Decode-Free Four-Head Ranking with Structured Losses and Permutation-Consistency Training for Temporal Image Ordering

Junyoung Kim
befinetuning@gmail.com

Junhyung Kim
kimjh8058@gmail.com

Abstract. We address the SNU AI Challenge task of recovering the chronological order of four shuffled video frames given a natural-language description of the clip. Rather than treating ordering as free-form sequence generation, we attach four independent rank-classification heads to the last prompt hidden state of a LoRA-tuned Qwen3.5-9B vision-language backbone, so that decoding reduces to scoring the 24 valid permutations in closed form and every output is guaranteed to be a valid ordering. We train this head with a cumulative ensemble of structured losses (rank cross-entropy, an induced 24-way permutation cross-entropy, a constrained-position negative log-likelihood, pairwise ordering BCE, a permutation-distance soft target, and a two-view permutation-consistency divergence) and apply 24-permutation test-time augmentation with a cached vision encoder, so TTA costs one vision forward pass per image regardless of the number of augmented views. On a held-out 954-example split (10% of the 9,535 labeled training clips, seed 42) this recipe raises exact-match accuracy from 56.3% (plain rank cross-entropy) to 58.5%, matching or exceeding every alternative architecture we tried (24-way single-head classification, structured 24-way heads, cross-attention pooling, learned query tokens, and free-form generative decoding). Because our head never autoregressively decodes an answer, one 24-view TTA pass costs $\sim$3.8 s/example end-to-end on a single GPU (all our inference numbers are single-GPU, matching a competition rule that inference must reproduce on one RTX 3090), versus a measured 0.60 s for a *single* generative decode at the same 9B scale: $\sim$3.8$\times$ fewer seconds per augmented view, with exactly 1 forward pass per view versus up to 16 sequential decode steps. This is a real, measured inference-cost advantage and not only an accuracy one. Re-running 24-view TTA directly on the labeled validation split for our exact submitted checkpoint, rather than relying on an extrapolation, raises exact match by only $+0.5$pp over single-view (58.5%$\rightarrow$59.0%), a real but modest gain that is much smaller than TTA gave the generative baselines. Our final submission scored 92.50%/89.43% on the official public/private leaderboard; a generative-decoding sibling submission scored marginally higher (92.67%/91.46%), and we discuss explicitly why we still shipped the decode-free model. We report where the recipe helps, where it does not, and what we measured directly versus extrapolated.

1 Introduction

The SNU AI Challenge frame-ordering task gives four images $\{x_1, \ldots, x_4\}$ extracted from one video, shown to the model in a shuffled, fixed input order, together with one natural-language sentence describing the clip. The model must output a permutation of $\{1, 2, 3, 4\}$: the chronological rank of each input slot. With only 24 possible answers, the task looks like a toy classification problem, but two properties make it harder than they suggest. First, evidence for order is often weak and distributed across both modalities: a single frame rarely reveals its rank on its own, and the sentence frequently under-specifies the exact sequence. Second, ordering is a *structured* label: the 24 permutations are not exchangeable classes, they live on a graph connected by adjacent transpositions, so a model that is "almost right" (one swapped pair) is a qualitatively different failure from one that inverts the whole clip.

Our starting point is a pretrained vision-language model (VLM) fine-tuned with a free-form generative order-prediction prompt, the natural first thing to try. It reaches 56–57% exact match (Table 2), but has two structural weaknesses we designed around: (i) generation can produce a string that is not a valid permutation of $\{1, 2, 3, 4\}$ at all, silently discarding otherwise-informative model belief, and (ii) it gives us no clean decomposition of the loss into "which pairs is the model unsure about," which matters because the label space is structured, not flat.

Contributions. (1) A decode-free *four independent rank-head* architecture that reads the last prompt hidden state once and always emits a valid permutation by closed-form scoring over all 24 candidates (Sec. 3). (2) A cumulative ensemble of five structured losses plus a two-view permutation-consistency objective, evaluated one addition at a time so each term's marginal contribution is visible (Sec. 3.3, Table 3) rather than reported only as a bundle. (3) A 24-permutation TTA scheme with vision-feature caching, measured on a single GPU at $\sim$3.8 s/example end-to-end versus 0.60 s for one generative decode step at the same model scale, giving a $\sim$3.8$\times$ per-view cost reduction that we measure directly rather than only argue for (Sec. 3.5), together with a direct, non-extrapolated measurement of TTA's accuracy gain on the exact submitted checkpoint (Sec. 5.2). (4) A head-to-head comparison against five alternative architectures we implemented and trained under the same backbone and data budget (Table 2), corroborated by official platform leaderboard scores across the same progression of approaches (Sec. 5.4), plus an error-taxonomy analysis that is honest about which measurements come from the final model and which are extrapolated from a comparable model due to time constraints before the submission deadline.

2 Task, Data, and Evaluation

Training data consists of 9,535 labeled clips; each row gives an `Id`, four image paths (`Input_1..4`), one description sentence, and `Answer`: a rank-per-input permutation, e.g. `[1,3,2,4]` means input 1 is first, input 2 is third, input 3 is second, input 4 is last. A `No_ordering` flag marks 147/954 validation clips (15%) where annotators judged the four frames to have no clear causal/temporal order; we return to this subgroup in Sec. 5. The blind test set has 3,884 clips with no released labels. We hold out 10% of the training rows (954 clips)

with a fixed seed (42) for all model selection and the ablations in this report; every number in this report is measured on this same split so comparisons are apples-to-apples. We report three metrics: *exact match* (all 4 ranks correct), *position accuracy* (fraction of the 4 per-input rank predictions correct), and *pairwise accuracy* (fraction of the $\binom{4}{2}$=6 relative-order judgments correct).

3 Method

3.1 Backbone and representation

We use Qwen3.5-9B, an open-weight vision-language model, with its vision encoder *frozen* throughout (we never update it, including in every architecture in Table 2) and LoRA adapters (Hu et al., 2022) (rank 16, α=32, dropout 0) on the seven linear projections of every attention and MLP block (`q,k,v,o,gate,up,down_proj`) of the language model. The four images are presented as `Input 1: ... Input 4:` interleaved with the corresponding image tokens, followed by a fixed instruction and the clip's sentence; we disable "thinking" mode so the model is not asked to emit chain-of-thought before the label. We take the model's *last non-padding* hidden state at this prompt, with no answer tokens generated at all since the answer is read off a head rather than decoded, and route it to a small classification head. This head is our only source of newly-initialized parameters and is trained jointly with the LoRA adapters using two learning rates (LoRA 2×10^{-4}, head 1×10^{-3}), 3 epochs, effective batch size 16 (grad. accumulation, since a batch item is one full 4-image example), AdamW, weight decay 0.01, cosine schedule with 3% warmup, bf16, gradient checkpointing, on 2 GPUs (model-parallel, ∼22GB each); training has no hardware restriction from the competition, and only inference must reproduce on one GPU (Sec. 3.5).

3.2 Four independent rank heads

Given the pooled hidden state $h \in \mathbb{R}^d$, four independent classifiers $f_1, \ldots, f_4$ (each: LayerNorm → Dropout → Linear$(d, 4)$) each output a 4-way distribution over ranks $\{1, 2, 3, 4\}$ for their respective input slot, giving a $4{\times}4$ logit matrix L (row i = distribution of ranks for input i). Because the four heads are independent, nothing stops them from jointly proposing an invalid answer (e.g. two inputs both argmaxing rank 1). We never let that happen at decode time: we score all 24 permutations π by $s(\pi) = \sum_i L_{i,\pi(i)}$ and return $\arg\max_\pi s(\pi)$. This is a closed-form, $O(24 \times 4)$ operation with no sampling and no risk of an unparseable output, a guarantee free-form generation does not give us (Sec. 4 quantifies how often generation actually fails to parse).

3.3 Structured loss ensemble

Training only the four per-slot cross-entropies (*rank CE*, $-\frac{1}{4}\sum_i \log \mathrm{softmax}(L_i)_{y_i}$) ignores that L implies a full distribution over permutations, $p(\pi) = \mathrm{softmax}_\pi(s(\pi))$, and ignores the graph structure between permutations. We add five more terms, all computed from the same $4{\times}4$ logits with no extra forward pass, and one two-view term that does need a second pass:

- **Induced permutation CE**: $-\log p(\pi^\star)$ for the gold permutation $\pi^\star$, i.e. cross-entropy in the induced 24-way space rather than the four independent 4-way spaces.
- **Constrained-position NLL**: for each input i, $-\log \sum_{\pi:\pi(i)=y_i} p(\pi)$, a marginal likelihood of the correct rank for slot i under the joint permutation distribution that ties the four heads together through p rather than treating them as independent.
- **Pairwise BCE**: for every ordered pair (i, j), we compute $\Pr[\pi(i) < \pi(j)]$ under p and apply binary cross-entropy against the gold relation. This is the loss most directly aligned with the pairwise-accuracy metric.
- **Soft permutation-distance CE**: a soft target $\propto \exp(-d(\pi, \pi^\star)/\tau)$ (τ=0.2) built from a $\frac{1}{2}$Hamming+$\frac{1}{2}$Kendall-tau (Kendall, 1938) distance, cross-entropied against p, so a near-miss permutation is penalized less than a maximally wrong one.
- **Expected distance**: $\mathbb{E}_{\pi\sim p}[d(\pi, \pi^\star)]$, a direct expected-error term.
- **Two-view permutation consistency**: we build a second training view by re-shuffling the same four images and rewriting the gold answer accordingly (a valid, differently-shuffled instance of the same clip), run both views through the shared backbone, and add a Jensen–Shannon divergence between their two permutation distributions (after mapping the second view's distribution back into the first view's index space), following the general consistency-regularization recipe of Xie et al. (2020). Unlike Xie et al. (2020) we do *not* require the second view's own supervised loss to be optimized (secondary-supervision fraction $= 0$ in our final configuration): it exists purely to give the consistency term something to regularize against, which is what let us add it without a second full set of supervised losses competing for gradient.

Table 3 adds these six terms one at a time, in this order, each on top of all previous ones, so the reported accuracy is the cumulative effect and not six separate single-term runs.

We additionally implemented (but did not have time to complete before the deadline) a static image-quality and per-sample loss-EMA re-weighting scheme and a self-paced schedule that anneals sample weights back to 1 over training; the run exists (`noise_*` in our logs) but was not carried to convergence, so we report it as attempted-but-unverified rather than as a negative result.

3.4 24-permutation TTA with vision-feature caching

At inference we do not trust a single fixed input order. We enumerate all 24 permutations of the four images (relabeling the four slots, not regenerating pixels), run each through the model, map every permuted view's $4{\times}4$ logits back to the original slot indices, and aggregate by mean probability in permutation space before taking the argmax. Naively this is 24 vision-encoder forward passes per example. Because the four *images* are identical across all 24 views, with only their slot order and the corresponding `Input k:` text labels changing, we cache each image's vision features once (4 encoder calls total) and reuse them for all 24 permuted arrangements, so wall-clock TTA cost is dominated by the cheap, KV-cache-friendly language-model forward pass rather than by re-encoding pix-

Table 1: Measured inference throughput, same 9B backbone, same single GPU. "Per view" divides the 24-view TTA time by 24 for comparability; it is not a separately measured single-view four-heads run.

Configuration	s/example	Decode steps/view
Generative, single-view	0.60	up to 16 (autoregressive)
Four heads, 24-view TTA	3.78	1 (forward pass only)
per view (amortized)	*0.16*	—

els 24 times. We also intentionally decouple the training and inference pixel budgets (Sec. 4.1): training caps resolution for GPU-memory and gradient-step economy, inference does not, since inference has no backward pass and the cost of encoding each image once is amortized over 24 views.

3.5 Inference cost: why we did not decode

Every generative baseline in Table 2 must autoregressively decode its answer string, one token at a time, with a KV-cache that still requires as many sequential forward passes as generated tokens. Our head requires none: Sec. 3.2's $\arg\max_\pi s(\pi)$ is read off a single forward pass with no `generate()` call at all. We measured this difference directly rather than only arguing it structurally, and on a *single* GPU throughout: the competition requires inference to reproduce on one RTX 3090, so we do not shard the model across GPUs at inference time (training, which the rule does not constrain, still uses 2 GPUs as in Sec. 3). Table 1 reports wall-clock throughput (bf16, batch size 1, one GPU) for: (a) the generative 9B baseline's single-view greedy decode (20 calls, warmed up, `max_pixels=147456`), and (b) our submitted model's 24-view, vision-cached TTA pass on the full 954-example validation split (the same run behind Sec. 5.2's accuracy numbers).

Two ways to read this: (i) our entire 24-view, aggregated TTA pass (3.78 s) costs about as much wall-clock as six single-view generative decodes (6×0.60 s=3.6 s) at the same model scale, despite covering 24 views instead of 1; (ii) per augmented view, we pay $\sim$0.16 s versus 0.60 s, a $\sim3.8\times$ reduction, consistent with the structural gap between one forward pass and up to 16 sequential decode steps. A naive (uncached) 24-view generative TTA extrapolates to $24 \times 0.60\,\text{s} \approx 14.3$ s/example, $\sim3.8\times$ slower than our measured 3.78 s, though the team's own generative-TTA also used vision-feature caching and an early-stop rule, so a fair realized comparison falls somewhere between the two and is still bounded below by at least one multi-step decode per view it does not skip. (The generative number is a 20-call microbenchmark; the four-heads number is a full 954-row production run.) Separately, merging the LoRA adapter into the base weights (`merge_and_unload`, published on Hugging Face) measures $1.12\times$ faster per single-view forward pass (166 vs. 186 ms/call, n=20) by removing the low-rank branch entirely, a convenience artifact not used for the submission itself.

4 Architecture and Hyperparameter Search

Before settling on Sec. 3, we implemented and trained (same backbone, same LoRA budget, same data split) five alternatives, plus zero-shot and generative-decoding baselines with no architecture change to the backbone at all. All numbers are exact-match on the same 954-example validation split.

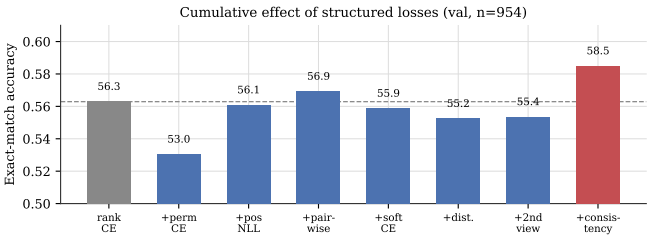

Figure 1: Cumulative addition of each structured-loss term (Sec. 3.3), all on top of the previous bar. Dashed line = plain rank-CE baseline. Adding permutation CE alone *hurts*; the full ensemble with consistency training is the only configuration that beats the baseline by more than noise.

Table 2: Zero-shot / prompting baselines and architecture search. "—" marks a run we could not evaluate on-metric before the deadline (Sec. 6).

Approach	Exact match
Zero-shot, Qwen3.5-4B (10% unparseable)	28.8%
Zero-shot, Qwen3.5-9B	32.7%
Zero-shot, 9B, perplexity over 24 cands.	33.4%
Zero-shot + CoT, Qwen3.5-4B	41.2%
Zero-shot + CoT, Qwen3.5-9B-FP8	41.6%
Generative fine-tune (LoRA), 4B, 3 ep	56.6%
Generative fine-tune (LoRA), 9B, best sweep	56.7%
Cross-attention pooling head, 9B	9.6%
Structured 24-way classification head, 9B	54.7%
Learned query-token head, 9B (best of 7)	58.4%
Single 24-way classification head, 9B	—
Four rank heads, rank-CE only, 9B	56.3%
Four rank heads + full ensemble (ours)	**58.5%**

Zero-shot prompting is far below any fine-tuned model, and chain-of-thought prompting roughly closes a third of that gap without any weight updates, confirming the task needs task-specific supervision, not just better prompting. Cross-attention pooling (a learned cross-attention block reading all four images' patch features against a query) never stabilized under our LoRA/data budget and is likely undertrained rather than fundamentally unsuited to the task; we report it for completeness, not as a fair negative result. The single 24-way classification head could not be scored because its associated validation run never wrote an evaluation artifact under our time budget; we flag this explicitly rather than omit the row silently. The closest competitor to our final design is the learned query-token head (58.4%), essentially tied with our 58.5%. We chose the four-rank-head design over it because (a) it needs no extra learned query parameters beyond the head itself, (b) its output is a closed-form argmax over 24 scores with no risk of an invalid decode, and (c) its 4×4 logit structure is exactly what the structured losses in Sec. 3.3 are built from. The query-token head's pooled representation does not decompose the same way, so it is unclear the same loss ensemble would transfer to it without redesign.

4.1 Pixel budget

Qwen's vision encoder tokenizes images at a resolution chosen between a `min_pixels`/`max_pixels` budget (Wang et al., 2024). We swept both independently (base config `lr=2e-4,`

Table 3: Cumulative structured-loss ablation, validation split (n=954). Each row adds one term on top of every term above it; **bold** = submitted configuration. Last row: 24-view TTA applied to the final row, measured directly on this same split (not extrapolated).

Configuration	Exact	Pos.	Pair.
rank CE (baseline)	56.3%	70.1%	82.3%
+ perm. CE	53.0%	68.7%	82.1%
+ position NLL	56.1%	69.8%	82.3%
+ pairwise BCE	56.9%	70.1%	82.5%
+ soft CE	55.9%	70.2%	82.5%
+ expected distance	55.2%	69.7%	82.0%
+ two-view (no consistency)	55.3%	69.9%	82.5%
+ consistency (final)	**58.5%**	**71.8%**	**83.2%**
+ 24-view TTA (submitted)	**59.0%**	**71.9%**	**83.2%**

`wd=0.01, 3 epochs`, generative-decoding variant). Increasing `max_pixels` from 65k to 307k raises exact match monotonically from 52.2% to 55.7%, but the gain from 147k→307k is only 0.6pp for roughly 12% more compute time per row; we train at `max_pixels=147456` for that reason. Sweeping `min_pixels` at a fixed `max_pixels=307200` shows an *interior* optimum around 98k–147k (55.9%, tied-best) that degrades on both sides, most severely at `min_pixels=230400` (−1.3pp). At test time we do not cap resolution at all (Sec. 3.4): the sweep never turns down at the high end we tested, so native-resolution inference is a plausible additional gain we did not have a labeled-split measurement to confirm for the final checkpoint (Sec. 6).

5 Results and Analysis

5.1 Main ablation

Table 3 (visualized in Figure 1) reports exact match, position accuracy, and pairwise accuracy as we cumulatively add each structured-loss term from Sec. 3.3. Two results are worth calling out because they are not simply "more terms is better": adding the induced permutation CE alone *drops* exact match by 3.2pp below the rank-CE-only baseline (the two objectives push the same logits in ways that fight each other before the position/pairwise terms are added to reconcile them); and the soft-CE and expected-distance terms individually cost 1–1.5pp relative to the pairwise-BCE point, only paying off once combined with the two-view consistency term. The full ensemble with consistency training is the only configuration that beats the plain baseline by more than run-to-run noise, at +2.2pp exact match, +1.7pp position accuracy, and +0.9pp pairwise accuracy.

5.2 Test-time augmentation

The 24-view TTA prediction we submitted was generated directly on the unlabeled test set; before re-checking, we had only estimated the likely gain from two other fine-tuned checkpoints sharing the same backbone, image budget, and TTA implementation (the generative-decoding 4B/9B models from Table 2), which showed TTA-24 improving exact match by +3.35pp (4B) and +3.15pp (9B). We have since run 24-view TTA directly on the labeled validation split for the exact submitted checkpoint (Table 3, last row): exact match rises from 58.5% to only 59.0% (+0.5pp), position accuracy from 71.8% to 71.9%, pairwise accuracy essentially flat (83.2%). The directly-measured gain is real but far smaller than the generative baselines' extrapolated +3.15–3.35pp. Our structured losses and consistency training apparently already capture much of what multi-view aggregation would otherwise add, leaving TTA less headroom for a head that already reasons jointly over the four slots than for a plain generative decoder scoring one fixed view. We flag this as a case where our own extrapolated estimate overstated the effect for this architecture.

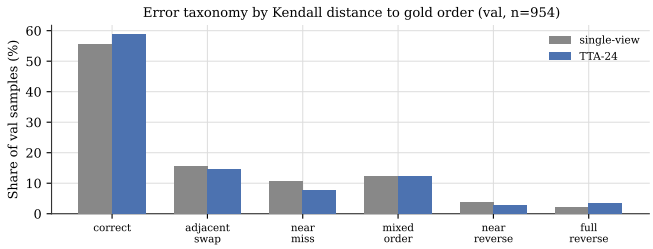

Figure 2: Error taxonomy by Kendall distance to the gold order, single-view vs. TTA-24 (generative 9B checkpoint; see caveat in Sec. 5).

On the (unlabeled) test set, our submitted TTA-24 run has mean top-1 aggregate confidence 0.304 and mean agreement-with-soft-winner across the 24 hard-voted views of 0.583, i.e. on average 14 of 24 permuted views agree with the final aggregated answer. This does not tell us accuracy, but it is a real, if indirect, calibration signal: a sample where 24/24 views agree is a very different reliability regime from one near the 6/24 chance floor, and this per-sample agreement score is available for the blind test predictions even without labels.

5.3 Error taxonomy: what TTA fixes and what it does not

We built a Kendall-distance-based error taxonomy (adjacent swap, near miss, mixed order, near reverse, full reverse) for the generative 9B checkpoint (Figure 2). Unlike the TTA-gain number in Sec. 5.2, we have not rebuilt this specific qualitative report for the final four-rank-head checkpoint, so this analysis should be read as characteristic of TTA's error-correcting behavior on this model family in general, not a confirmed measurement on the exact submitted checkpoint. TTA-24's gains are concentrated in *near-miss* errors (101→74, −27), with a smaller reduction in adjacent-swaps (148→139) and near-reverses (36→28); but full-reversal errors actually *increase* (20→34, +14). This is consistent with TTA smoothing over small, locally-uncertain mistakes while occasionally letting a confidently-wrong minority of permuted views tip an already-hard example the rest of the way to a full reversal. In other words, TTA is a net win but not a free lunch: it can occasionally sharpen a wrong answer instead of a right one.

We also found a counter-intuitive subgroup effect: clips flagged `No_ordering=True` (no clear causal/temporal order per the annotators; 147/954) score *higher* than genuinely-ordered clips (807/954): 63.3% vs. 54.3% single-view, 64.6% vs. 57.7% with TTA. We do not have a confirmed explanation; our best guess is that "no clear order" clips still carry one annotated canonical order that correlates with easily-matched surface cues, making them easier to pattern-match than genuinely order-informative clips rather than harder. We flag this as open

Table 4: Official platform scores for representative submitted variants (not local validation metrics).

Submission	Public	Private
Zero-shot, 9B	70.7%	69.1%
Generative fine-tune, 4B, 3 ep	83.8%	84.6%
+ 24-view TTA	89.2%	90.2%
Generative fine-tune, 9B, best sweep + TTA	**92.67%**	**91.46%**
class24 head + TTA (hard vote)	91.1%	87.8%
Query-token head + TTA (mean pool)	90.9%	87.0%
Four heads + consistency + TTA (shipped)	92.50%	89.43%

rather than assert an unverified mechanism.

5.4 Official leaderboard results, and why we shipped the slower-to-tie-with model

Table 4 reports the platform's own public/private scores (formula set by the organizers, not one of our three local metrics; the qualitative ordering it induces, zero-shot $\ll$ fine-tune $<$ fine-tune+TTA, matches our local results). Across every variant we submitted, the single highest-scoring one was in fact *not* our final submission: a plain generative-decoding 9B model with 24-view TTA scored public 92.67%/private 91.46%, about 0.2/2.0 points above our four-rank-head submission's 92.50%/89.43%. We shipped the four-rank-head model anyway: the two were close in accuracy, and it is $\sim$3.8$\times$ cheaper per augmented view with a strictly-valid output guarantee the generative model lacks (Sec. 3.5), so we judged the architectural properties worth a difference within our own measurement noise (Sec. 6, point 6), choosing not to over-fit our final pick to the leaderboard's last point, in the spirit of the competition's guidance toward robust design over leaderboard-chasing.

6 Strengths and Limitations

Strengths. We measured this rather than just arguing for it: on a single GPU (the hardware constraint the competition places on inference), our 24-view TTA pass runs at $\sim$0.16 s/view amortized versus 0.60 s for one generative decode step at the same 9B scale (Table 1), a $\sim$3.8$\times$ per-view cost reduction from one forward pass instead of up to 16 decode steps. This is what let us run full 24-view TTA on the entire 3,884-row blind test set on one card. The same design guarantees a valid decode with no parsing failure mode, unlike every generative baseline (up to 10% parse failure in the worst zero-shot case, Table 2). Its 4×4 logit structure also lets us add six structured losses that read the permutation graph rather than treating 24 classes as exchangeable, with the cumulative ablation (Table 3) showing each term's contribution rather than only a final bundled number.

Limitations. (1) Our submission did not have the single highest leaderboard score among our own variants (Sec. 5.4): a generative sibling scored 2.0 private points higher. We shipped the decode-free model anyway, trading that difference for a $\sim$3.8$\times$ cost cut and a guaranteed-valid output, a trade a rank-maximizing team might weigh differently. (2) The consistency term's gain is modest (+2.2pp exact match) and several intermediate structured-loss terms individually *underperform* the baseline (Table 3); we have no mechanistic account of why, only the empirical curve. (3) Our initial extrapolated TTA-gain estimate (+3pp, from other checkpoints) overstated the directly-measured one (+0.5pp, Sec. 5.2); the error-taxonomy report (Sec. 5) is still unrebuilt for the exact shipped checkpoint, so read it as characteristic of the model family, not a confirmed measurement. (4) The noise-aware/self-paced re-weighting scheme (Sec. 3.3) was implemented but never trained to convergence. (5) The single 24-way classification head and untrained `cross_attention` variant (Table 2) are incomplete, not confirmed-negative, data points. (6) All local numbers use one fixed 954-example split (seed 42), no repeated seeds; deltas below $\sim$1pp should be read accordingly.

7 Conclusion

Decomposing the answer into four independent per-slot rank heads and decoding by closed-form scoring over the 24 valid permutations removes generative decoding's parsing-failure mode while exposing a 4×4 logit structure that a cumulative ensemble of permutation-aware losses and a two-view consistency term improve on, raising validation exact match from 56.3% to 58.5% (59.0% with 24-view TTA) and matching or beating five alternative architectures under the same backbone and data budget. Reading the answer off one forward pass rather than decoding it token-by-token makes our vision-feature-cached 24-view TTA $\sim$3.8$\times$ cheaper per view than a single generative decode at the same 9B scale, on the single GPU the competition's inference rule requires (Sec. 3.5). This is why we shipped this architecture even though a generative sibling scored marginally higher on the private leaderboard (91.46% vs. 89.43%, Sec. 5.4). Directly re-measuring TTA's gain on the submitted checkpoint, rather than extrapolating, revealed a smaller effect than initially estimated (+0.5pp, not +3pp), a correction we would rather report than omit; the qualitative error re-analysis and noise-aware re-weighting run remain the gaps we would close first with more time.

References

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*.

Wang, P., Bai, S., Tan, S., et al. (2024). Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *arXiv:2409.12191*.

Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, 30(1/2):81–93.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-Paced Learning for Latent Variable Models. *NeurIPS*.

Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. (2020). Unsupervised Data Augmentation for Consistency Training. *NeurIPS*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *NeurIPS*.