텍스트로 풀어보는 장면의 재구성

0. 용어 정리

용어	뜻
순열	프레임 4장을 시간순으로 늘어놓는 방법 하나. 가능한 경우의 수는 4! = 24가지다
항등 순열	입력 순서가 그대로 정답인 경우(정답이 `[1, 2, 3, 4]`). 학습 데이터의 15.5%가 여기 해당한다
전수 채점	모델이 답을 직접 써 내려가게 하는 대신, 24가지 후보를 하나씩 모델에 넣어 "이 순서가 맞을 확률"을 재고 가장 높은 것을 고르는 방식
뷰	입력 4장의 자리만 바꿔 만든 같은 문제의 변형. 학습에서는 데이터 증강으로, 추론에서는 여러 뷰의 점수를 평균해 편향을 줄이는 데 쓴다
사전확률 보정	정답 분포가 균등하지 않다는 사실을 점수에 반영하는 항. 후보별 로그 사전확률에 가중치 λ를 곱해 더한다
가중치 평균	같은 설정으로 시드만 바꿔 3번 학습한 뒤 가중치를 산술평균해 모델 하나로 만드는 것. 모델 여러 개를 각각 돌려 투표시키는 앙상블과는 다르며, 최종 산출물은 가중치 파일 하나다

성능을 재는 자는 두 가지다. 하나는 학습 데이터 9,535개 중 954개를 미리 떼어 학습에 쓰지 않은 **검증셋**이고, 다른 하나는 대회 **리더보드**다. 리더보드는 test 819개 중 573개(70%)로 공개 점수를, 나머지 246개(30%)로 최종 점수를 매긴다. 이 둘을 어떻게 나눠 썼는지가 4장의 내용이다.

1. 문제와 접근

1.1 데이터에서 먼저 확인한 것

학습 9,535 샘플, 평가 819 샘플, 이미지 총 38,140장. 한 샘플은 문장 한 개와 뒤섞인 프레임 4장으로 이루어진다. 프레임은 영상에서 뽑은 것이라 해상도가 낮은 편이고(640×360이 60%, 작은 것은 128×128), 문장은 중앙값 26단어다. 정답 라벨 9,535개 중 1,478개(15.5%)가 항등 순열이었다.

1.2 답을 쓰게 하지 말고, 후보를 채점하게 한다

프레임 4장의 순서는 경우의 수가 24가지로 딱 정해져 있다. 그런데 모델에게 답을 문장으로 쓰게 하면 형식이 깨질 수 있고 무엇보다 24개 후보를 서로 견주지 않은 채 앞에서부터 한 글자씩 확정해 버린다. 실제로 미세조정 없이 답을 쓰게 했을 때 정답률은 13.5%(200샘플 확인)로, 아무 생각 없이 `[1, 2, 3, 4]`만 찍는 15.5%보다도 낮았다. 이대로는 쓸 신호가 없다는 뜻이다.

그래서 방향을 바꿔 24개 후보를 전부 모델에 넣고 각 후보의 확률을 재서 가장 높은 것을 답으로 삼았다. 이렇게 하면 형식이 깨질 일이 원천적으로 없고 후보끼리의 비교가 명시적으로 이뤄진다. 비용도 생각만큼 늘지 않는다. 24개 후보가 공유하는 부분(이미지와 문장으로 이루어진 앞부분)의 계산 결과를 한 번만 구해 재사용하면, 후보마다 새로 계산해야 하는 부분은 답 문자열 몇 글자뿐이기 때문이다.

1.3 정답 분포는 학습이 아니라 채점식에 넣는다

항등 순열 15.5%는 문장으로도 이미지로도 구분되지 않았다. 세 차례 따로 확인했지만 이 그룹과 나머지의 문장·픽셀 통계는 같은 분포였다. 반대로 나머지 23가지 순열은 거의 균등하게 나타났다. 즉 이 15.5%는 모델이 데이터를 보고 알아낼 수 있는 정보가 아니라, 정답이 어떻게 분포하는지에 관한 사전 지식이다.

그래서 모델의 점수에 이 분포를 λ만큼 섞는 형태로 분리했다. λ = 0.5일 때 검증 정답률이 58.81%에서 60.27%로 1.46%p 올랐다. 이 분포는 학습 데이터 통계에서만 계산했다.

1.4 순서를 섞는 것 자체가 증강이자 추론 평균이다

입력 4장이 놓인 자리는 원래 의미가 없다. 자리를 바꾸고 정답도 같이 바꾸면 뜻이 보존된 새 학습 샘플이 된다(경우의 수만큼 최대 24배 증강, 덤으로 항등 순열만 찍는 지름길이 막힌다). 같은 변환을 추론에도 적용해 여러 배치의 점수를 평균하면 특정 자리에 쏠리는 편향이 상쇄된다. 뷰 1개에서 8개로 늘리자 검증 정답률이 57.86%에서 60.27%로 올랐다.

2. 방법

2.1 학습

항목	값
베이스 모델	Qwen3-VL-8B-Instruct (2025-10 공개, 규정 기준일 2026-05-31 이전)
학습 방식	LoRA (r=64, α=128) — 언어 모델의 모든 선형층과 이미지-언어 연결부 260개 모듈, 학습 대상 1억 7,900만 파라미터(전체의 약 2%). 이미지 인코더는 고정
학습 목표	프롬프트 뒤에 시간순 나열 문자열(예: `[2, 3, 1, 4]`) 한 줄을 생성하도록 학습
하이퍼파라미터	3 epoch, 학습률 1e-4(코사인 감쇠, 워밍업 3%), 유효 배치 33, bf16, 이미지당 픽셀 상한 768×28×28
소요	시드당 3.7 GPU-h. 32GB 카드는 배치 3, 24GB 카드는 배치 1에 누적 33으로 같은 유효 배치를 재현(최대 메모리 17.7GiB)

학습과 추론이 문자열을 똑같이 다루도록 프롬프트와 정답을 직접 이어 붙여 토큰화하는 경로를 따로 두었다. 자리 섞기는 데이터를 꺼낼 때마다 즉석에서 수행하므로 별도 전처리 비용이 없다.

2.2 추론

한 샘플을 처리하는 절차는 다음과 같다.

1. 뷰 하나를 골라 프레임 4장을 그 순서로 배치하고, 이미지와 문장으로 이루어진 프롬프트를 한 번 계산한다
2. 그 계산 결과를 재사용해 24개 후보 문자열의 확률을 차례로 잰다
3. 뷰 때문에 뒤바뀐 후보 번호를 원래 자리 기준으로 되돌린다
4. 뷰 8개에 대해 1~3을 반복해 점수를 평균하고, 사전확률 보정을 더한 뒤 가장 높은 후보를 답으로 삼는다

점수식은 `S(후보) = 뷰별 로그확률의 평균`이고, 최종 선택은 `argmax [ S(후보) + λ · log P(후보) ]`, λ = 0.5다. 샘플 하나당 채점하는 후보 문자열은 24 × 8 = 192개다.

계산한 점수는 항상 `[샘플 수 × 24]` 행렬로 저장했다. 덕분에 λ를 바꿔 보거나 오류를 분석할 때 GPU를 다시 쓰지 않았고 이 습관이 대회 내내 수십 시간을 아꼈다. 낮은 정밀도 연산과 계산 재사용이 점수를 바꾸지 않는지는 배정밀도 전량 재계산과 대조해 확인했다(차이 0.000000). 채점 코드를 건드릴 때마다 같은 대조를 반복했다.

2.3 모델을 하나로 유지하는 방법

시드 3개(42/1337/7)로 같은 설정을 학습한 뒤, 각 결과를 베이스 모델에 합쳐 변화량을 구하고 그 변화량을 평균했다. LoRA는 두 행렬의 곱이라 행렬을 따로 평균하면 다른 결과가 나오므로 합친 뒤 평균하는 순서가 중요하다. 결과물은 가중치 파일 하나이므로 모델 앙상블 금지 규정에 걸리지 않는다. 이 점은 주최측에 미리 확인받았다. 시드만 바꿨을 때의 점수 흔들림은 0.2~0.3%p였고 평균으로 얻은 이득은 0.2%p 수준으로 크지 않지만 비용도 거의 들지 않는다.

2.4 RTX 3090 24GB 한 장에 맞추기

조치	효과
프롬프트 계산 시 마지막 위치의 출력만 남기도록 수정	어휘 크기만큼 커지는 중간 텐서 약 1.8GiB를 없앰 — 24GB에서 메모리가 터지던 실제 원인이었다. 점수가 바뀌지 않음을 재검증
평균한 가중치를 다시 LoRA 형태(rank 192)로 압축해 배포	17GB짜리 가중치를 새로 쓰지 않고 베이스 모델 + 작은 어댑터로 불러올 수 있다. 복원 오차 1e-07 수준으로 사실상 무손실
메모리 할당 관련 환경변수 사용하지 않음	24GB 카드에서 이 설정이 프로세스를 죽이는 것을 확인해, 환경변수에 의존하지 않는 경로로 확정
이미지 픽셀 상한 고정	해상도가 큰 프레임에서 처리 시간이 튀는 것을 방지

그 결과 test 819개를 끝까지 돌렸을 때 최대 메모리 21.44GiB, 24GB 카드 기준 6.2시간이 걸렸다. 규정 제한 24시간의 26%다. 검증 환경과 같은 24GB 카드에서 실제로 완주시켜 확인한 수치다.

3. 결과

3.1 단계별 정답률 (검증셋 954개 기준)

단계	검증 정답률	공개 점수
항등 순열만 찍기	15.51%	15.53%
미세조정 없이 답 생성 (Qwen2.5-VL-7B)	13.5%	—
LoRA 미세조정 + 전수 채점 (7B, 뷰 4개)	56.6%	87.78%
베이스 모델 교체 (Qwen3-VL-8B, 같은 설정)	58.91%	91.45%
+ 시드 가중치 평균	59.54%	91.45%
+ 뷰 8개 평균, λ = 0.5	60.27%	—
최종: 전체 9,535개로 재학습한 시드 3개 평균	(검증셋도 학습에 포함)	**92.50%**

가장 큰 도약은 "생성에서 채점으로" 바꾸고 미세조정한 단계다(13.5% → 56.6%). 그다음이 베이스 모델 교체(+2.31%p)다. 모델 크기를 32B로 키우는 방향은 오히려 뒤처졌다(4장).

3.2 뷰 개수와 사전확률 가중치 λ

뷰	λ=0	λ=0.5	λ=1.0	3090 기준 소요
1	56.39	57.44	57.86	샘플당 3.4초 (전체 0.8시간)
4	58.60	59.54	59.12	13.7초 (3.1시간)
8	58.81	60.27	59.85	27초 (6.2시간)
16	59.12	60.59	60.06	55초 (12.4시간)

뷰가 적을수록 최적 λ가 커진다. 뷰 평균이 줄여 주던 흔들림을 사전확률이 대신 메우는 셈이라 빠른 모드로 배포한다면 λ를 다시 맞춰야 한다는 뜻이다. 뷰 16개는 검증셋에서 0.32%p 더 나았지만 공개 점수는 0.52%p 낮았다. 최종 설정이 뷰 8개인 이유다(판정 근거는 4.3).

3.3 어디서 틀리는가 (검증셋, 뷰 8개, λ=0.5, 오답 379개)

정답이 몇 등이었나. 틀린 379개에서 정답 순열이 24개 후보 중 몇 등이었는지 세면, 2등이 24.3%, 3~4등 23.7%, 5~7등 16.9%, 8등 이하가 35.1%다. 2등짜리를 전부 맞혔다면 정답률은 69.9%가 된다. 한편 오답을 정답과 비교하면 30.9%는 인접한 두 장의 앞뒤만 뒤바뀐 경우였다. 대부분 같은 장면에서 연달아 나온 프레임들이다.

문장 길이가 성패를 가른다.

문장 길이	≤15단어	16~22	23~30	31단어 이상
샘플 수	251	118	314	271
정답률	30.3%	35.6%	67.5%	90.4%
정답이 8등 이하	28.7%	24.6%	9.6%	0.7%

문장이 길면 사건이 언급된 차례가 곧 시간 순서여서 90%를 맞힌다. 문장이 짧으면 그 단서가 사라지고 프레임의 미세한 변화만 남는데, 여기서 정답률이 30%대로 내려간다. 이 구간이 남은 오류의 본체다.

모델은 자신이 모를 때를 안다. 1등과 2등 후보의 점수 차이가 0.5 미만인 샘플이 전체의 32.7%인데 이들의 정답률은 24.0%다. 점수 차이가 2.0 이상인 42.3%는 정답률 96.5%다. 확신도가 잘 맞아떨어지므로 "확신 없는 구간만 다시 푸는" 방법을 여러 번 시도했지만 모두 실패했다. 확신이 낮은 이유가 모델이 애매해서가 아니라 프레임에 정보가 없어서였기 때문이다.

4. 무엇을 근거로 결정했는가

이 대회에서 가장 오래 붙든 문제는 새 기법이 아니라 "개선인지 아닌지 어떻게 아느냐"였다. 결론부터 적으면 13개 축의 개선안을 시험해 **하나도 채택하지 않았다**. 리더보드 점수가 올랐다는 이유로 설정을 바꾼 적도 없다.

4.1 판단은 검증셋으로 한다

954개를 학습에서 완전히 배제하고 모든 비교를 여기서 했다. 954개에서 정답률의 표준오차는 약 1.6%p이므로 그보다 작은 차이는 우연과 구분되지 않는다. 그래서 두 모델을 비교할 때는 같은 샘플에서 정답/오답이 어떻게 엇갈렸는지 세는 방식(McNemar 검정)을 함께 썼다. **실험을 시작하기 전에 "얼마 이상 좋아지면 채택한다"를 적어 두고** 그 기준으로만 판정했다. 뒤늦게 기준을 고쳐 살린 실험은 없다.

최종 모델만은 예외적으로 검증셋을 포함한 전체 9,535개로 다시 학습했다. 다만 이때는 설정을 이미 고정한 뒤였고 데이터를 늘린 것 외에 아무것도 바꾸지 않았다.

4.2 그 검증셋에도 한계가 있다

검증셋은 학습 데이터에서 떼어낸 것이라 학습 라벨과 성질이 같다. 만약 라벨 일부가 틀려 있다면 검증 점수는 실제 실력보다 낮게 나오고 두 모델의 차이도 그만큼 줄어 보인다. 공개 점수(0.925)와 검증 점수(0.6027)의 관계를 그대로 대입하면 라벨이 어긋나는 비율은 대략 35% 수준으로 추정되는데, 오류 분석에서 "정답이 8등 이하로 밀려난" 덩어리와도 방향이 맞는다.

이 감쇠는 비교 검정에도 똑같이 걸린다. 계산해 보면 954개 검증셋이 유의하게 잡아낼 수 있는 실제 개선폭은 약 2.6%p부터다. 대회 후반에 남아 있던 개선안들은 기대 효과가 그보다 작았다. 그러니 **검증셋만으로는 판정이 불가능한 구간**에 들어와 있었다. 이 사실을 인정하는 것이 이후 결정의 출발점이었다.

4.3 리더보드는 채택이 아니라 기각에만 썼다

판정 불가 구간에서 택할 수 있는 길은 셋이었다. 근거 없이 바꾸거나, 그대로 두거나, 표본이 더 큰 리더보드에서 한 번 확인해 보거나. 우리는 세 번째를 택하되 다음 규칙을 미리 못 박았다.

- 바꾸는 쪽이 입증 책임을 진다. 결과가 애매하면 기존 설정을 유지한다
- 검증셋과 리더보드가 어긋나면 채택하지 않는다
- 성능이 같으면 사전확률 가중치 λ가 작은 쪽을 고른다. 정답 분포 가정에 덜 기대는 설정이 최종 점수 구간에서 더 안전하다고 봤다
- 평가 데이터 자체는 열어 보지 않는다. 우리가 받은 정보는 공식 채널이 돌려준 점수 하나뿐이다

실제 결과는 다음과 같다. 다섯 번 모두 "기존 설정 유지"로 끝났다.

확인한 변경안	검증셋 변화	공개 점수 변화	판정
뷰 16개로 확대	+0.32%p	−0.52%p	기각(뷰 8개 유지)
학습률 2e-4로 재학습	+0.63%p	0.00%p	기각(819개 중 39개 예측이 바뀌었으나 점수는 동일)
유사 프레임 묶어 재정렬	+0.42%p	−0.35%p	기각
이미지 인코더까지 미세조정	+0.42%p	−2.79%p	기각
프레임을 영상으로 묶어 재학습	−0.7%p	−1.22%p	기각

여기서 얻은 규칙은 단순하다. **검증셋에서 1%p 미만으로 나아진 변경은 이 과제에서 신호가 아니다.** 이 기준을 세운 뒤로는 남은 GPU를 소수의 큰 변경에만 썼다. 총 제출 11회 중 후반 5회가 위 확인이고, 나머지는 초기 형식 점검과 단계별 성능 기록이다.

4.4 검증셋 단계에서 끝난 것들

4.3의 다섯 건은 리더보드까지 확인한 경우다. 나머지 여덟 건은 검증셋에서 판정이 끝나 제출까지 가지 않았다. 둘을 합친 열세 건이 앞서 말한 13개 축이다.

축	시도	결과
정규화	라벨 스무딩	프레임워크가 이 모델 계열에서 손실을 잘못 계산하는 문제를 발견. 학습 손실은 정상처럼 보였고 검증셋이 20분 만에 잡아냈다
정규화	R-Drop / weight decay	−1.9%p / −0.6~1.0%p
모델 크기	32B 4bit 미세조정	에폭마다 8B보다 낮음(51.9 / 56.4 / 56.3) → 사전 기준대로 중단
모델 크기	30B MoE	전문가 가중치 구조 때문에 4bit 적용 불가, 로드 자체가 안 됨
학습 설정	LoRA 크기·에폭 수 탐색	둘 다 변화 없음(같이 돌린 학습률은 4.3)
데이터	의심 라벨 정제 후 재학습	원본과 동률. 엇갈린 84개를 사람이 직접 판정해 보니 오히려 원본 쪽이 나았다
데이터	어려운 배치 위주 증강 / 문장 일부 가리기	동률 / 구간별로 상쇄되어 전체는 동률
학습 목표	후보 간 순위를 직접 학습(ranking loss·SLiC)	−1.16%p / 동률
추론 후처리	λ를 샘플마다 조정, 후보 합의 방식(MBR·Kemeny), 저확신 샘플 고해상도 재추론	전부 무효 또는 음수

가장 배운 것이 많았던 실패는 4.3의 이미지 인코더 미세조정이다. 3.3에서 본 "짧은 문장에서 무너진다"는 결과는 자연스럽게 눈이 부족하다는 해석으로 이어졌다. 그래서 그때까지 고정해 두었던 이미지 인코더를 풀어 학습시켰고, 프레임을 영상으로 묶어 다시 학습시키기도 했다. 두 실험이 같은 말을 한다. 해상도가 낮은 프레임에서 미세한 변화를 읽어내려 애쓰는 것보다 문장을 신뢰하는 편이 실제로 더 정확하다. 눈에 자유도를 주면 프레임의 잡음에 과적합할 뿐이다. 모델이 문장에 기대는 것은 결함이 아니라 이 데이터에 맞는 올바른 적응이었다.

4.5 일반화는 어떻게 지켰고, 결과는 어땠나

일반화를 위해 파이프라인에 넣은 장치는 이렇다. 시드 3개 가중치 평균으로 시드별 흔들림을 줄였고, 입력 자리 편향은 뷰 평균으로 상쇄했다. 샘플마다 λ를 바꾸는 적응형 보정은 검증 결과 이득이 없어 폐기했다.

결과를 보면 공개 점수 0.92495에서 최종 점수 0.91056으로 1.44%p 낮아졌다. 최종 점수 표본이 246개라 표준오차가 ±1.8%p이므로 이 차이는 통계적으로 구분되지 않는다. 둘을 합치면 test 819개 전체로는 0.9206이다. 공개 점수에 맞춰 과적합된 흔적은 없다고 본다. 판정이

애매해서 기각했던 세 변경안의 최종 점수도 0.9106~0.9228로(가장 높은 것이 뷰 16개의 0.92276) 우리 것과 같은 오차 범위 안이다. 그 변경들에 실제 효과가 없었다는 4.3의 결론과 맞는 결과다. 반면 크게 벗어났던 두 변경안(이미지 인코더 0.8984, 영상 형식 0.8699)은 최종 점수에서도 확실히 낮아 기각이 옳았음을 확인했다.

5. 한계와 앞으로

남은 오류는 눈을 키운다고 닫히지 않는다. 4.4까지 네 방향으로 지각을 보강해 봤지만 전부 실패했고, 640×360급 프레임에는 인접한 두 장을 가르는 정보 자체가 부족하다는 것이 우리 결론이다. 다음에 시도한다면 프레임 간 차이를 명시적으로 입력에 넣거나, 시간 순서를 미리 학습한 영상 모델로 백본을 바꾸는 쪽일 것이다.

자를 먼저 만들었어야 했다. 검증셋의 분해능(4.2)을 넓히는 길은 검증셋 라벨을 사람이 다시 판정하는 것인데 우리는 84개만 해봤다. 이 작업을 수백 개 규모로 먼저 해두었다면 후반의 판정을 제출 기회에 의존하지 않아도 됐고, 기본 하이퍼파라미터 탐색도 특이한 아이디어보다 앞에 놓을 수 있었을 것이다.

부록 A

```
# 0) 환경: python -m venv .venv_qwen3 && .venv_qwen3/bin/pip install -r requirements_venv_qwen3.lock
# 1) 최종 가중치(시드 3개 평균을 rank-192 어댑터로 압축, 1.1GB) 내려받기
bash scripts/download_weights.sh

# 2) test 819개 채점 → 제출 파일 (뷰 8개, λ=0.5, 3090 기준 약 6.2시간)
.venv_qwen3/bin/python src/score_perms.py --model Qwen/Qwen3-VL-8B-Instruct \
  --adapter outputs/final_soup3_as_adapter --split test --views 8 --lam 0.5 \
  --out outputs/final_soup3_test_tta8

# 3) 형식 검증 (제출한 파일과 일치하는지 확인)
.venv/bin/python src/validate_submission.py outputs/final_soup3_test_tta8_submission.csv
```

처음부터 학습하는 경로와 검증셋 평가 명령은 `README.md`에 있다. 보고서 3.2~3.3의 수치는 저장소에 포함한 점수 행렬로 `scripts/analyze_errors.py`가 GPU 없이 그대로 재현한다.

부록 B

- `src/train_sft.py` 학습 · `src/score_perms.py` 전수 채점·뷰 평균·사전확률 보정
- `src/common.py` 순열 계산과 불변식 검사 · `scripts/soup_to_adapter.py` 가중치 평균을 어댑터로 압축
- `docs/PLAYBOOK.md` 실험별 사전 기준과 판정을 남긴 결정 기록 · `docs/GPU_LEDGER.md` 시간·비용 원장
- 최종 제출 파일 `outputs/20260717_final9535_soup3_tta8_lam05_submission.csv`