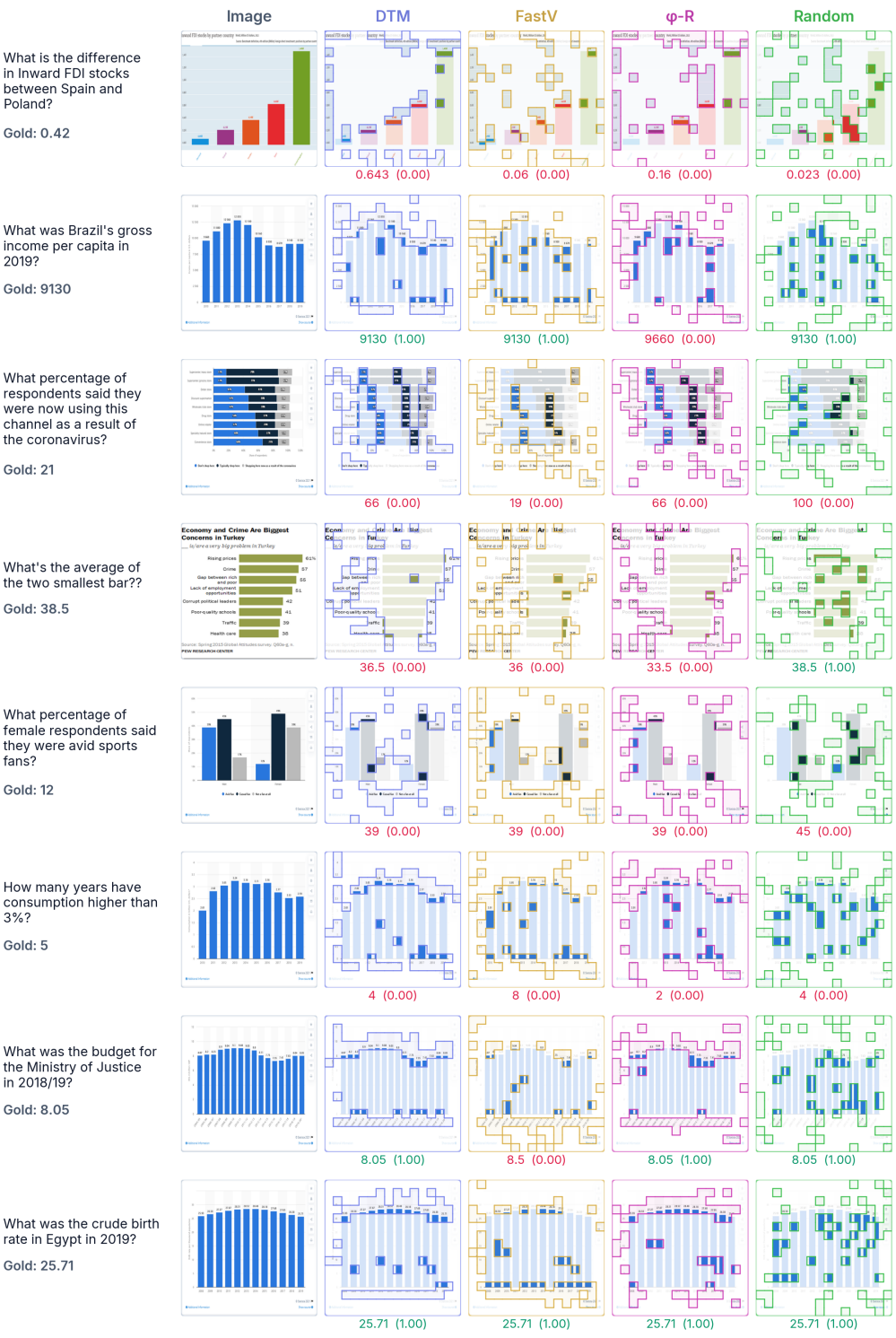


Figure 27: **Randomly sampled ChartQA questions at matched 80% deletion on T5Gemma 2 1B–1B (1 of 2).** Each row shows the input with its question and gold answer, then the 52 of 256 visual tokens kept by each method in full contrast (outlined), with deleted patches washed out. Below each panel: the prediction and its relaxed-accuracy score. Blue: **DTM**; gold: **FastV**; magenta: **φ-R**; green: **Random**. **DTM** concentrates its budget on labels, tick values, and bar ends.

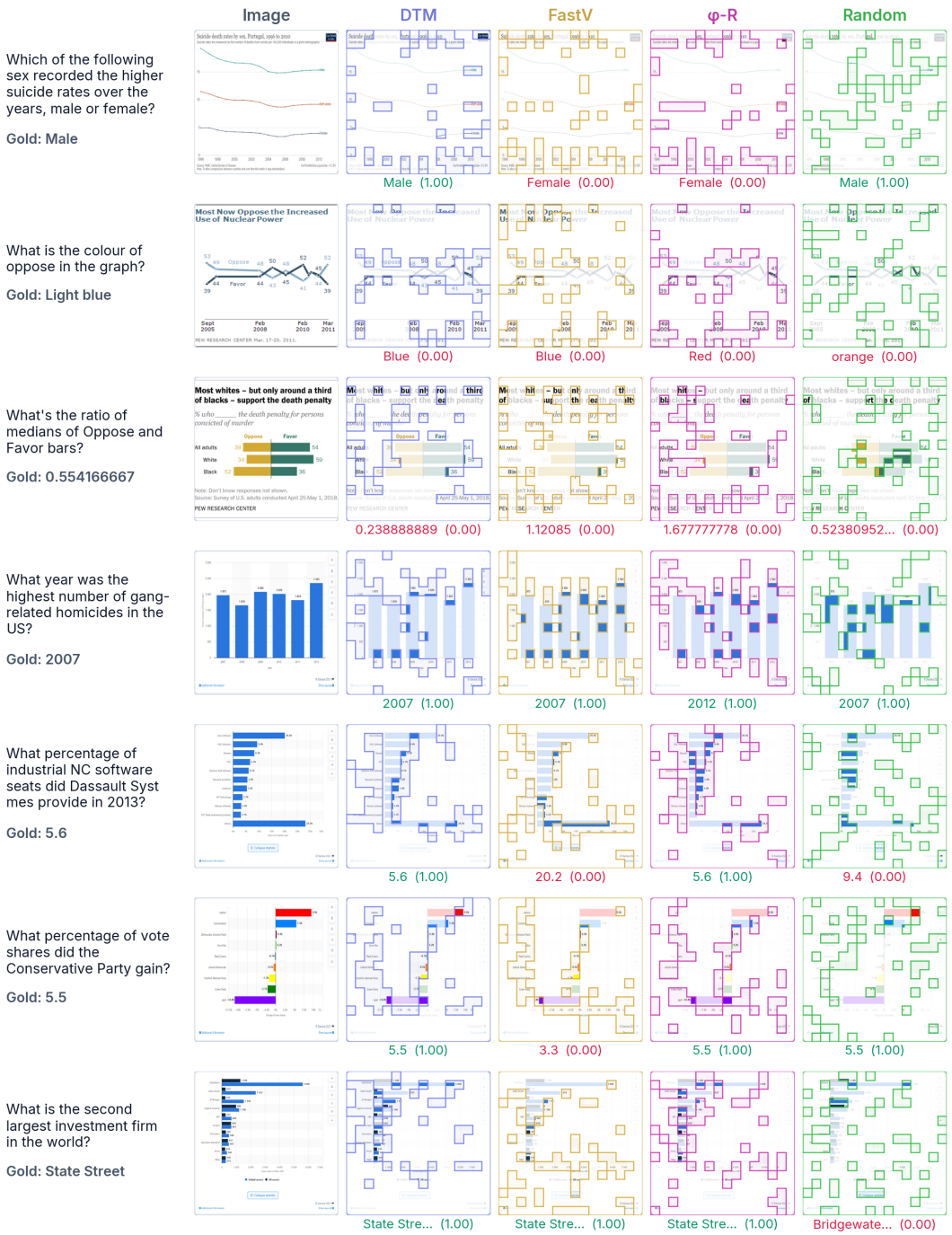


Figure 28: **Randomly sampled ChartQA questions at matched 80% deletion on T5Gemma 2 1B–1B (2 of 2).** Presented as in Figure 27: kept tokens in full contrast, predictions and relaxed-accuracy scores below each panel; blue **DTM**, gold **FastV**, magenta **φ-R**, green **Random**.

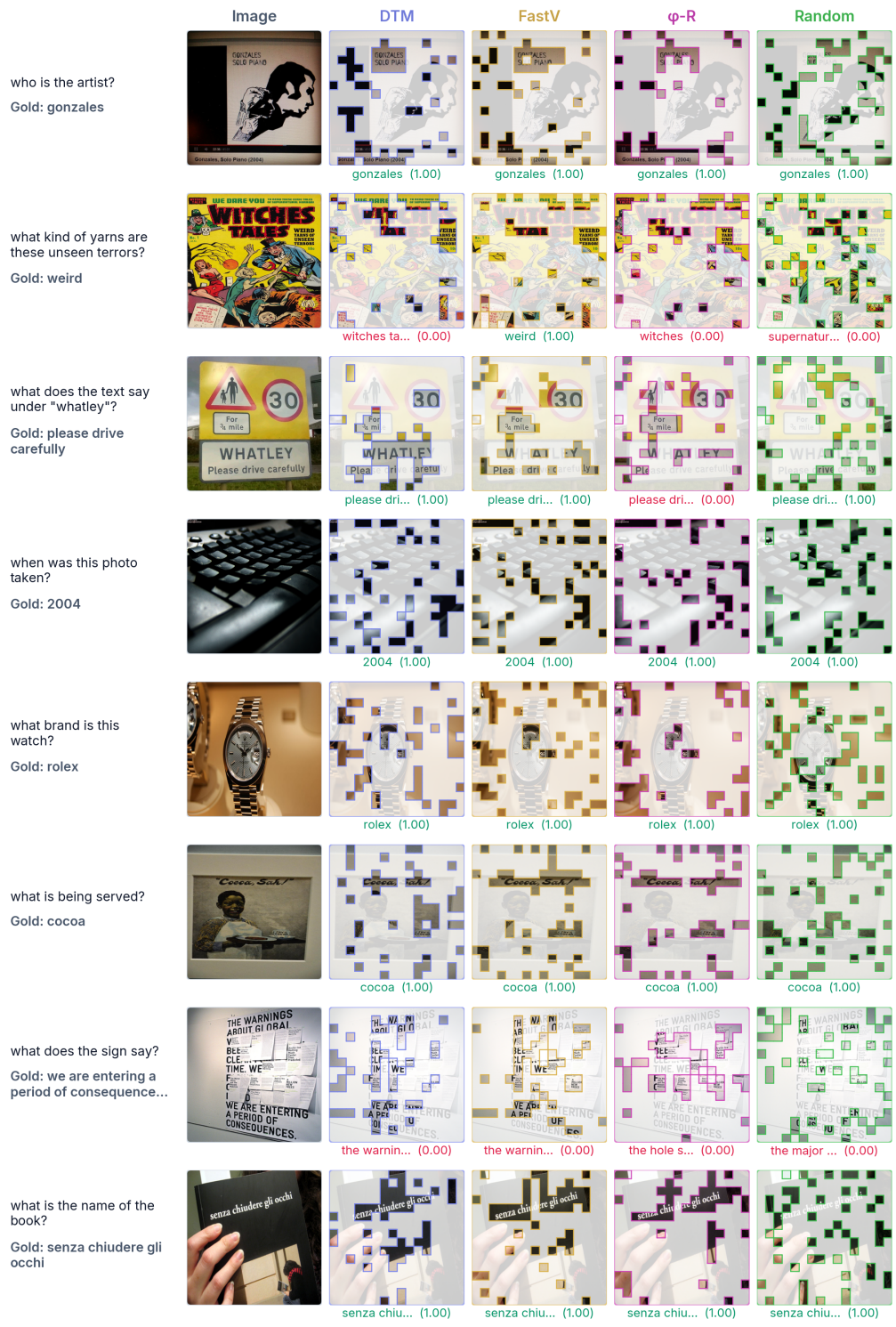


Figure 29: **Randomly sampled TextVQA questions at matched 80% deletion on T5Gemma 2 1B–1B (1 of 2).** Each row shows the input with its question and gold answer, then the 52 of 256 visual tokens kept by each method in full contrast (outlined), with deleted patches washed out; below each panel, the prediction (long answers truncated) and its VQA-accuracy score. Blue: **DTM**; gold: **FastV**; magenta: **φ-R**; green: **Random**. All three informed selectors keep much of the scene text; they differ in which words remain.



Figure 30: **Randomly sampled TextVQA questions at matched 80% deletion on T5Gemma 2 1B–1B (2 of 2).** Presented as in Figure 29; blue **DTM**, gold **FastV**, magenta **φ-R**, green **Random**.

F FLOPs Metrics

F.1 FLOPs Calculations

Tables 14, 15, and 16 list, for each benchmark, the input and output text lengths and the FLOPs saved at each visual deletion rate after layer 5. Savings are nearly constant across the nine benchmarks with short prompts and lower on RefCOCO+, whose longer text prompts and outputs are never deleted. At every rate, Gemma 3 4B saves slightly more than T5Gemma 2 1B–1B, and T5Gemma 2 270M–270M saves the least; at 80% deletion, the means are 61.4%, 58.4%, and 53.1%. Table 17 (Appendix F.2) shows how the savings shrink as the gate moves later, and Appendix G.5 defines the complementary cost attributable to visual tokens alone.

Table 14: **Per-dataset input and output lengths and FLOPs savings for T5Gemma 2 1B–1B at L5.** FLOPs are calculated consistent with the description in Appendix G. Input text counts are unpadded, nonvisual encoder-input tokens, including prompt and special markers but excluding the generated answer. Output text counts include generated stop tokens and are pooled across the nine deletion rates. The additional selection FLOPs of **DTM**, **FastV**, and **φ-R** are negligible (Appendices G.4.2 and G.4.3). SD is the population standard deviation across evaluated examples. Each saving is computed from summed theoretical forward costs at the recorded input and generated-output lengths; the vision encoder and image projector are excluded. The final row weights tasks equally. Savings are nearly constant across benchmarks except RefCOCO+, whose longer prompts and outputs are never deleted.

Dataset	Input text tokens mean ± SD	Output text tokens mean ± SD	FLOPs saved (%) at visual deletion rate								
			10%	20%	30%	40%	50%	60%	70%	80%	90%
DocVQA	18.9 ± 4.8	5.9 ± 3.5	7.3	14.9	22.1	29.6	37.1	44.3	51.7	58.9	66.3
ChartQA	23.6 ± 5.6	4.4 ± 1.9	7.2	14.7	21.9	29.4	36.8	43.9	51.2	58.3	65.6
InfoVQA	22.0 ± 5.3	4.3 ± 2.2	7.3	14.8	22.1	29.5	37.0	44.2	51.6	58.7	66.0
TextVQA	16.3 ± 2.5	3.7 ± 2.2	7.4	15.2	22.6	30.2	37.8	45.2	52.7	60.0	67.5
VQAv2	15.3 ± 2.0	2.3 ± 0.7	7.5	15.3	22.8	30.5	38.3	45.6	53.3	60.6	68.2
VizWiz	16.9 ± 7.0	3.6 ± 1.3	7.4	15.1	22.5	30.2	37.8	45.1	52.7	59.9	67.4
GQA	17.9 ± 3.7	2.1 ± 0.3	7.5	15.2	22.6	30.3	37.9	45.3	52.9	60.1	67.7
TallyQA	15.8 ± 2.1	2.0 ± 0.1	7.5	15.3	22.8	30.5	38.2	45.6	53.3	60.6	68.2
POPE	16.2 ± 0.4	2.0 ± 0.0	7.5	15.3	22.8	30.5	38.2	45.6	53.2	60.5	68.1
RefCOCO+	62.9 ± 2.4	29.0 ± 0.8	5.8	11.8	17.5	23.5	29.4	35.1	41.0	46.6	52.4
Mean across tasks	—	—	7.3	14.8	22.0	29.4	36.9	44.0	51.4	58.4	65.8

Table 15: **Per-dataset input and output lengths and FLOPs savings for Gemma 3 4B at L5.** FLOPs savings are presented in the same way as Table 14. Gemma 3 saves slightly more than T5Gemma 2 1B–1B at every deletion rate.

Dataset	Input text tokens mean ± SD	Output text tokens mean ± SD	FLOPs saved (%) at visual deletion rate								
			10%	20%	30%	40%	50%	60%	70%	80%	90%
DocVQA	18.9 ± 4.8	5.8 ± 3.5	7.6	15.6	23.2	31.1	39.0	46.5	54.4	62.0	69.9
ChartQA	23.6 ± 5.6	4.4 ± 2.1	7.6	15.4	23.0	30.8	38.6	46.1	53.8	61.3	69.1
InfoVQA	22.0 ± 5.3	4.2 ± 2.2	7.6	15.5	23.1	31.0	38.8	46.4	54.2	61.7	69.5
TextVQA	16.3 ± 2.5	3.7 ± 2.5	7.8	15.9	23.6	31.7	39.7	47.4	55.4	63.0	71.0
VQAv2	15.3 ± 2.0	2.3 ± 0.7	7.9	16.0	23.9	32.0	40.1	47.9	55.9	63.7	71.7
VizWiz	16.9 ± 7.0	3.5 ± 1.3	7.8	15.9	23.6	31.6	39.6	47.3	55.3	63.0	70.9
GQA	17.9 ± 3.7	2.1 ± 0.3	7.8	15.9	23.7	31.7	39.8	47.5	55.5	63.1	71.1
TallyQA	15.8 ± 2.1	2.0 ± 0.1	7.9	16.0	23.8	32.0	40.1	47.8	55.9	63.6	71.7
POPE	16.2 ± 0.4	2.0 ± 0.0	7.8	16.0	23.8	31.9	40.0	47.8	55.8	63.6	71.6
RefCOCO+	62.9 ± 2.4	29.0 ± 0.6	6.1	12.4	18.5	24.8	31.1	37.2	43.5	49.5	55.7
Mean across tasks	—	—	7.6	15.5	23.0	30.9	38.7	46.2	54.0	61.4	69.2

Table 16: **Per-dataset input and output lengths and FLOPs savings for T5Gemma 2 270M–270M at L5.** FLOPs savings are presented in the same way as Table 14. T5Gemma 2 270M–270M saves the least of the three backbones at every rate.

Dataset	Input text tokens mean ± SD	Output text tokens mean ± SD	FLOPs saved (%) at visual deletion rate								
			10%	20%	30%	40%	50%	60%	70%	80%	90%
DocVQA	18.9 ± 4.8	5.8 ± 3.5	6.9	13.9	20.6	27.4	34.2	40.6	47.2	53.4	59.9
ChartQA	23.6 ± 5.6	4.4 ± 2.0	6.9	13.9	20.5	27.4	34.1	40.4	46.9	53.1	59.4
InfoVQA	22.0 ± 5.3	4.2 ± 2.2	6.9	14.0	20.7	27.5	34.3	40.7	47.2	53.5	59.9
TextVQA	16.3 ± 2.5	3.7 ± 2.1	7.1	14.3	21.1	28.2	35.1	41.6	48.3	54.7	61.2
VQAv2	15.3 ± 2.0	2.3 ± 0.7	7.2	14.5	21.5	28.6	35.6	42.2	49.0	55.5	62.0
VizWiz	16.9 ± 7.0	3.5 ± 1.3	7.1	14.3	21.1	28.1	35.1	41.6	48.3	54.6	61.1
GQA	17.9 ± 3.7	2.1 ± 0.3	7.1	14.4	21.3	28.4	35.4	42.0	48.7	55.1	61.6
TallyQA	15.8 ± 2.1	2.0 ± 0.1	7.2	14.5	21.5	28.6	35.6	42.3	49.1	55.5	62.1
POPE	16.2 ± 0.4	2.0 ± 0.0	7.2	14.5	21.5	28.6	35.6	42.2	49.0	55.4	62.0
RefCOCO+	62.9 ± 2.4	29.0 ± 0.0	5.2	10.6	15.7	20.9	26.0	30.9	35.8	40.5	45.4
Mean across tasks	—	—	6.9	13.9	20.5	27.4	34.1	40.4	47.0	53.1	59.5

F.2 FLOPs Savings Across Deletion Layers

Table 17: **Per-layer FLOPs savings on DocVQA.** Each row shows the FLOPs saved (%) when the gate follows the indicated layer, evaluated across visual deletion rates. FLOPs are calculated in the same way as Table 14.

Gate layer	FLOPs saved (%) at visual deletion rate								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
T5Gemma 2 1B–1B									
3	8.0	16.2	24.1	32.3	40.5	48.4	56.5	64.3	72.4
4	7.6	15.6	23.1	31.0	38.8	46.3	54.1	61.6	69.4
5	7.3	14.9	22.1	29.6	37.1	44.3	51.7	58.9	66.3
6	7.0	14.2	21.1	28.2	35.4	42.2	49.3	56.1	63.2
7	6.6	13.5	20.1	26.9	33.7	40.2	46.9	53.4	60.1
Gemma 3 4B									
3	8.2	16.7	24.8	33.2	41.7	49.7	58.1	66.2	74.6
4	7.9	16.1	24.0	32.2	40.3	48.1	56.3	64.1	72.2
5	7.6	15.6	23.2	31.1	39.0	46.5	54.4	62.0	69.9
6	7.4	15.0	22.4	30.0	37.6	44.9	52.5	59.9	67.5
7	7.1	14.5	21.6	28.9	36.3	43.3	50.7	57.7	65.0
T5Gemma 2 270M–270M									
3	7.9	15.9	23.5	31.4	39.1	46.4	53.9	61.1	68.4
4	7.4	14.9	22.1	29.4	36.6	43.5	50.6	57.3	64.1
5	6.9	13.9	20.6	27.4	34.2	40.6	47.2	53.4	59.9
6	6.4	12.9	19.1	25.5	31.7	37.7	43.8	49.6	55.6
7	5.9	11.9	17.6	23.5	29.3	34.8	40.4	45.8	51.3

G FLOPs Accounting

G.1 Notation and counting convention

We count two FLOPs per multiply–add. Counts include dominant matrix multiplications in the language backbone and vocabulary projection, but exclude SigLIP and the multimodal projector. Normalization, RoPE, softmax, pointwise activations, residual additions, biases, averaging, ranking, and token gathering are omitted. These are aggregate forward-FLOP estimates, not wall-clock latency or detailed cached-decoding costs.

For T5Gemma 2, N and M are encoder and decoder lengths; L_e, L_d are their depths and W_e, W_d their local-attention windows. The hidden width, head width, GeGLU intermediate width, query-head count, key/value-head count, and vocabulary size are denoted by d_{model}, d_{head}, d_{ff}, n_{heads}, $n_{\text{kv-heads}}$, and n_{vocab}, respectively. Dimensions and layer types are read from each stack's configuration; we do not assume $d_{\text{model}} = n_{\text{heads}} d_{\text{head}}$. The main text's n, m, d, d_h correspond to $N, M, d_{\text{model}}, d_{\text{head}}$ here.

G.2 T5GEMMA 2

An attention pair consists of one query position and one key position; their projected vectors determine one attention score. The functions below count these pairs *per query head*, under the stated approximations, for encoder and merged-attention decoder layer i, respectively:

$$\mathcal{P}_{\text{enc}}(N, i) = \begin{cases} N^2, & \text{global,} \\ N \min(N, W_e), & \text{local,} \end{cases} \tag{10}$$

$$\mathcal{P}_{\text{dec}}(N, M, i) = \begin{cases} MN + M^2, & \text{global,} \\ MN + M \min(M, W_d), & \text{local.} \end{cases} \tag{11}$$

Every decoder query attends to all encoder keys, even in local decoder layers. The M^2 term deliberately uses a full-square approximation rather than exact causal pairs. Local counts assume only window-permitted pairs are computed, not a dense matrix followed by masking, and ignore window-boundary effects.

Grouped-query attention shares key/value heads across query heads. The query and output projections therefore scale with n_{heads}, whereas the key and value projections scale with $n_{\text{kv-heads}} \leq n_{\text{heads}}$. Each query head still computes its own attention scores and weighted value sums, so both attention products scale with n_{heads}, not $n_{\text{kv-heads}}$.

For each query head, $QK^\top$ computes scores from projected queries and keys, and AV aggregates projected value vectors using the normalized attention weights A. Each product costs $2d_{\text{head}}$ per query–key pair. The GeGLU MLP has gate, up, and down projections. The encoder-layer cost is therefore

$$\begin{aligned} F_{\text{enc-layer}}(N, i) = {} & \underbrace{4N d_{\text{model}} d_{\text{head}} (n_{\text{heads}} + n_{\text{kv-heads}})}_{Q,K,V \text{ and output projections}} \\ & + \underbrace{4\mathcal{P}_{\text{enc}}(N, i) n_{\text{heads}} d_{\text{head}}}_{QK^\top \text{ scores and } AV \text{ value aggregation}} + \underbrace{6N d_{\text{model}} d_{\text{ff}}}_{\text{GeGLU MLP (gate, up, down)}}. \end{aligned} \tag{12}$$

The decoder uses the same components, plus K/V projections of the encoder states within its merged attention:

$$\begin{aligned} F_{\text{dec-layer}}(N, M, i) = {} & \underbrace{4M d_{\text{model}} d_{\text{head}} (n_{\text{heads}} + n_{\text{kv-heads}})}_{\text{decoder-state } Q,K,V \text{ and output projections}} \\ & + \underbrace{4N d_{\text{model}} d_{\text{head}} n_{\text{kv-heads}}}_{\text{encoder-state } K,V \text{ projections}} \\ & + \underbrace{4\mathcal{P}_{\text{dec}}(N, M, i) n_{\text{heads}} d_{\text{head}}}_{QK^\top \text{ scores and } AV \text{ value aggregation}} + \underbrace{6M d_{\text{model}} d_{\text{ff}}}_{\text{GeGLU MLP (gate, up, down)}}. \end{aligned} \tag{13}$$

There is no separate cross-attention module. Summing layers gives

$$\begin{aligned} F_{\text{encoder}}(N) &= \sum_{i=1}^{L_e} F_{\text{enc-layer}}(N, i), \\ F_{\text{decoder}}(N, M) &= \sum_{i=1}^{L_d} F_{\text{dec-layer}}(N, M, i). \end{aligned} \tag{14}$$

Adding the projection to vocabulary logits at the M output positions,

$$\begin{aligned} F_{\text{T5Gemma}}(N, M) = {} & F_{\text{encoder}}(N) + F_{\text{decoder}}(N, M) \\ & + 2M d_{\text{model}} n_{\text{vocab}}. \end{aligned} \tag{15}$$

G.3 GEMMA 3

Gemma 3 has a single decoder-only stack. Let N_{prompt} be prompt length, M_{gen} generated length, $T = N_{\text{prompt}} + M_{\text{gen}}$, and L_g, W_g its depth and window. Its layer cost has the same form as the T5Gemma encoder, with Gemma's own dimensions:

$$\mathcal{P}_g(T,i) = \begin{cases} T^2, & \text{global,} \\ T\min(T, W_g), & \text{local,} \end{cases} \tag{16}$$

$$F_{\text{Gemma-layer}}(T,i) = \underbrace{4Td_{\text{model}}d_{\text{head}}(n_{\text{heads}} + n_{\text{kv-heads}})}_{Q,K,V \text{ and output projections}} + \underbrace{4\mathcal{P}_g(T,i)n_{\text{heads}}d_{\text{head}}}_{QK^\top \text{ scores and } AV \text{ value aggregation}} + \underbrace{6Td_{\text{model}}d_{\text{ff}}}_{\text{GeGLU MLP (gate, up, down)}}, \tag{17}$$

$$F_{\text{Gemma3}}(N_{\text{prompt}}, M_{\text{gen}}) = \sum_{i=1}^{L_g} F_{\text{Gemma-layer}}(T,i) + 2M_{\text{gen}}d_{\text{model}}n_{\text{vocab}}. \tag{18}$$

There is no encoder or merged-attention term. Only output logits are counted. The full-square T^2 convention is an aggregate proxy, not permission for prompt tokens to attend to future generated tokens. It uses $N_{\text{prompt}} + M_{\text{gen}}$ positions rather than the $N_{\text{prompt}} + M_{\text{gen}} - 1$ inputs of exact cached generation. Local counts use compacted lengths, without original-position gaps or image-specific attention masks.

G.4 TOKEN PRUNING METHODS

G.4.1 GENERAL PRUNING BACKBONE

DTM, **FastV**, and **φ-R** share the following backbone computation for fixed original lengths, retained length, and pruning layer. Each method then adds its own selection cost.

Let k be the one-indexed layer *after which* pruning occurs: $1 \le k \le L_e$ for T5Gemma or $1 \le k \le L_g$ for Gemma. Thus pruning *after layer* k here places the gate *before layer* $k+1$ in the main-text convention (Figure 1); our L5 setting is $k = 5$ here. Write $\widetilde{N}$ for the retained encoder length. For total-input deletion rate δ, $\widetilde{N} = (1-\delta)N$, where $0 \le \delta < 1$. The shared T5Gemma computation, before adding a selection criterion, is

$$\begin{aligned} F_{\text{pruned-backbone}}(N, \widetilde{N}, M, k) = &\sum_{i=1}^{k} F_{\text{enc-layer}}(N,i) + \sum_{i=k+1}^{L_e} F_{\text{enc-layer}}(\widetilde{N},i) \\ &+ F_{\text{decoder}}(\widetilde{N}, M) + 2Md_{\text{model}}n_{\text{vocab}}. \end{aligned} \tag{19}$$

Gemma instead shortens the prompt during prefill from N_{prompt} to a retained length $\widetilde{N}_{\text{prompt}} = (1-\delta)N_{\text{prompt}}$; generated tokens are never removed. Define $\widetilde{T} = \widetilde{N}_{\text{prompt}} + M_{\text{gen}}$, analogously to $T = N_{\text{prompt}} + M_{\text{gen}}$ for the unpruned sequence. With

$$\begin{aligned} F_{\text{Gemma-backbone}}\left(N_{\text{prompt}}, \widetilde{N}_{\text{prompt}}, M_{\text{gen}}, k\right) = &\sum_{i=1}^{k} F_{\text{Gemma-layer}}(T,i) \\ &+ \sum_{i=k+1}^{L_g} F_{\text{Gemma-layer}}(\widetilde{T},i) \\ &+ 2M_{\text{gen}}d_{\text{model}}n_{\text{vocab}}. \end{aligned} \tag{20}$$

Empty sums are zero: pruning after the final encoder layer can still save T5Gemma decoder work, whereas pruning after Gemma's final layer saves no backbone computation. Fractional retained lengths are allowed for expected FLOPs.

In our visual-pruning experiments, only visual states are removed. The retained encoder or prompt length is therefore the unchanged text-token count plus the retained visual-token count; output lengths are unchanged.

G.4.2 DTM

DTM adds one scalar projection per gated input state, not another attention or feed-forward block. When gating all input tokens,

$$F_{\text{DTM}} = F_{\text{pruned-backbone}} + 2Nd_{\text{model}}, \tag{21}$$

$$F_{\text{Gemma3-DTM}} = F_{\text{Gemma-backbone}} + 2N_{\text{prompt}}d_{\text{model}}. \tag{22}$$

The gate runs once, including at zero deletion. Its small cost is retained, not rounded away. For visual-only DTM, the gate terms instead become $2N_v d_{\text{model}}$ for N_v visual encoder tokens or $2N_{\text{prompt},v}d_{\text{model}}$ for $N_{\text{prompt},v}$ visual prompt tokens. Text tokens are not gated; no gate runs when there are no visual tokens.

G.4.3 FastV and φ-R

Our FastV and φ-R baselines use *all-mean* importance: attention received by each eligible key is averaged over all query heads and all available input queries in layer k's attention pattern. Gemma uses prompt queries only, never future generated queries. We omit separate equations for other attention-aggregation variants, as we find their differences in total FLOPs negligible in the evaluated settings.

FastV reuses normal post-RoPE attention values and adds no dominant matrix multiplication. φ-R (ϕ_{-R}) reuses projected pre-RoPE Q/K (including Gemma's Q/K normalization), but adds one RoPE-free score contraction. It adds neither new projections nor a second value-aggregation product AV:

$$F_{\text{FastV}} = F_{\text{pruned-backbone}}, \tag{23}$$

$$F_{\phi_{-R}} = F_{\text{pruned-backbone}} + 2\mathcal{P}_{\text{enc}}(N,k)n_{\text{heads}}d_{\text{head}}, \tag{24}$$

$$F_{\text{Gemma3-FastV}} = F_{\text{Gemma-backbone}}, \tag{25}$$

$$F_{\text{Gemma3},\phi_{-R}} = F_{\text{Gemma-backbone}} + 2\mathcal{P}_g(N_{\text{prompt}},k)n_{\text{heads}}d_{\text{head}}. \tag{26}$$

Thus φ-R uses the original input's global or local pair count, not the retained length or Gemma's combined length T. FastV's zero *additional matmul* cost assumes reusable attention values; averaging and selection are omitted, not literally free. For visual-only pruning, only visual keys are eligible for retention, but the all-mean criterion still uses all available input queries. φ-R retains the full-input score contraction; a visual-key-only optimization is not assumed. Neither method runs a pruning criterion when no visual tokens exist.

G.5 Marginal visual-token FLOPs

Our gates target visual tokens, so we seek to quantify both the backbone cost attributable to processing those tokens and how much of that cost the gates remove. We therefore define *marginal visual-token FLOPs* as the difference between a method's cost with visual input and its cost on an otherwise identical text-only input. Comparing this marginal cost with the unpruned baseline then measures how much visual processing work the gate saves, rather than expressing savings only relative to the entire model.

For either model family, let N_t, N_v denote the numbers of text and visual input tokens, and M the output text length. For T5Gemma 2, the input is the encoder sequence and M is the decoder length. For Gemma 3, the input is the prompt and $M = M_{\text{gen}}$. For any method $\mathcal{M}$ applied to either backbone, define

$$F^{\mathcal{M}}_{\text{visual}}(N_t,N_v,M) = \underbrace{F_{\mathcal{M}}(N_t,N_v,M)}_{\text{with visual tokens}} - \underbrace{F_{\mathcal{M}}(N_t,0,M)}_{\text{text-only reference}}. \tag{27}$$

Here $F_{\mathcal{M}}$ is the whole-model cost for the chosen model family and method, using the visual-only pruning rules above. Both evaluations keep the model, text and output lengths, pruning layer, and requested visual-retention setting fixed. The text-only reference contains no visual tokens at any layer and skips visual pruning and its criterion, so it equals the text-only baseline.

This is a *marginal attribution*, not a separation into independent text and visual operations. For example, the global encoder pair increment is $(N_t+N_v)^2 - N_t^2 = N_v^2 + 2N_tN_v$: counting only

visual–visual pairs misses interactions with text. T5Gemma attribution also includes the decoder's attention to visual states and their K/V projections. For Gemma, it includes interactions between visual prompt states and text/generated states under the full-square proxy. Vocabulary FLOPs cancel; SigLIP and the multimodal projector remain excluded. The attribution includes visual-pruning overhead and still depends on text/output length; it is not the full cost of image processing.