TEMPO: Temporal Event Matching via Permutation-Scored Ordering with Kendall-Tau-Stratified Hard-Negative Listwise Ranking

Gyuyeon Lim [1]

Abstract

I present TEMPO, my submission to the SNU AI Challenge 2026 task of reordering four shuffled video frames to match a caption. Rather than generating the ordering directly, I reformulate it as verification: a vision-language model scores all $4! = 24$ candidate orderings by $\log P(\text{Yes}) - \log P(\text{No})$, returning the best-scoring one. Training uses Kendall-tau-stratified hard-negative sampling and a listwise softmax loss corresponding to the top-1 probability under a Plackett–Luce model. My claim: combining Kendall-tau-stratified sampling with a joint, not pairwise, softmax loss focuses the gradient on the hardest near-misses, removing the need for hand-tuned distance weights – isolated through controlled, mostly single-variable changes across 21 tracked versions. The same ranking formulation was retained when scaling the backbone from 8B to 32B under QLoRA, which produced my largest raw gain, raising the public score from 0.90052 to 0.91099 (private: 0.90650). I report the ablations, error analyses, dead ends, and hardware-constrained re-engineering needed to reproduce this checkpoint on a single 24GB GPU. Code and the full version history are available at https://github.com/lky473736/snu-ai-challenge-2026.

1. Introduction

Given four frames sampled from a short video, shuffled into a random order, and a single caption describing the underlying event, the task is to recover the exact permutation of the four frames that matches the true temporal order (Kaggle competition snuaichallenge). Out of $4! = 24$ possibilities, the model is scored purely by exact-match accuracy: a prediction counts only if all four positions are simultaneously correct.

Objective. Rather than maximizing the leaderboard score by blind trial and error, my objective was to isolate which modeling choices actually drive performance in fine-grained temporal reasoning. Kaggle-style competitions often reward bundling several changes into one submission; I made the opposite bet. Changing at most one variable per version traded iteration speed for diagnostic clarity, so I could trust negative results as much as positive ones.

Origin of the Approach. My first baseline (v1: Qwen2-VL-7B, LoRA, adjacent-swap negatives only) reached 0.79057. A held-out validation diagnostic immediately revealed a pattern: remaining errors were overwhelmingly one adjacent swap from the ground truth, not randomly scrambled. This shaped everything that followed – the task is not a uniform 24-way classification problem with a flat difficulty surface, but one with a structured difficulty landscape, and a procedure spending equal effort on a one-swap near-miss and a six-swap reversal is solving the wrong problem.

Core Contributions. Two design choices follow from this observation, and I argue it is their specific combination – not either piece alone – that constitutes the final system's training formulation: (1) Kendall-tau-stratified hard-negative sampling that oversamples near-miss negatives rather than sampling uniformly (Section 3.2), paired with (2) a listwise, not pairwise, loss that automatically allocates larger gradients to whichever negative is currently most confusing (Section 3.3). Earlier pipeline versions achieved a similar effect only by hand-tuning per-distance loss weights (an AdaptiveDistanceLoss); Section 3.3 shows this becomes unnecessary once the loss is a joint softmax over the full candidate group, since the emphasis on hard negatives then emerges from the normalization itself. Section 4 also examines the effect of model capacity separately from the ranking formulation: holding the training recipe fixed while scaling the backbone from

[1]SNU AI Challenge 2026. Correspondence to: Gyuyeon Lim <lky473736@gmail.com>.

SNU AI Challenge 2026 — Gyuyeon Lim. Methodology report submitted for code and reproducibility review.

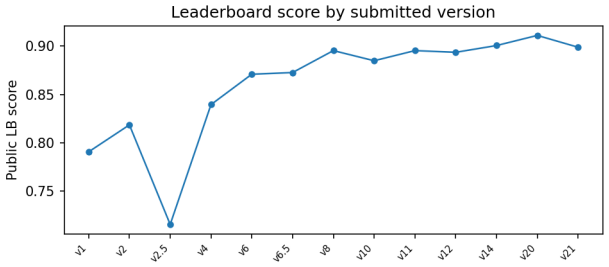

Figure 1. Public leaderboard score by submission, in chronological order. The sharp early drop was caused by a test-time horizontal-flip augmentation that destroyed directional information; the final point is my submitted model.

8B to 32B parameters yielded my single largest raw gain.

2. Iteration History: 21 Versions

Before detailing the final system, Table 1 and Figure 1 outline my complete iteration history. Recording failed attempts is crucial, as they are often as informative as the successes. My first two versions conflated multiple changes at once, making it impossible to confidently attribute the resulting score shifts to a specific cause; from v4 onward, each version changed at most one variable relative to its predecessor. TPRU-7B (Gao et al., 2026), swapped in briefly at v2, is a Qwen2.5-VL-7B variant trained with reinforcement fine-tuning for temporal and procedural understanding, released by the paper's authors on Hugging Face.

The remainder of this report focuses on my final submitted model: its architecture and training/inference pipeline (Section 3), the findings behind each design decision (Section 4), its strengths and limitations (Section 5), and the engineering needed to reproduce this checkpoint on constrained hardware (Section 6).

3. Method: The TEMPO System

3.1. Architecture and Verification Formulation

Let $x = (f_1, \ldots, f_4)$ be the shuffled frames and c the caption. The straightforward approach is an autoregressive model outputting the ground-truth permutation $\sigma^* \in S_4$ directly (e.g., "2, 4, 1, 3"). I avoid this because it gives the model no way to explicitly compare all 24 candidate permutations before committing to one; more incidentally, autoregressive generation also introduces unnecessary sequential decoding and can propagate an early wrong token through the rest of the sequence (exposure bias).

Instead, TEMPO frames the task as a verifier. Given

Table 1. Selected version history highlighting major turning points; intermediate versions are omitted for brevity. LB = public leaderboard exact-match accuracy. † = final submission.

Ver.	Change (vs. previous)	LB
v1	Qwen2-VL-7B, adjacent-swap negatives	0.791
v2	Backbone → TPRU-7B, ListNet loss	0.818
v4	Distance-covering hard negatives	0.839
v6	Backbone → Qwen3-VL-8B	0.871
v6.5	Live hard-negative resampling	0.873
v8	LoRA r=128, LR 5×10^{-5}	0.895
v9	DoRA / PiSSA+rsLoRA (LoRA variants)	discarded
v10	Noun-count auxiliary loss	0.885
v11	LLM frozen, vision-only LoRA	0.895
v12	Joint LLM+vision LoRA	0.894
v14	Listwise softmax loss (Equation (2))	0.901
v18	Pairwise Bradley–Terry (abandoned)	–
v20†	Backbone → 32B + QLoRA	0.911
v21	LoRA rank 128 → 256	0.899

ordering σ, the model receives a chat prompt with caption c and frames reordered as $x_\sigma = (f_{\sigma(1)}, \ldots, f_{\sigma(4)})$, instructed to answer only "Yes" or "No". No text is generated: I run a single teacher-forced forward pass and extract the "Yes"/"No" logits at the final position, defining a scalar score:

$$s_\theta(\sigma) = \log P_\theta(\text{Yes} \mid x_\sigma, c) - \log P_\theta(\text{No} \mid x_\sigma, c). \quad (1)$$

$s_\theta(\sigma)$ is a signed log-odds: positive when "Yes" is more likely, with magnitude indicating confidence. Since $|S_4|$=24, no decoding strategy is needed – I score all 24 candidates and predict $\hat{\sigma} = \arg\max_\sigma s_\theta(\sigma)$. This scales poorly beyond $n = 4$, but at $n = 4$ it eliminates the approximation error of autoregressive beam search. Because Equation (1) relies only on final-token logits, the predicted ranking is effectively invariant to batching and device placement under numerically equivalent computation – a property useful for hardware-constrained reproduction (Section 6).

The backbone is Qwen3-VL-32B-Instruct (Bai et al., 2025), fine-tuned with LoRA (Hu et al., 2022) (r=128, α=256, dropout 0.05) on the

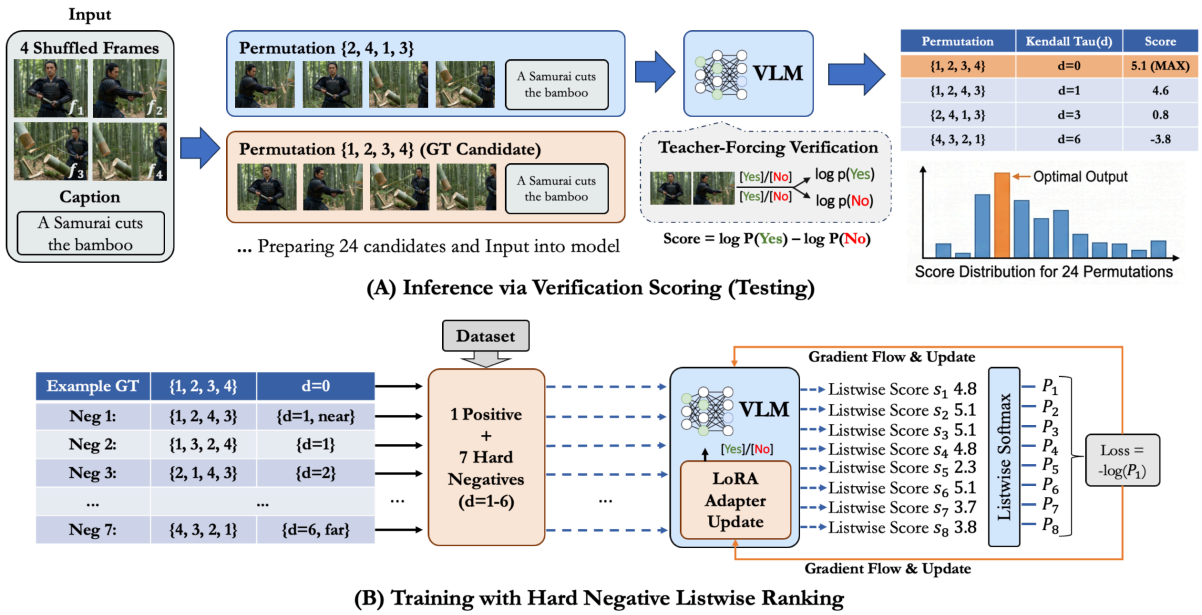

Figure 2. TEMPO overview. (A) Inference: All 24 permutations are evaluated via a single Yes/No verification forward pass each (Equation (1)), and the permutation with the maximum score is returned. (B) Training: One positive sample and seven Kendall-tau-stratified hard negatives are scored jointly and optimized using the listwise softmax loss (Equation (2)), backpropagating only into LoRA adapters attached to a frozen, quantized backbone.

query/key/value/output/gate/up/down projections of every decoder layer (∼1.07B of 34.4B parameters, 3.12%). Following QLoRA (Dettmers et al., 2023), the decoder is loaded in 4-bit NF4 with double quantization. One detail matters: the vision tower and output head stay in bfloat16, never quantized – in my own experiments, quantizing the vision path measurably reduced accuracy.

3.2. Training: Hard-Negative Sampling by Kendall-Tau Distance

For $\sigma, \sigma^* \in S_4$, let the Kendall-tau distance $d = \tau(\sigma, \sigma^*)$ be the minimum number of adjacent transpositions converting σ into σ^*. For $n = 4$, d ranges from 0 (ground truth) to 6 (complete reversal).

This gives a precise way to act on Section 1's observation: uniformly sampling negatives from the 23 incorrect permutations is inefficient, since most random negatives are trivial (large d), while the remaining errors are disproportionately concentrated among small-d near-misses – which Section 4 shows correlates with these near-misses being visually more similar to the ground truth.

For every positive, I sample a fixed multiset of K=7 negatives stratified by distance with counts $\{d$=1:2, d=2:1, d=3:1, d=4:1, d=5:1, d=6:1$\}$, so every group of eight covers the full difficulty spectrum while anchoring the gradient on d=1 near-misses. In my 4×H100 training setup, K=7 was the largest negative set I could fit without OOM; K=8 failed regardless of batch-size adjustments.

3.3. Training: Listwise Softmax Ranking Loss

The 1+K=8 scored candidates $\{s_1, \ldots, s_8\}$ (index 1 is the positive) are treated as one draw from a Plackett–Luce ranking model (Plackett, 1975), where item k ranks first with probability $\exp(s_k)/\sum_j \exp(s_j)$. Training minimizes the negative log-likelihood of ranking the positive first:

$$\mathcal{L} = -\log \mathrm{softmax}(s)_1, \quad \frac{\partial \mathcal{L}}{\partial s_j} = \mathrm{softmax}(s)_j - \mathbf{1}[j{=}1]. \tag{2}$$

The gradient in Equation (2) shows that each negative $j \neq 1$ receives a penalty proportional to $\mathrm{softmax}(s)_j$: this shrinks toward zero once the model scores it well below the positive, and stays largest for whichever negative currently competes closest – empirically, almost always a d=1 near-miss (Figure 3).

This joint normalization reproduces, as an emergent property, what I previously had to engineer by hand via distance-dependent weighting (used through v13). Switching to this listwise loss eliminated that manual tuning while improving the leaderboard score.

Optimization uses AdamW (weight decay 0.01), cosine

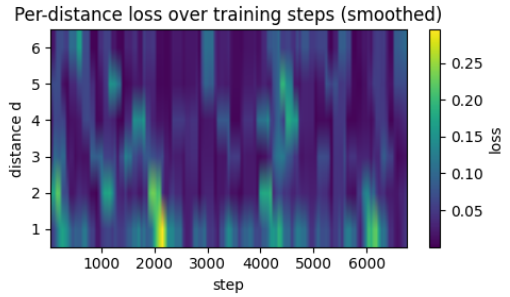

Figure 3. Per-distance loss L_d for the predecessor run that introduced this loss, decomposed by Kendall-tau distance: x-axis is training step, y-axis is $d \in \{1, \ldots, 6\}$ (bottom to top), color intensity is loss magnitude (brighter = harder). The persistent brightness of the bottom row (d=1) shows near-misses stay hardest throughout training, echoing Figure 4.

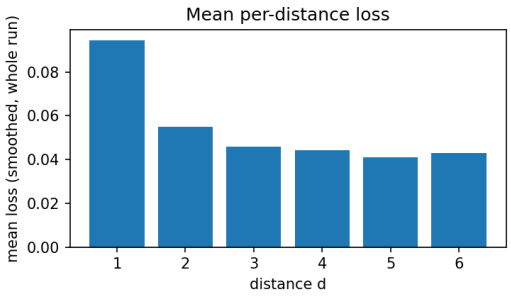

Figure 4. Mean per-distance loss L_d averaged over the predecessor training run in Figure 3. The monotonic decrease from d=1 to d=6 mirrors the visual-similarity finding: distance in the model's loss and distance in raw pixel space exhibit a consistent monotonic relationship.

schedule, peak LR 5×10^{-5}, 0.05 warmup, and gradient clipping at 1.0, on 4×H100 80GB GPUs via accelerate DDP. The dataset is 9,535 caption-annotated frame groups; 5% (476 groups, fixed seed 42) is held out for validation, ensuring the test set was never directly used for model selection or hyperparameter tuning. A dynamic chunk-size counter halves itself on OOM and persists the reduced size, without affecting output logits, since candidate verification is independent under causal attention masking.

4. Results and Error Analysis

My final model scores public 0.91099 / private 0.90650, the best of 21 versions in Table 1, obtained by holding the v14 recipe above fixed and scaling the backbone from 8B to 32B under QLoRA (necessitating the switch to 4-bit quantization to fit the larger model in memory). This backbone change produced the largest single raw gain of the whole project, larger than any change to the loss or sampling scheme. The ranking formulation was kept fixed across this scaling experiment, allowing the effect of model capacity to be examined separately from changes to the training objective. Figure 3 shows the training dynamics behind the shared recipe.

Error analysis suggests the d=1 bottleneck is partly visual, not merely statistical. The difficulty is concentrated near the decision boundary, and temporally adjacent frames are intrinsically more visually similar than non-adjacent ones, not simply an artifact of under-training. A pixel-RMSE and histogram-intersection analysis of 1,500 training samples supports this: adjacent frames are measurably more similar than non-adjacent ones (Cohen's d=0.349, consistent across resolutions from 32 to 448 pixels; Figure 4). This did not translate into a fix: two dedicated d=1 tie-breaking losses (pairwise Bradley–Terry; difference-image tie-break) both underperformed the listwise loss and were discarded – a dead end worth recording, since correctly diagnosing an error source did not yield an effective training signal.

Caption complexity and error patterns. Beyond permutation-level difficulty, I also checked which caption properties correlate with errors. Regressing correctness against 14 features, sentence length looked predictive (r=0.435) but was highly correlated with noun count (r=0.892); controlling for both, only noun count remained significant (p=0.0002 vs. p>0.08) – concrete, groundable objects, not verbosity, drive difficulty. Custom noun-weighted auxiliary losses still failed to improve on the baseline.

Validation accuracy does not always predict leaderboard rank. v10 and v12 both beat their predecessors on held-out validation yet scored lower publicly. This may reflect differences between the validation distribution and the leaderboard's 70% test subset, though I did not investigate further to avoid risking the data-leakage rule. I treat leaderboard score, not validation accuracy, as authoritative in Table 1.

Vision-encoder fine-tuning did not help, under two isolation strategies: frozen LLM + vision-only LoRA (v11), and joint LLM+vision LoRA (v12); both underperformed the LLM-only baseline (v8). Two independent negative results let me stop investigating this with confidence.

Rank scaling beyond r=128 is inconclusive, not negative. v21 (r=256) reached my best-ever validation accuracy but a lower public score than v20, tying it privately – likely overfitting from a learning rate never

re-tuned for the larger rank, an open question rather than a ceiling on rank scaling.

5. Advantages and Limitations

Advantages. Verification (Equation (1)) turns a 24-way ordering problem into 24 independently scored binary verification problems compared by a single argmax, avoiding the exposure bias of autoregressive generation. The listwise loss (Equation (2)) needs no hand-tuned distance weight to focus on hard negatives – an emergent property of joint normalization – and the single-variable discipline behind Table 1 means every gain, and every failure, is attributable to one isolated change.

Limitations. Exhaustive scoring is tractable only because n=4; it does not scale to longer sequences without a different search strategy. My error analysis is correlational: I identified what correlates with failure (visual similarity at d=1, noun count), but turning each correlation into a training signal did not improve results, so a genuine fix for the d=1 bottleneck remains open.

6. Reproducibility on Constrained Hardware

The competition requires the submitted checkpoint be reproducible on a single RTX 3090 (24GB), hardware I never used during development (4×H100 80GB throughout). Loading the 32B backbone alone consumed 22–23GB of the 23.56GB usable, leaving no room for the ∼2GB LoRA adapter. Catching OOM exceptions, shrinking chunk size, and raising max_memory limits did not help, since the problem was static weight residency, not activation memory.

The fix exploits an asymmetry already in the architecture (Section 3.1): the vision tower and output head were never quantized, so relocating them to CPU changes only where a bfloat16 tensor sits, not the arithmetic on it. Two further fixes were needed: re-registering accelerate's device-migration hooks, which do not survive LoRA injection into an already-dispatched model, and restricting the LM head to the final token (logits_to_keep=1) with the KV cache disabled (use_cache=False), since Equation (1) only ever reads that one position. Neither changes any computed value. The 819-sample test set now completes in ≈5h20m on a single RTX 3090 (23–25s/sample), comfortably within the 24-hour limit.

7. Conclusion

Reformulating temporal ordering as exhaustive verification, with Kendall-tau-stratified hard-negative sampling and a listwise softmax loss that emergently emphasizes hard negatives without any hand-tuned distance weight, produced my best result across 21 versions (public 0.91099, private 0.90650) via a single backbone-scaling change, which in turn required switching to QLoRA to fit the larger model in memory. The negative results – DoRA and PiSSA/rsLoRA (Liu et al., 2024), vision fine-tuning, and two auxiliary-loss attempts – matter as much as the positive result, since the single-variable discipline behind most of this history is what makes them trustworthy rather than anecdotal, and each one narrows down what to try next. Concretely, two directions follow directly from this report: since correlational fixes for the d=1 bottleneck (tie-breaking losses, noun-weighted auxiliary losses) did not work, pretraining the vision encoder directly on adjacent-frame discrimination before LoRA fine-tuning is a natural next attempt; and extending exhaustive verification beyond n=4 will need a pruning or beam-style approximation to keep the $n!$ search tractable at larger scale.

References

Bai, S., Cai, Y., Chen, R., et al. Qwen3-VL technical report. arXiv preprint arXiv:2511.21631, 2025.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. QLoRA: Efficient finetuning of quantized LLMs. In Advances in Neural Information Processing Systems, 2023.

Gao, Z., Wang, X., Tan, X., and Xie, Y. TPRU: Advancing temporal and procedural understanding in large multimodal models. arXiv preprint arXiv:2602.18884, 2026.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. DoRA: Weight-decomposed low-rank adaptation. In International Conference on Machine Learning, 2024.

Plackett, R. L. The analysis of permutations. Journal of the Royal Statistical Society: Series C (Applied Statistics), 24(2):193–202, 1975.