SNU AI Challenge 2026 Report

Team Deepest

Sunghee Ahn **Donghyun Kim** **Younggil Do** **Seungyeop Yi**

{snusnowhite, joshua2031, gilyoung7, tmootu76}@snu.ac.kr

Seoul National University

1 Introduction

This report presents our approach to the SNU AI Challenge 2026, covering the entire pipeline from dataset analysis and model selection to training and inference. Our method achieved **2nd place** on both the public and private test sets.

2 Dataset

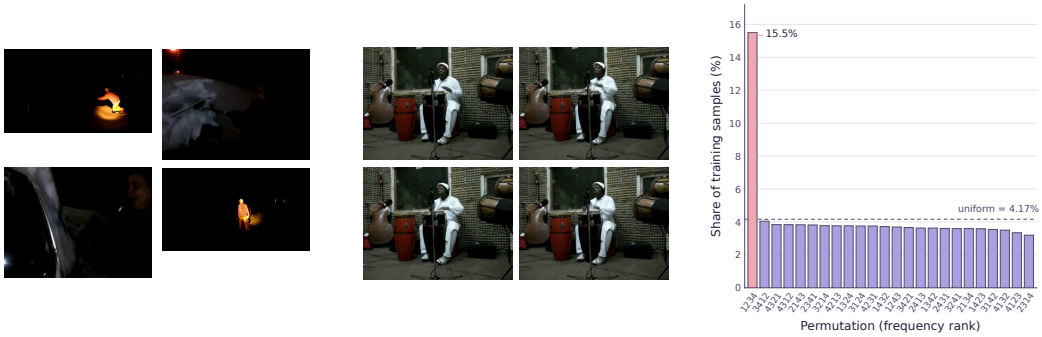

Figure 1: Training-set analysis: *(left)* a sample with nearly dark frames, *(middle)* a sample with near-duplicate frames, and *(right)* the distribution of target permutations.

Analysis. We begin by examining the training set through manual inspection and label statistics. During manual inspection, we find that a non-negligible number of samples contain nearly dark or visually similar frames, making temporal ordering trivial. Figure 1 shows examples of such samples. This observation leads us to introduce a filtering process.

In our statistical analysis of the labels, we examine the distribution of target permutations. Ideally, labels should be distributed uniformly across all $4! = 24$ possible permutations. However, as shown in the right panel of Figure 1, the distribution is substantially skewed. This motivates us to introduce target-label shuffling.

Two-Stage Filtering. Our filtering process consists of two stages. The first stage filters out training samples containing *uninformative* frames, such as nearly black, blank, uniform, or extremely blurry images. Each raw frame is converted to grayscale, after which the standard deviation of its pixel values and the mean absolute image gradient are computed. A frame is classified as *uninformative* by thresholding these two values, and any training sample containing two or more such frames is removed.

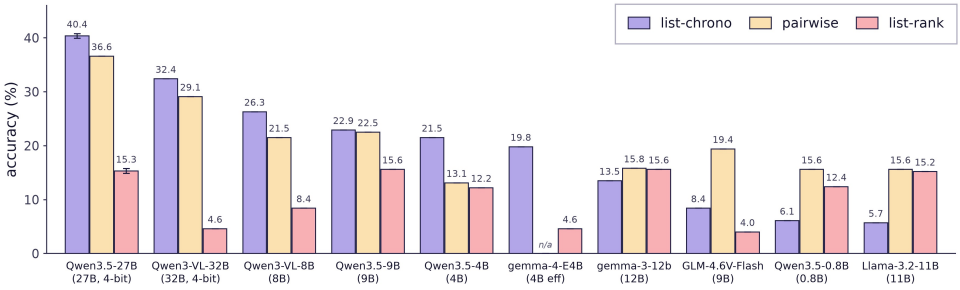

Figure 2: Performance comparison across base models and prompting strategies.

The second stage filters out samples containing *near-duplicate* frames. Each frame is converted to grayscale and resized to 8×8, from which a 64-bit average hash is computed. A frame pair is classified as *near-duplicate* if the Hamming distance between their hashes is four or less. A sample is removed if all six possible frame pairs are classified as near-duplicates.

Dynamic Shuffling. To mitigate potential training bias caused by the dataset's skewed label distribution, the four frames in each sample are randomly shuffled at every training epoch, and the corresponding target labels are updated accordingly. This procedure discourages the model from exploiting positional shortcuts and encourages it to rely on visual and textual evidence.

Final Dataset. After two-stage filtering, 9,258 of the original 9,535 samples remained. We removed 231 samples containing *uninformative* frames and 46 containing *near-duplicate* frames. All development experiments—base-model selection, prompt and inference-strategy comparisons, and the ablation studies—use the unfiltered dataset with a stratified 95/5 split: 475 held-out validation samples and 9,060 training samples.

3 Model and Training Setup

Base Model Selection. Since the task requires both visual and temporal understanding, we adopt a vision-language model (VLM). To select a base model, we evaluate several candidate models using different prompts. Figure 2 summarizes the results. Among the models that fit within our 24 GB VRAM budget, the **4-bit-quantized Qwen3.5-27B** (`Qwen3.5-27B-4-bit`) achieves the best performance with the *list-chrono* prompt. We therefore select it as our base model.

Prompt Design. We evaluate three prompt strategies with different output formats: *pairwise*, *list-rank*, and *list-chrono*. The *pairwise* prompt predicts which frame occurs earlier for each of the six frame pairs. The *list-rank* prompt assigns a temporal rank to each frame in the given input order, whereas the *list-chrono* prompt directly lists the frame indices from earliest to latest. As shown in Figure 2, *list-chrono* performs best across different base models. We therefore adopt it for subsequent experiments. The complete prompts are provided in the submission code.

SFT without Thinking. To determine whether long chain-of-thought (CoT) reasoning improves performance on the challenge task, we compare thinking-enabled and thinking-disabled models on the validation set. As shown in Figure 3, enabling thinking provides no significant improvement. We therefore rule out approaches requiring long reasoning, including SFT with reasoning trajectories generated by a stronger model and reinforcement learning. Instead, we directly fine-tune the model to output only the ordered list, with the additional benefit of reducing the cost of training-data preparation, model training, and inference.

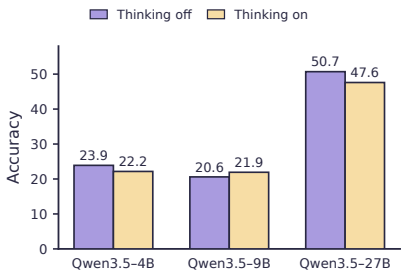

Figure 3: Thinking-off/on accuracy. Budgets: 512 tokens (4B/9B) and 1,536 tokens (27B).

Training Details. We fine-tune the 4-bit-quantized Qwen3.5-27B with QLoRA, applying LoRA adapters (r=16, α=32, dropout 0.05) to the language stack while freezing the vision tower. This yields 79.7M trainable parameters (0.53%).

We train with answer-only loss for two epochs using paged 8-bit AdamW, a learning rate of 10^{-4}, cosine decay, and an effective batch size of 16. We select the epoch count based on an 8B pilot run, in which validation loss begins to increase after epoch 2.06. For the final run, we merge the validation split back into the training set and train on all 9,258 filtered samples.

Training takes 8.3 hours on a single A100 80GB GPU. Optimized attention kernels reduce the step time from 51.6 seconds to 25.8 seconds.

4 Inference-Time Strategies

With only 24 possible orderings, one checkpoint supports several decoding schemes at no training cost. The model outputs four digits: the first names the earliest frame, the second the next, and so on. At each of the four decoding steps we allow only the tokens `1`–`4`, so every step yields a probability over the four frames. Choosing the most likely digit at each step separately can repeat a frame, for example `1123`, which is not a valid ordering. We therefore score all 24 valid orderings by multiplying their four step probabilities and output the best one. This costs one forward pass over the prompt plus four decode steps. On top of this decoder we pursued two strategies.

Presentation-order TTA. The model is sensitive to where each frame sits in the prompt, so we present every sample under eight position-balanced arrangements (four rotations and their reverses), which we call views, map each view's output back to the original frame indices, and aggregate the eight predictions. We evaluated two probability-based aggregation rules: averaging the per-position probabilities before permutation scoring, and averaging the normalized 24-permutation distributions after scoring each view. We refer to the latter as *posterior aggregation*. The two aggregation rules were evaluated on the held-out validation set. As shown in Table 1, posterior aggregation achieved 59.54% exact match, compared with 59.12% for position-probability averaging (+0.42%p).

Table 1: TTA aggregation ablation.

Aggregation rule	Validation exact-match (%)	Δ (%p)
Position-probability average	59.12	0.00
24-permutation posterior average	**59.54**	**+0.42**

For clarity, "posterior" here denotes a normalized score over the 24 candidate permutations. For view v and candidate permutation σ, we first compute the log-score

$$s_v(\sigma) = \sum_{t=1}^{4} \log p_{v,t}(\sigma_t), \tag{1}$$

where $p_{v,t}(i)$ is the constrained-decoding probability of image i at chronological position t. We convert these scores into a per-view distribution using a temperature T:

$$q_v(\sigma) = \frac{\exp(s_v(\sigma)/T)}{\sum_{\sigma' \in \mathcal{S}_4} \exp(s_v(\sigma')/T)}. \tag{2}$$

The final TTA distribution is the arithmetic mean across the eight views,

$$\bar{q}(\sigma) = \frac{1}{8} \sum_{v=1}^{8} q_v(\sigma), \qquad \hat{\sigma} = \arg\max_{\sigma \in \mathcal{S}_4} \bar{q}(\sigma), \tag{3}$$

and $\hat{\sigma}$ is converted to the required output by deterministic post-processing.

Self-refining sequence scoring (SRSS). Self-refining sequence scoring is a test-time heuristic introduced in this work. Let $x = (I_1, I_2, I_3, I_4)$ denote the current presentation of the frames. At each

round, the constrained decoder provides a probability $p_t(i \mid x)$ for the image index at chronological position t. For each of the 24 permutations $\sigma \in \mathcal{S}_4$, we compute its log-score

$$S(\sigma \mid x) = \sum_{t=1}^{4} \log p_t(\sigma_t \mid x), \qquad \hat{\sigma} = \arg\max_{\sigma \in \mathcal{S}_4} S(\sigma \mid x). \tag{4}$$

We then physically reorder the frames according to $\hat{\sigma}$ and run the same scoring procedure again. If $R_{\hat{\sigma}}(x)$ denotes this reorder operation, the next presentation is

$$x_{k+1} = R_{\hat{\sigma}_k}(x_k). \tag{5}$$

The iteration stops when the model returns the identity permutation, meaning that the presented frames are already in chronological order. It also stops when a previous presentation is revisited (a cycle) or when a fixed round limit is reached; in either case, we return the first-round prediction. Thus SRSS repeatedly checks whether the model's own proposed order is self-consistent, while preserving a deterministic fallback to the ordinary single-pass decoder. The procedure converged in 1.91 rounds on average in our internal evaluation.

Table 2: Comparison of inference strategies. Teacher-forced 24-candidate scoring is the reference.

Strategy	val@475 (%)	Δ (%p)
Teacher-forced 24-candidate scoring	60.6	0.0
B (K=8 posterior TTA)	**62.7**	**+2.1**
SRSS	62.3	+1.7

Results. On val@475, K=8 posterior TTA improved exact match from 60.6% to 62.7% (+2.1%p), while SRSS reached 62.3% (+1.7%p). Thus both test-time strategies improved over the same teacher-forced reference, with TTA showing a 0.4%p advantage over SRSS on this validation split. Based on this comparison, we selected the $K = 8$ posterior TTA strategy for the final inference pipeline.

5 Training Analysis

To quantify the contribution of each component of the final training pipeline, we compare the effects of the training hyperparameters, the adaptation method, and the preprocessing. All comparisons keep the backbone and output format fixed whenever possible and change one factor at a time.

Preprocessing ablation. Sec. 2 introduced a two-stage filter that removes the samples containing multiple uninformative or near-duplicate frames. To measure its contribution, we compared two models trained under identical settings, one with the filter applied to the training data and one without. As shown in Table 3, filtering improves accuracy by 0.5 %p (61.1% versus 60.6%).

Table 3: Preprocessing ablation.

Setting	acc (%)	Δ (%p)
no filter	60.6	0.0
filter*	**61.1**	+0.5

Hyperparameter tuning. To confirm that the chosen training configuration is optimal, we varied each hyperparameter in turn (Figure 4); every number is the mean of three inference runs. On Qwen3.5-27B, performance peaked at two epochs with a learning rate of 1×10^{-4}, and dropped again at three epochs, suggesting that longer training leads to overfitting. The adaptation method mattered most: QLoRA reached 60.6%, whereas full fine-tuning collapsed to 4.8%. This shows that directly updating all weights on a small dataset can destroy the pretrained representations, and it is the basis for our final choice of QLoRA.

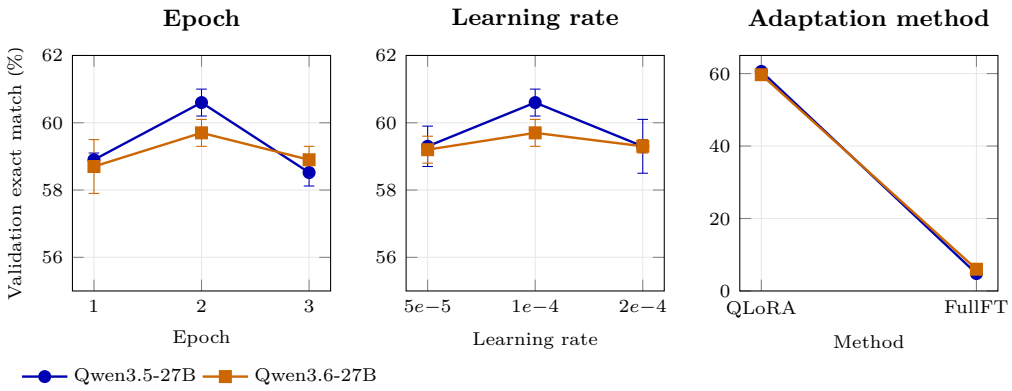

Figure 4: Training-recipe selection experiments: The base VLM, prompt, and output format are held fixed while only the epoch count, learning rate, and adaptation method are varied. The first two panels truncate the y-axis to 55–62% to magnify the performance differences, while the adaptation panel uses the 0–65% range to also show the Full-FT collapse. Full-FT keeps the vision tower frozen. Error bars denote the standard deviation over three inference runs with identical settings.

6 Limitations

A fixed TTA budget spends the same compute on easy and hard samples. Presentation-order TTA runs all eight views on *every* test sample, even though on 87% of the evaluation set the views agree unanimously and the seven extra passes only re-confirm the first. An adaptive schedule—decoding views one at a time and stopping early once the model is confident enough—would spend the budget where it matters. We did not pursue this, since eight views already fit comfortably inside the 24-hour limit, but it is the first thing we would relax under a tighter compute constraint.

The validation ranking of the two decoding strategies did not transfer to the private split. We selected eight-view TTA because it was stronger on validation, and it also led on the public split (545 versus 540); on the private split, however, SRSS was better (234 versus 232). The margin we measured was simply smaller than the variation induced by the shift between splits. Our validation protocol has little resolution for this comparison: the two decoders share a checkpoint and agree on all but a few percent of the 475 validation samples. More fundamentally, they differ only where the model is already unsure—all 26 disagreements fall inside the 13% of samples whose eight views are not unanimous. Under distribution shift that ambiguous region changes size, and with it the relative standing of the two decoders; there is no reason for averaging over presentation orders to be uniformly more robust than feeding the model's own hypothesis back as evidence, and on the private split it was not.

The two decoders were treated as alternatives rather than combined. Using both is not an ensemble in the usual sense—they decode from the same fine-tuned checkpoint and differ only in how the prompt is arranged across passes—so a combined decoder is admissible and, given the above, likely more robust than either alone: invoking SRSS only on the non-unanimous 13%, or reconciling the two posteriors where they disagree instead of discarding one of them by fiat.