SNU AI Challenge 2026 · Final Report

Ensemble-Like Gains from a Single Model: Threshold-Gated Test-Time Cascades for Video Frame Ordering

Taeyoun Kwon[1,2] Seungjin Kim[1,2]
[1]Maum AI Inc. [2]Seoul National University
ty8352@snu.ac.kr starsj59@snu.ac.kr

Abstract

Given four video frames in shuffled order and a caption narrating them in chronological order, the task is to recover the original frame order. Zero-shot accuracy saturates at 27B in the non-thinking mode we deploy: a model fifteen times larger in total parameters is worth −0.3 EM. Supervised finetuning carried it from 43.8 to 62.1 EM. After that the memorized half of the split never moved again, and every later gain landed in the rest: 0.9 from reinforcement learning, and 1.8 more from inference-time computation on the same frozen weights. Finetuning placed no loss on the thinking block, so the base model's thinking mode survived untrained and supplies those votes. Following their plurality loses accuracy outright; replacing an answer only past a majority threshold turns the same votes into a gain, and makes early stopping provably equivalent to full voting. Votes drawn per item fall from 8 to 3.2, which is what puts three channels from a single weight set on a single RTX 3090 inside the 24-hour limit. The entry placed fifth on the private split. Code, weights, and the reference logs every number here is computed from are at https://github.com/sdfbiasdf/SNUAI.

1. Introduction

Given four frames of a video in shuffled order and a caption that narrates the events in chronological order, the task is to recover the original order of the frames.

Before training anything, we measured how far open-source frontier models already take it. Zero-shot on DEV1000, Qwen3.5-9B [6] scores 22.0 EM without thinking and 43.0 with it, and 27B scores 43.8 and 46.8. A 397B-A17B model with about fifteen times the total parameters scores 43.5 and 50.1. In the non-thinking mode we go on to deploy, fifteen times the parameters is worth −0.3 EM: **scale has already saturated at 27B**. With thinking it still buys 3.3, so what headroom scale leaves is in deliberation rather than in parameters, and that is where our inference-time design goes to collect it.

Measuring the data next changed what we thought we were measuring. Frames recur across items, to the point that 449 of the 1,000 items in our development split have all four frames byte-identical to frames inside training items and can therefore be answered from memory. We report every number decomposed into that segment (dup) and the rest (nondup), treating nondup as the honest estimate of generalization, and we corrected 58 wrong official labels along the way (§2).

Training stops paying, and inference does not. Supervised finetuning raised the development split from 43.8 to 62.1 EM. We trained in non-thinking mode but placed no loss on the empty thinking block, so the base model's thinking mode survived intact. GRPO [7] then added 0.5 EM, all of it inside nondup (§3.1). There the returns on training and on inference-time computation separated: the GRPO gain is not distinguishable from noise, whereas leaving the weights untouched and spending computation at inference was worth 1.2 EM.

The mechanism is that we used thinking to produce votes rather than answers. A single thinking vote scores 57.9 against non-thinking's 62.6, yet the correct answer sits somewhere in the votes for 75.7% of items. Following the plurality [8] overwrites items already answered correctly and loses accuracy outright, whereas replacing only past a majority threshold turns that loss into a gain, and a majority threshold also lets votes be drawn one at a time and stopped early with provably identical results. A third pass re-harvests items whose captions carry no temporal cue word. These three channels come from a single weight set (§3), quantized so that the cascade runs on one RTX 3090 within 24 hours (§4). The entry scored 0.94066 on the public leaderboard and 0.93495 on the private split, where it placed fifth.

Contributions.

1. **We established what we were measuring before measuring it.** Frames recur across items, so no clean split exists; we report dup and nondup separately throughout, and corrected 58 labels, screening candidates with a model that had memorized the data.
2. **We trained the answer and kept the reasoning.** The rules bar training on generated reasoning traces, closing the usual cold-start route to a thinking model. Finetuning therefore never touched the thinking block, leaving the base model's thinking mode usable as a second channel.
3. **Ensemble-like gains without an ensemble.** Three channels from one weight set, diversified by reasoning mode and then by prompt form, replace an answer only past a majority threshold. The gain comes from the gate rather than the vote, and early stopping is provably equivalent to full voting, which makes a 2.5× saving free.

2. Data Analysis and Curation

2.1. Making the evaluation trustworthy

Splits. From the 9,535 official training items we drew 500 at random as VAL500, a further 1,000 from what remained as DEV1000, and trained on the other 8,035, using DEV1000 for training evaluation and checkpoint selection. VAL500 was withheld from those sweeps: no checkpoint, threshold, or prompt was ever chosen using it, and it was read only once those choices were frozen (§5). It is not a set we never looked at, since 35 of its items went through the label audit below, and it later served as one of the two criteria for choosing the final entry.

The training data is not independent of the holdouts. Hashing all 38,140 frames shows the same frame recurring across items: in the graph linking items that share a frame, the largest connected component reaches 4,293 items, 45% of the set. As a result 449 of the 1,000 items in DEV1000 have all four frames byte-identical to frames belonging to some training item.

We therefore report dup and nondup separately. An item whose frames were seen in training can be answered by recalling its order, so a single average blends memorization with generalization. We split every accuracy number into items whose four frames all appear in the training data (dup, 449) and items where they do not (nondup, 551), treating nondup as the honest estimate of generalization. The decomposition is what later localizes the GRPO gain: across training, dup moved only between 404 and 405 of its 449 items, and all variation occurred in nondup (§3.1).

2.2. Making the training data trustworthy

A bias built into the labels. The answer distribution is skewed from the start. Of the 9,535 items, 1,478 (15.5%) are marked `No_ordering` and the answer for every one of them is the input order `[1,2,3,4]`, while the other 23 permutations are spread evenly at 365 to 386 occurrences each. Against a uniform expectation of 397, identity is oversampled 3.7×: a model that learns nothing and always answers `[1,2,3,4]` floors at 15.5%, and training has an incentive to drift there. To counter it we added one frame-permuted copy of every item, expanding the 8,035 training items to 16,070 rows.

How we selected label suspects. While inspecting items the model repeatedly got wrong, we found cases where the official answer itself was wrong. Auditing all 9,535 by hand was not possible, so we needed a criterion: *an item whose official answer is still refused by a model that has memorized the training data.* A model simply being wrong is not a signal, but a model that has already seen an item's answer during training and rejects it at inference is in conflict with the label. We operationalized this by shortlisting items where the non-thinking answer differs from the official label *and* at least 12 of 16 thinking votes converge on a single answer that is not the official one; a human made the final call on the frames.

What adjudication returned. Of 191 candidates we reviewed 135 and corrected the answers of 58, judging the official answer correct in 74 cases and several orders equally valid in 3. Adding to those 3 the 32 items containing byte-identical frames within a single item, whose order is therefore undetermined, gives 35 items with no unique order. For these we trained SFT on the single official answer but credited the entire set of valid answers in GRPO.

3. Method

3.1. Two-stage training

Why we trained the answer and not the reasoning. This task rewards reasoning: zero-shot on DEV1000, thinking led non-thinking by 46.8 to 43.8 at 27B, and by 21.0 pp at 9B. That lead sits on the memorized segment, where thinking gains 8.5 pp while losing 1.4 on the rest (Table 3), so what the mode supplies is not a better single answer but a second and diverse view of the same item, which §3.3 spends as votes. The natural next step is to train a thinking model. The standard route to one opens with a cold-start SFT stage on reasoning traces, but the rules prohibit training on traces produced by a generative model, whether obtained by

teacher distillation or by self-generated rejection sampling. Running GRPO on thinking with no cold start is permitted, but it starts from an untrained trace distribution where reward variance across rollouts is high, and that did not fit our compute budget.

So we inverted the design: train the answer only, leave the thinking block untouched so the base model's thinking mode is preserved rather than degraded, and spend that mode at inference, where test-time augmentation is permitted.

Finetuning, and the transfer to a channel we never trained. We fully finetuned the language backbone in non-thinking mode only, freezing the vision tower and the aligner, using ms-swift [9]. No loss was placed on the empty thinking block, so the training signal fell on the answer alone and *the reasoning trace was never a training target.* Under non-thinking inference this raised DEV1000 EM from 43.8 to **62.1** (+18.3).

The design works: the thinking channel rose as well, from 46.8 to **57.9** (+11.1), despite never being trained. We improved it without ever learning a reasoning trace. Its gain is smaller, so non-thinking leads after finetuning, but this preserved channel is why one weight set yields two reasoning modes (§3.2).

Reward design for GRPO. Rewarding exact match alone makes an answer with two adjacent frames swapped score identically to a completely reversed one, so the signal cannot separate *nearly right* from *entirely wrong.* We added the fraction of the six frame pairs placed in the correct relative order [4] as an auxiliary term; the reward sums exact match, this pairwise credit, and format compliance in the ratio 1.0 : 0.5 : 0.2. The 35 items admitting several correct orders receive full credit for any answer in their valid set: SFT trains under teacher forcing and must choose one target, whereas GRPO scores sampled outputs and can express a set without contradiction.

The limit of further training. GRPO, run as a LoRA [3] of rank 32 and merged back into the weights, raised DEV1000 EM from 62.1 to **62.6** (+0.5). The gain came *entirely from nondup*, while dup held at 90.2. The same held across the whole run: over 15 checkpoints spanning 1,400 steps, dup stayed inside a 0.2 pp band (90.0–90.2) while nondup moved over 2.0 pp (38.1–40.1). Whatever training had left to take was in nondup alone. On VAL500, however, every checkpoint of the run lands between 62.2 and 62.8 and step 600 is not the peak (§5), so the +0.5 is a property of the split we selected on. We took that as the point where training had stopped paying, and redirected the remaining budget into inference design.

Ch.	base	prompt	think	decoding	scope
1	—	plain	×	single sample	all
2	ch. 1	plain	✓	8 votes, thr. 6	all
3	ch. 2	observe-first	✓	8 votes, thr. 5	cue0

Table 1. Channel configuration. Channel 1 draws a single sample at a fixed seed. Each channel takes the previous channel's answer as its base.

3.2. Test-time cascade

One weight set yields three channels.

The three channels are the same weights run in different ways, and the axis of diversity differs at each step: from channel 1 to 2 the *reasoning mode* changes (non-thinking to thinking), from 2 to 3 the *reasoning style requested by the prompt.* They stack by **base-and-overwrite**, each channel taking the preceding answer as its base and overwriting it only once its own votes have accumulated sufficiently.

3.3. Threshold voting and early stopping

The gate. Among the 374 items non-thinking gets wrong, 41.4% carry the correct answer somewhere in the thinking votes, so there is something to extract. The problem is how. Adopting the plurality recovers 55 of them but loses 80 of the 626 that non-thinking already had right, and DEV1000 falls from 62.6 to **60.1**, a drop of 2.5 pp.

So we use a threshold instead. A non-base answer replaces the base only when it reaches 6 of 8 votes. Because the threshold is a strict majority, two answers cannot reach it simultaneously, and a split vote leaves the base standing. *Holding the votes fixed and changing only the rule turns* -2.5 *pp into* $+0.6$ *pp.*

Other aggregation rules do not recover it either: Borda count and pairwise re-aggregation both produce a net loss at every margin band we tried, and letting the model judge its own candidates, best-of-n with self-verification, scores 31.8 against the 39.4 of plain majority voting on the same items.

The gain comes from the gate, not from the vote.

Early stopping. There is no need to draw all eight votes on every item. *Drawing votes one at a time and stopping the moment the outcome is settled yields exactly the same answer as full voting*, guaranteed by the threshold being a strict majority ($2t > n$): once a non-base answer reaches t no other one can, and once even the best non-base answer satisfies *current* + *remaining* $< t$ none can. We verified this by exhaustively enumerating every possible vote sequence. Stopping self-consistency early has been studied before [1, 5], but those criteria read the accumulated confidence and so change the answer distribution; a

strict-majority gate is what makes stopping exact here. Votes consumed per item fell from 8 to an average of **3.2** with no change in the answers (§4).

3.4. Caption-cue routing

After channel 2, **368 items of dev1000 remain wrong**. Inspecting them, we found one property that stood out. Items whose caption contains no temporal cue word make up 40.5% of the errors but only 11.2% of the correct answers. Their captions are also shorter, 18.6 words against 27.9 on average.

If the caption does not convey order, the item must be solved from the visual evidence alone, so on this segment we harvested votes again under a prompt that first makes the model enumerate what is visible in each frame. The segment, cue0, is the captions containing none of twelve temporal cue words (`then`, `after`, `until` and so on; full list in the code).

Applied to the cue0 segment, this channel **fixed six items and broke none**, lifting DEV1000 from 63.2 to **63.8**.

4. Deployment under Constraints

The submitted pipeline must process all 819 test items on a single RTX 3090 (24 GB) within 24 hours. A bf16 27B model does not fit in that memory, so quantization was a precondition rather than an optimization. We quantized to a Q5_K_M GGUF with imatrix calibration and kept the vision encoder at F16, running on llama.cpp [2] at a pinned commit. Table 2 gives the resulting budget.

Memory and latency. Peak VRAM reaches 23,104 of 24,576 MiB under the deployment setting (`--parallel 4 -c 53248`). Most of the generated tokens belong to the thinking channel, and early stopping is what makes that affordable, cutting the votes drawn per item from 8 to an average of 3.2 (§3.3); the third channel adds to this only on cue0 items. From the 58.8 tok/s measured on a physical RTX 3090 and the generated tokens of the shipped run, the three channels take 21.2 hours over the 819 items, inside the limit. Slot count was set by latency, not by memory: at `--parallel 1` the run is bit-reproducible but decodes at 35.2 tok/s, at `--parallel 2` it peaks lower, 21,136 MiB, yet projects to 24.5 hours, and `--parallel 4` is the only setting inside both limits. The measurement logs are in `reference/deploy/` of the reproduction package.

Build cost. Rebuilding the submitted weights costs 27.6 GPU-hours on 8×H100, 3.5 hours of wall-clock time, plus about 15 CPU-minutes of data preparation. Steps 601 to 1,400 of the GRPO run were a checkpoint sweep and are not counted here; §5 shows that curve to be flat on held-out data, so the sweep was exploration rather than a build step. No external commercial API was used for training or inference, at a cost of zero.

Weights (Q5_K_M + imatrix)	19.23 GB
Vision encoder (F16)	0.93 GB
Peak VRAM	23,104 / 24,576 MiB
Generated tokens	4,489,762
Votes consumed	8 → 3.2 average
Training compute	27.6 GPU-h
SFT / GRPO	8.7 / 18.9 GPU-h
Inference compute	19.4 GPU-h (H100)
External API	none, zero cost

Table 2. Resource and cost summary. Compute figures are H100 hours.

Stage	All	dup	nondup
Zero-shot non-thinking	43.8	61.7	29.2
Zero-shot thinking, 1 vote	46.8	70.2	27.8
Supervised finetuning	62.1	90.2	39.2
thinking, 1 vote (untrained)	*57.9*	*87.1*	*34.1*
+ GRPO	62.6	90.2	40.1
+ ch. 2 (thr. 6)	63.2	90.6	40.8
+ ch. 3 (cue0, thr. 5)	**63.8**	90.6	**41.9**

Table 3. Stage-by-stage DEV1000 EM. dup covers 449 items, nondup 551.

5. Experiments

Main results. Table 3 decomposes the accuracy by stage. Training carries the model from 43.8 to 62.6, and the test-time cascade adds 1.2 more on top of frozen weights. That 1.2 is diluted by the 449 dup items already at 90; on nondup, the segment where training had also been the only thing moving, the cascade adds 1.8.

The final entry scored **0.94066** on the public leaderboard (539/573) and **0.93495** on private (230/246), placing fifth.

Contrast against an ensemble. The rules forbid combining the outputs of several models but explicitly permit test-time augmentation within the inference time limit. A three-model ensemble, which we measured but never submitted, scores 64.7 on DEV1000 against the cascade's 63.8: **rule compliance costs 0.9 EM**, and the compliant route needs one weight set instead of three.

Validating the choices on a held-out split. The threshold $t = 6$ was selected on DEV1000. Re-sweeping it on VAL500, which no checkpoint or threshold choice had ever seen, puts the argmax at $t = 6$ again, with a net gain of 3 items in 500 and 27 items flipped.

The same re-measurement shows the GRPO checkpoint curve to be flat on VAL500: every checkpoint from step 100 to 1,400 lands between 62.2 and 62.8, and step 600 is not the peak there. Checkpoint selection was a local property of DEV1000, consistent with §3.1: the gain from further training was not separable from noise.

Generalization. The public split returns a score on hidden answers twice a day, which makes it a selection surface: repeated readings on 573 items will eventually favour a candidate that suits them for reasons invisible to us. Training, checkpoint, and threshold decisions were made on the internal DEV1000, which gated every submission as well. Choosing which of the scored candidates to enter as final, however, is a decision the public score does enter, and we did not let it decide alone: the final entry had to hold up on VAL500 as well. Because VAL500 took no part in the sweeps that produced the candidates (§2.1), it reads that blind spot instead of repeating it. We also reported every number decomposed into dup and nondup rather than chasing the saturated segment.

The private split bears this out. Our score fell by **0.57 pp** between the two splits, **the second smallest drop** among the fifteen highest-placed teams on the public leaderboard, against a mean drop of 1.68 pp.

6. Limitations and Conclusion

Limitations. The vote-supplying channel was never trained, by design (§3.1), and the ceiling that leaves is visible: the correct answer sits somewhere in the eight votes for 75.7% of items while the cascade delivers 63.8, a gap of 11.9 pp. Of the residual errors, 43% have no correct answer in any vote, so most of what remains is beyond the reach of any aggregation rule. A directly trained thinking channel is the most plausible way to close it. Lacking one, the threshold sits at 6 of 8, since a channel trailing non-thinking by 4.7 pp cannot carry the answer alone; a lower threshold recovers more buried answers but breaks more that were already right.

Nearly half of DEV1000 can be answered by recalling an order seen in training. On nondup alone, the segment that measures generalization, the same cascade scores 41.9 rather than 63.8. The overall average is not a generalization estimate, and the leaderboard score carries the same caveat. The third channel's separate contribution to the test score is also not isolated, since it ships inside the submitted entry.

Conclusion. Once task adaptation had taken this model to 62.6 EM, further training returned noise while the same frozen weights returned 1.2 EM more at inference. The gain came not from voting, which by itself loses accuracy here, but from gating the vote behind a majority threshold, and the provable equivalence of early stopping to full voting is what made that gate affordable on one 24 GB GPU.

References

[1] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *EMNLP*, pages 12375–12396, 2023.

[2] Georgi Gerganov and llama.cpp contributors. llama.cpp. https://github.com/ggml-org/llama.cpp, 2026.

[3] 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 *ICLR*, 2022.

[4] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[5] Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *ICLR*, 2024.

[6] Qwen Team. Qwen3.5-27b. https://huggingface.co/Qwen/Qwen3.5-27B, 2026.

[7] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[8] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

[9] Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A scalable lightweight infrastructure for fine-tuning. In *AAAI*, 2025.