SNU AI Challenge 2026 · 텍스트로 풀어보는 장면의 재구성
최종 기술 보고서

GeekSeek 팀

1. 서론

본 과제는 한 영상에서 뽑아 무작위로 뒤섞은 네 장의 프레임을, 함께 주어진 한 문장의 캡션을 단서로 원래 시간 순서대로 복원하는 문제다(예: 섞인 네 장 → '[3, 1, 4, 2]'). 채점은 완전일치 정확도(Exact Match, 네 순서가 정답과 완벽히 같을 때만 정답이며 부분 점수 없음)로 이루어지므로, 한 곳만 어긋나도 그 표본은 0 점이다. 이 평가 방식에서는 모델을 키우는 것만으로 곧 한계에 부딪혔다. 정답 순서를 문장으로 '생성'하는 표준 방식도 순서 판단과 출력 형식을 동시에 처리해야 하므로, 형식 오류가 곧 실점으로 이어졌다.

그래서 우리는 문제를 다르게 바라보았다. 정답 순서를 문장으로 써 내려가게 하는 대신, 프레임을 둘씩 비교해 어느 쪽이 먼저인지 판별하고 그 결과를 모아 하나의 순서로 디코딩하도록 문제를 다시 정의했다. 시각-언어 모델 Qwen3.5-27B 하나를 RTX 3090(24GB) 한 장에서 운용했으며, 문제 구조·추론 대칭성·실제 오류 분포를 설계에 반영해 팀 최고 공식 성적(public 0.93891)을 얻었다. 이어지는 절에서는 데이터 활용(2 절), 모델 설계와 학습(3 절), 추론(4 절), 실험(5 절), 자원 효율과 비용(6 절)을 차례로 다룬다.

2. 문제 정의 및 데이터 분석

학습 데이터는 약 8,500 개 표본으로 구성되며, 각 표본은 캡션·네 프레임·정답 순서를 포함한다. 일부 정적 장면은 원래 제시된 순서가 정답이다. 정답을 단일 라벨로만 쓰지 않고, 여섯 쌍의 선후 관계와 시간 간격을 정답에서 규칙적으로 유도해 학습 신호로 함께 사용했다(외부·생성 데이터 미사용). 학습할 때는 프레임 제시 순서와 쌍 질문의 나열 순서·방향을 무작위로 바꿔, 모델이 우연한 배치가 아니라 실제 내용에 근거해 판단하도록 했다.

캡션 난이도	train	test
평균 단어 수	24.2	42.0
짧은 캡션 비율	29.4%	12.0%
시간 접속어	부분	대체로 포함

[표 1] train vs test(채점) 캡션 난이도. test 가 더 길고 쉬움.

캡션 유형	쌍-비교 정확도
시간 접속어 포함	6 / 8
다장면(long)	5 / 6
짧은 캡션	0 / 8
정적/identity	1 / 5

[표 2] 인간 블라인드 파일럿. 쌍 단위 조립 시 전체 14/30.

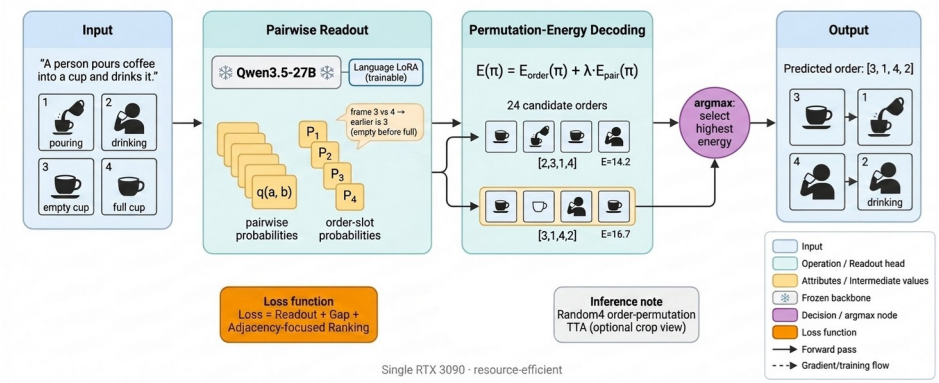
데이터 분포와 평가 점수의 관계도 확인했다. 리더보드 점수는 test 의 일부(public 70%, private 30%)로 산정된다. test 는 train 보다 캡션이 길고 짧은 캡션이 적어 대체로 더 쉽다([표 1]). 규정상 test 정답은 알 수 없으므로 test 에 맞춰 튜닝하지 않았다. 대신 train 을 train/val 로 나누고, val 을 no_ordering 여부와 캡션 길이에 따라 층화해 오프라인 검증 결과가 채점 분포와 크게 어긋나지 않도록 관리했다.

설계에 앞서 사람이 train 을 블라인드로 직접 풀어 본 결과, 네 장을 한꺼번에 배열할 때보다 두 프레임씩 선후를 비교한 뒤 조립할 때 더 안정적으로 맞혔다([표 2]). 이 결과는 3 절의 '쌍 단위 판별 후 조립' 구조를 뒷받침한다.

오류 분석 결과도 설계에 반영했다. 모델이 완전히 무작위로 틀리는 경우는 드물었고, 대부분 시간상 이웃한 두 프레임의 선후를 뒤바꾸는 형태였다. 완전일치 채점에서는 인접 오류 한 번만으로 표본 전체가 오답 처리된다. 따라서 3.2 절의 손실은 이 오류를 직접 줄이도록 설계했다.

3. 파이프라인 개요 (Pipeline Overview)

전체 파이프라인은 백본 확장과 구조화된 시간 추론으로 구성된다. 먼저 작은 시각-언어 모델 Qwen2.5-VL-7B 를 표준 생성 방식(정답 순서를 문장으로 출력)으로 LoRA 미세조정했지만, public 점수는 0.82024 에 그쳤다. 트랜스포머의 연산량을 늘리는 방법은 두 가지로 나눌 수 있다. 하나는 층을 늘려 토큰당 계산을 깊게 하는 깊이 추론(depth reasoning)이고, 다른 하나는 추론 시점에 토큰을 더 생성해 참조할 문맥을 넓히는 시간 추론(time reasoning; test-time compute)이다. 깊이 추론은 백본을 Qwen3.5-27B 로 확장해 확보했다. 더 큰 32B·38B 모델은 단일 RTX 3090 환경에서 뚜렷한 이득이 없었기 때문에 성능·메모리·QLoRA 안정성을 고려해 27B 를 선택했다. 시간 추론에는 최종 순서를 정하기 전에 여섯 쌍의 선후를 먼저 판독하는 구조적 스캐폴드를 사용했다. 추론 토큰을 늘려 test-time compute 를 확보한다는 점은 같지만, 자유 생성과 달리 쌍별 선후 판독이 최종 디코딩에 직접 반영된다. 생성 데이터도 사용하지 않아 규정에 어긋나지 않는다. 다시 말해 순서를 '생성'하는 대신 '판별한 뒤 디코딩'한다. 전체 시스템은 [그림 1]에 제시했으며, 3.1 절에서는 판독과 디코딩을, 3.2 절에서는 학습 방법을 설명한다.



[그림 1] 전체 시스템 개요. 프롬프트의 두 확률 판독 → 24 개 순열의 에너지 E(π) → argmax. 추론 시 4-뷰 확률 평균(TTA), 학습 시 정답 유도 세 손실(L_read·L_gap·L_rank)로 지도.

3.1 Pairwise 판독과 순열 에너지 디코딩 (Pairwise Readout & Permutation-Energy Decoding)

완전일치 채점에서 자유 생성의 형식 오류는 곧 실점으로 이어진다. 또한 자유 생성만으로는 이 과제의 구조인 '여섯 쌍의 선후 관계와 하나의 전역 순서'를 충분히 활용하기 어렵다. 2 절의 실험에서도 사람은 네 장을 한꺼번에 배열할 때보다 쌍 단위로 비교할 때 더 정확했다. 이에 전체 순서를 직접 쓰게 하지 않고, 각 쌍의 선후를 확률로 읽는 pairwise 판독(Pairwise Readout)을 사용했다. [그림 2]의 고정 프롬프트가 3 절에서 설명한 구조적 스캐폴드다. 최종 순서를 묻기 전에 여섯 쌍의 선후 질문을 나열해 추론 토큰(중간 판단)을 늘리고, 각 질문을 숫자 바로 앞의 단어 'frame'에서 끊는다. 이 지점에서 다음에 이어질 숫자(1~4)의 확률을 읽으면 새 문장을 생성하지 않고도 모델의 판단을 수치로 얻을 수 있다. 프롬프트에는 숫자가 없으며, 각 판독 위치(□)의 정답 숫자는 학습 목표로만 사용한다.

```
Caption: "{캡션 문장}"
The four images below are shuffled frames from one video.
[프레임 1] [프레임 2] [프레임 3] [프레임 4]

Frame 1 vs frame 2: … the earlier one is frame □  ┐
Frame 1 vs frame 3: … the earlier one is frame □  │
      ⋮                                           │ 6 개 쌍 판독
Frame 3 vs frame 4: … the earlier one is frame □  ┘  (□에서 숫자 1~4 확률을 읽음)

The chronological order … is: frame □, frame □, frame □, frame □
                              └──────── 4 개 시간순 자리 판독 ────────┘
```

[그림 2] 프롬프트(스캐폴드) 구조. 각 □ 위치에서 모델의 다음-토큰 숫자 확률을 판독한다(생성이 아님).

한 번의 순전파(forward pass)에서 두 종류의 확률을 얻는다. 첫째는 각 시간 자리 k 에 어떤 프레임이 올지 나타내는 순서 판독 확률이다. 프레임 f 가 자리 k 에 올 확률을 P_k(f)로 표기한다. 둘째는 각 쌍 (a, b)에서 어느 프레임이 더 이른지 나타내는 쌍 판독 확률이다. f 가 더 이를 확률을 q_ab(f)로 표기한다. 두 확률 모두 모델의 출력층에서 직접 읽는다.

여섯 쌍 판독과 네 자리 판독은 서로 어긋날 수 있으므로 하나의 유효한 순서로 통합해야 한다. 각 후보 순서 π 가 두 판독에 얼마나 부합하는지를 나타내는 순열 에너지(Permutation Energy) E(π)를 정의하고(식 1~3), 이 값이 가장 큰 순서를 선택한다.

$$E(\pi) \;=\; E_{\text{order}}(\pi) \;+\; \lambda\, E_{\text{pair}}(\pi) \qquad\qquad \hat{\pi} \;=\; \arg\max_{\pi} E(\pi) \tag{1}$$

$$E_{\text{order}}(\pi) \;=\; \sum_{k=1}^{4} \log P_k(\pi_k) \tag{2}$$

$$E_{\text{pair}}(\pi) \;=\; \sum_{(a,\,b)\,\in\,P} \Big(\big[\, a \prec_{\pi} b \,\big] \log q_{ab}(a) \;+\; \big[\, b \prec_{\pi} a \,\big] \log q_{ab}(b) \Big) \tag{3}$$

E(π)는 π 가 선택한 항목에 모델이 부여한 확률의 로그 합(로그우도)이다. 값이 클수록 모델이 해당 순서를 정답으로 판단할 가능성이 높다는 뜻이다. 순서 항 E_order 는 각 자리 k 에 놓인 프레임 확률 P_k(π_k)의 로그 합이며(식 2), 쌍 항 E_pair 는 각 쌍에서 π 가 더 이르다고 지정한 프레임 확률의 로그 합이다(식 3; [·]는 조건이 참이면 1 이 되는 Iverson 괄호, a ≺π b 는 'π 에서 a 가 b 보다 이름', λ 는 균형 계수). 24 개 순서를 모두 평가해 E(π)가 가장 큰 순서를 고르므로(식 1 의 argmax), 출력은 항상 유효한 순열이다. 실제 채점에서도 형식 오류로 인한 실점은 0 건이었다.

3.2 인접 오류를 겨냥한 랭킹 학습

학습은 시간 추론을 개선하는 데 초점을 맞췄다. 2 절의 관찰에 따르면 모델은 장면을 대체로 올바르게 이해했지만(깊이 추론은 충분), 이를 선후 관계로 옮기는 과정에는 취약했다. 따라서 vision tower(시각 인코더)는 고정하고 language 부분에만 저순위 어댑터를 붙여 미세조정했다. LoRA 는 원래 가중치를 고정한 채 작은 저순위 행렬만 학습하는 방식이다. 사전학습 가중치를 4bit(NF4)로 양자화해 고정하고 어댑터만 학습하는 QLoRA 를 사용하면 27B 모델도 그래픽카드 한 장에서 학습할 수 있다. 학습 대상을 시간 추론으로 좁혀 과적합 위험과 계산 비용도 함께 줄였다.

$$\mathcal{L} \;=\; \mathcal{L}_{\text{read}} \;+\; \alpha\, \mathcal{L}_{\text{gap}} \;+\; \beta\, \mathcal{L}_{\text{rank}} \tag{4}$$

학습 목적함수는 세 손실의 가중 합으로 구성했다(식 4). 세 손실 모두 하나의 정답 순서에서 규칙적으로 유도한 신호만 사용한다. 판독 손실 L_read, 간격 손실 L_gap, 순위 손실 L_rank 를 차례로 설명한다.

$$\mathcal{L}_{\text{read}} \;=\; -\sum_{k=1}^{4} \log P_k(\pi^{*}_{k}) \;-\; \sum_{(a,\,b)\,\in\,P} \log q_{ab}(e^{*}_{ab}) \tag{5}$$

판독 손실 L_read(식 5)은 여섯 쌍 판독과 네 순서 판독이 각각 정답 프레임을 맞히도록 하는 교차 엔트로피(cross-entropy, 음의 로그우도)다. 여기서 π*는 정답 순서이고, e*_ab 는 쌍 (a, b)에서 실제로 더 이른 프레임이다. 각 판독 위치에서 올바른 프레임을 선택하도록 학습하는 손실이다.

$$\mathcal{L}_{\text{gap}} \;=\; -\sum_{(a,\,b)\,\in\,P} \log g_{ab}(d^{*}_{ab}), \qquad d^{*}_{ab} = \left|\, r^{*}(a) - r^{*}(b) \,\right| \tag{6}$$

간격 손실 L_gap(식 6)은 각 쌍의 두 프레임이 시간상 몇 칸 떨어졌는지(1·2·3 칸)를 맞히도록 한다. 정답 순서에서의 간격을 d*_ab 로, 이를 세 등급(1·2·3 칸)으로 분류하는 작은 분류기를 g_ab 로 표기한다. '어느 쪽이 이른가'뿐 아니라 '얼마나 떨어져 있는가'까지 학습해, 뒤의 순위 손실이 에너지 차이를 학습하는 데 필요한 신호를 보완한다.

$$\mathcal{L}_{\text{rank}} = \sum_{t\,\in\,\{\text{adj},\,\text{med},\,\text{rev}\}} w_t\, \operatorname{softplus}\!\big(m_t - \big(E(\pi^{*}) - E(\pi^{-}_{t})\big)\big) \tag{7}$$

순위 손실 L_rank 는 이 방법의 핵심이며, 이를 인접-겨냥 랭킹(Adjacency-focused Ranking)이라 부른다(식 7). 판독 결과를 정답에 맞추는 것만으로는 부족하다. 24 개 후보 가운데 가장 혼동하기 쉬운 오답은 정답과 한 자리만 다른 순서이기 때문이다. 2 절의 오류 분석에서도 실제 오답은 대부분 이웃한 두 프레임을 뒤바꾸는 형태였다. 따라서 정답 순서의 에너지 E(π*)가 이러한 hard-negative 보다 충분히 높아지도록 직접 학습한다.

오답 유형	정답과의 관계	여유 m_t	가중치 w_t	취지
인접 교환(adj)	이웃 한 쌍만 뒤바꿈	0.4 (작게)	1.0 (크게)	가장 잦고 값비싼 실수 — 집중
중간(med)	2~3 칸 어긋남	0.7	0.5	중간 난이도
역순(rev)	완전히 뒤집음	1.0 (크게)	0.2 (작게)	드문 실수 — 약하게

[표 3] hard-negative 세 유형. 자주 틀리는 인접 오류에 집중(큰 가중치·작은 여유).

식 (7)에서 E(π*)−E(π⁻_t)는 정답 순서의 에너지가 오답보다 얼마나 높은지를 나타낸다. m_t 는 해당 오답 유형에 요구하는 최소 차이(여유)이고, w_t 는 유형별 중요도다. softplus(m_t − (E(π*)−E(π⁻_t)))는 정답이 이미 m_t 이상 앞서면 0 에 가까워지고, 차이가 부족할 때만 벌점을 준다. 여유는 오답이 정답과 비슷할수록 작게 설정했다([표 3]: 인접 0.4, 역순 1.0). 반대로 가중치는 자주 발생하는 인접 오류에 크게(1.0), 드문 역순 오류에 작게(0.2) 두었다. 이를 통해 인접 경계에서 정답의 에너지가 오답보다 높아지도록 학습한다.

학습은 두 단계로 나눈다. 먼저 판독·간격 손실로 기본 판독을 학습하고(Stage A, α=0.35), 이후 순위 손실을 더한다(Stage B, β=0.20).

4. 추론 방법

추론 단계에서는 모델의 위치 편향을 줄이는 데 집중했다. 모델은 때때로 프레임이 화면에 제시된 순서를 그대로 정답으로 내놓았다. 이러한 경향은 캡션이 짧고 시간 단서가 적은 어려운 표본에서 특히 두드러졌다. 그러나 화면상의 위치는 무작위로 정해지므로 프레임의 실제 시간 순서와는 무관하다.

이 문제를 줄이기 위해 입력을 여러 형태로 바꿔 예측을 평균하는 추론 시점 증강 (test-time augmentation)을 적용했다. 구체적인 방법은 식 (8)과 같다.

$$\bar{P} = \frac{1}{4}\sum_{v=1}^{4} P^{(v)}, \qquad \hat{\pi} = \arg\max_{\pi} E(\pi; \bar{P}) \tag{8}$$

같은 네 장의 입력 순서를 네 가지로 바꿔 판독하면 뷰마다 확률이 나온다. 다만 이 확률은 '화면 위치'를 기준으로 하므로 그대로 평균할 수 없다. 각 뷰의 입력 셔플을 되돌려 '실제 프레임' 기준으로 맞춘 확률을 P^(v)로 두고, 네 뷰의 평균 P̄를 구한 뒤 한 번만 디코딩한다(식 8). 이렇게 하면 뷰마다 달라지는 위치 편향이 상쇄되어 더 안정적인 결과를 얻을 수 있다. 추가 학습 없이 이 절차만 적용해도 가장 어려운 구간(짧은 캡션·시간 단서 없음)의 내부 검증 정확도가 약 5.3%p 올랐고, 위치 편향은 절반가량 줄었다. 순전파 비용은 네 배로 늘지만 그래픽카드 한 장에서 처리할 수 있는 수준이었다.

공간축 증강(crop-TTA)도 시험했다. 이 데이터의 프레임은 대체로 피사체가 중앙에 있고 주변부의 변화는 적다. 따라서 화면 중앙을 확대한 crop 뷰를 추가해 평균하면 배경보다 행동과 변화에 더 집중할 수 있고, 순서축 증강(Random4)과 다른 정보를 얻을 수 있다. 실제로 private 점수가 0.91463 에서 0.92276 으로 올랐다. 다만 기한 내에는 실험을 마치지 못해 공식 제출에 포함하지 않았고, 대회 종료 후 late submission 으로 확인했다(공식 결과와 별도; 5 절·7 절).

5. 실험 및 분석

방법을 하나씩 추가했을 때의 누적 성적은 [표 4]와 같다. 모델을 7B 에서 27B 로 확장하자 생성 방식의 public 점수가 0.82024 에서 0.91099 로 올랐다. 여기에 pairwise 판독과 에너지 디코딩을 적용하면 0.93542 로 0.024 가 더 상승했고, 인접-겨냥 랭

킹을 추가한 최종 점수는 0.93891 이었다. 즉 모델 규모를 키워 기본 성능을 높인 뒤에도 pairwise 구조가 추가 성능 향상에 기여했다. 추론 증강(TTA)의 효과는 [표 5]에 정리했으며, [표 4]는 public, [표 5]는 private 점수를 사용했다.

방법 구성 (누적)	public
7B, 생성 (baseline)	0.82024
27B, 생성 (base)	0.91099
27B + pairwise·에너지 디코딩	0.93542
+ 인접-겨냥 랭킹 (최종)	0.93891

[표 4] 각 방법론의 public 기여(누적): 규모 +0.091, pairwise 구조 +0.024, 랭킹 +0.003.

추론 (late, 동일 adapter)	private
Random4 (순서축)	0.91463
+ crop (공간축)	0.92276

[표 5] crop-TTA 의 private 기여(동일 adapter, late): +0.008.

검증 점수의 향상이 항상 public 점수 향상으로 이어지지는 않았다. 예를 들어 검증셋에 맞춘 집계 방식은 검증 점수가 가장 높았지만 public 점수는 0.93368 로 떨어졌다. 이에 하이퍼파라미터를 하나의 검증 결과에 직접 맞추지 않고, 교차 검증으로 선택과 평가를 분리해 과최적화를 줄였다. 최종 모델에 남은 오답도 대부분 인접 오류였으며, 이는 3.2 절의 손실이 겨냥한 오류 유형과 일치했다.

6. 자원 효율성 및 구축 비용

3~4 절의 설계는 자원 효율도 함께 고려했다. 모델이 전체 문장을 생성하지 않고, 정답에 필요한 열 개의 판독 위치(쌍 6 개 + 순서 4 개)에서만 확률을 구한다. 나머지 순서 구조는 규칙에 따라 조립하므로 자유 생성에 드는 비용이 없다. 언어부만 저순위 어댑터로 학습하고 원 모델을 4bit 로 양자화한 결과, 최종 추론은 RTX 3090 1 대(24GB)에서 최대 21.1GiB 의 메모리를 사용했다. test 819 개 표본을 처리하는 데 약 40 분(표본당 약 2.9 초)이 걸렸으며, 형식 오류는 0 건이었다. 학습 시간은 단일 GPU 기준 약 14.4 시간이었다. 데이터 전처리에도 외부 API 를 사용하지 않았고, 필요한 신호는 정답에서 규칙적으로 유도했다(생성 데이터 0 건). 청구액 자료가 없어 구축 비용은 GPU 사용 시간으로 제시한다. 대형 모델 전체를 다시 학습하는 방식과 비교하면 필요한 자원을 크게 줄일 수 있었다.

7. 결론 및 한계

이 보고서에서는 시간순 복원을 순서 생성이 아니라 '판별한 뒤 디코딩하는' 문제로 다뤘다. 문제 구조, 추론 대칭성, 실제 오류 분포를 설계에 반영했고, Qwen3.5-27B 를 그래픽카드 한 장에서 운용해 팀 최고 공식 성적을 얻었다. 최종 성능 차이를 만든 것은 모델 규모 자체보다 과제에 맞춘 구조였다. 순위도 public 10 위, private 12 위, external 10 위로 데이터셋이 바뀌어도 큰 차이가 없었다. 데이터셋에 따라 순위 변동이 컸던 다른 팀들과 비교되는 결과다. 대칭 추론, 교차 검증, rule-safe 설계가 특정 리더보드에만 과도하게 맞춰지지 않았음을 보여 준다.

다만 학습 손실의 각 요소를 분리한 통제 실험이 없어, 성능 향상을 특정 손실 하나의 효과로 단정하기는 어렵다. SWA 와 crop-TTA 는 private 점수를 0.91463 에서 0.92276 으로 높였지만, 대회 종료 후 확인한 결과이므로 공식 성적과 구분해야 한다. 따라서 보고서의 결론은 실험으로 확인한 범위로 한정했다.

참고문헌

[1] E. J. Hu, Y. Shen, P. Wallis, et al., “LoRA: Low-Rank Adaptation of Large Language Models,” in Proc. ICLR, 2022.
[2] T. Dettmers et al., “QLoRA: Efficient Finetuning of Quantized LLMs,” in Proc. NeurIPS, 2023.
[3] Qwen Team, “Qwen3.5-VL Technical Report / Model Card,” 2025.