Meaningful Moments: VLM-Identified Temporal Importance Labels for Video Action Recognition

Cameron Keith

Spring 2026

A Thesis submitted to the Faculty in partial fulfillment of the requirements
for the degree of Bachelor of Arts in Computer Science

Advised by Professor SouYoung Jin

Dartmouth College
Hanover, New Hampshire

Keywords: video action recognition, vision-language models, temporal importance, pseudo-labels, frame selection, meaningful moments

Abstract

Not every moment in a video is equally useful for recognizing an action. A few decisive moments often carry much of the evidence: the instant an object moves, the peak of a dive, or the placement of one object on another. Yet existing video datasets typically provide clip-level captions, action labels, or task-specific temporal annotations rather than large-scale, action-conditioned labels identifying which short segments carry recognition evidence. This work asks whether vision-language models can identify meaningful moments at scale and whether the resulting labels improve video action recognition.

To investigate this question, I introduce Meaningful Moments (MM), a corpus of about 4.58 million VLM-generated per-segment importance scores around 500,000 class-labeled videos spanning Something-Something v2 (SSv2), Kinetics-400 (K400), and Diving-48 across 622 action classes. Each segment receives both a continuous importance score and a binary pseudo-label. To evaluate these labels, I freeze the video action classifier and vary only the temporal sampling strategy, using an α-parameterized framework that compares importance-led selection against an importance-inverted negative control. This separates gains attributable to the VLM importance signal from gains caused by variable-density sampling alone.

Across all three datasets, VLM-derived importance is informative: at the hard-cut endpoint, keeping VLM-important segments outperforms keeping VLM-designated filler. Its downstream usefulness, however, depends strongly on the dataset's class structure. On Diving-48, where discriminative evidence is concentrated in the dive itself, importance-led fast-forwarding preserves full-video recognition while the inverted control collapses by roughly 30 percentage points. On K400, hard-cut selection beats uniform sampling by 1.45 percentage points. On SSv2, VLM importance labels carry signal but do not recover full-video performance. Continuous-score thresholds and budgets further outperform the oracle-designated binary kept set, and in this annotation pipeline direct scoring matches greedy removal within its observed noise floor while using roughly 25$\times$ fewer oracle calls. A four-VLM cross-oracle study on 369 joint-precheck-pass videos shows moderate segment-level agreement but broadly consistent downstream recognition trends across oracles. Together, these results show that temporal importance is useful but conditional on dataset, recognizer, and sampling protocol.

Acknowledgements

I would like to thank my advisor, Professor SouYoung Jin, for her guidance and support throughout this thesis and my time in the SEE Lab. I sincerely appreciate the time and effort she has spent helping me develop this research, overcome challenges along the way, and grow as a student and researcher. I am also grateful to have taken courses with her, which have provided an important foundation for this thesis. Furthermore, I am grateful for the feedback from other members of the SEE Lab, who have provided invaluable insights and suggestions each week.

I would also like to thank the additional members of my thesis committee, Professor Alberto Li and Professor Sami Saydjari, for their thoughtful feedback, suggestions, and support. I feel fortunate to have taken exceptional classes with both of them during my time at Dartmouth. Their teaching has had a lasting impact on the way I approach research, think through complex problems, and connect technical ideas to broader questions. I am grateful for the role they have played in my academic development, as well as for the time and care they have devoted to helping me improve this thesis.

Finally, I would like to thank my family for their support and encouragement throughout this process. My parents have been a constant source of love and confidence, and I am grateful for everything they have done for me.

Contents

1 Introduction

1.1 Why temporal importance matters

Not every moment in a video is equally useful for recognizing an action. A clip may contain setup, repeated motion, context, or aftermath, while class-discriminative evidence may be concentrated in particular temporal regions: the instant an object moves, the peak of a dive, or the moment an object is placed on another. This temporal variation matters because video action recognition models are typically evaluated on fixed-length or uniformly sampled clips. Uniform sampling is computationally convenient and often effective, but it cannot adapt when the relevant evidence is brief, delayed, or concentrated in only part of the clip. Figure 1 shows a concrete example: in a K400 clip labeled *barbecuing*, the oracle assigns high importance to the grilling segments and low importance to an unrelated cat and indoor scenes in the same clip.

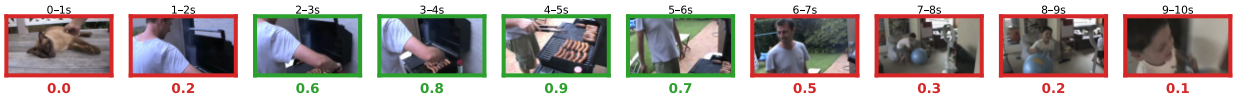

Figure 1: A Meaningful Moments example. A K400 test-split clip labeled *barbecuing* is segmented at $\Delta t = 1.0\,\text{s}$; the number under each segment is the oracle's continuous importance weight, and the border color marks the binary label (green VLM-important, red VLM-designated filler). Off-task content (a cat at 0–1 s, post-action and indoor scenes at 7–10 s) receives low weight, while the grilling action at 2–6 s forms the kept set.

The failure mode depends on the action. Some classes hinge on brief transitions, such as an object changing position. Others depend on a concentrated motion phase, such as the rotation of a dive. Still others require boundary-state information before or after the action, such as where an object ends up relative to another object. In each case, the question is not simply which frames are visually salient or high-motion, but which short temporal segments carry evidence for recognizing the target action class.

Existing datasets and methods address parts of this problem. Action recognition datasets provide class labels [7, 12, 16]; large video-language corpora provide captions or weak language supervision [21, 30, 5]; temporal grounding datasets localize query-relevant moments [14, 9, 15, 27]; frame-selection methods learn policies for choosing clips or frames [31, 13]; and saliency methods estimate model-specific importance post hoc [26, 22]. What is missing is a reusable, large-scale corpus of action-conditioned segment labels: annotations that identify which short temporal segments carry class-discriminative evidence in class-labeled action videos.

1.2 Meaningful Moments and the evaluation framework

To address this gap, I introduce *Meaningful Moments* (MM), a corpus of action-conditioned segment labels for video action recognition. MM contains approximately 4.58 million VLM-generated per-segment importance scores over approximately 500,000 class-labeled videos across 622 action classes. The corpus spans three established action recognition datasets: SSv2, K400, and Diving-48. Each short temporal segment receives both a continuous importance score and a binary importance pseudo-label, allowing the same annotation to be used either as a fine-grained ranking signal or as a hard kept/dropped segment decision.

Producing this kind of annotation with human labelers at the scale of hundreds of thousands of videos would be expensive and difficult to standardize. MM therefore uses a Qwen3-VL-32B

oracle [2] with dataset-specific prompts to assign segment-level importance scores conditioned on the video's action label. These labels are pseudo-labels, not human ground-truth annotations of temporal importance; I test their value empirically through downstream recognition. The three datasets stress different temporal structures: Diving-48 contains fine-grained athletic motions whose discriminative content is concentrated in the dive itself; K400 contains broad activities that are often visible throughout the clip; and SSv2 contains object interactions where boundary states and spatial relations can matter as much as the motion itself.

MM is not generic visual saliency, event-boundary detection, or video captioning; it is action-conditioned temporal evidence labeling over class-labeled videos. This distinction matters because visually salient or high-motion segments are not necessarily those that distinguish the target action. I therefore evaluate MM not by asking whether the labels match an abstract notion of visual importance, but by asking whether they improve or preserve recognition when used to control which parts of a video a classifier sees.

To make this test controlled, I pair the corpus with an α-parameterized sampling framework. The positive condition samples VLM-important segments densely and samples VLM-designated filler segments sparsely, producing an importance-led fast-forward through the video. The matched negative control inverts the rule, sampling filler densely and important segments sparsely. Under this reporting convention, $\alpha = 1$ corresponds to uniform sampling over the full video, while Keep-Important and Keep-Filler represent the hard-cut $\alpha = 0$ endpoint. This shared endpoint structure lets the experiments separate gains attributable to the VLM importance signal from gains due to variable-density sampling itself.

1.3 Contributions and findings

This thesis makes three contributions.

1. **Meaningful Moments.** I introduce a large-scale corpus of VLM-generated per-segment temporal importance scores and binary pseudo-labels for class-labeled videos across SSv2, K400, and Diving-48.

2. **Controlled evaluation framework.** I define an α-parameterized sampling protocol with an importance-inverted negative control to test whether temporal-selection gains come from the VLM importance signal rather than from variable-density sampling alone.

3. **Empirical and oracle-validation analysis.** I show that VLM-derived importance is informative across all three datasets, but that its downstream usefulness depends on dataset structure: high-keep selection works best on Diving-48, aggressive selection works best on K400, and on SSv2 selection remains below full-video performance. Additional analyses show that continuous importance scores outperform oracle-designated kept sets, direct scoring matches greedy removal within its observed noise floor while using roughly 25$\times$ fewer oracle calls, and a four-oracle study shows moderate segment-level agreement but broadly consistent downstream recognition trends across the strongest tested oracles.

Together, these contributions support the central claim of the thesis: VLM-derived temporal importance labels are useful for video action recognition, but their downstream value is conditional. The same importance signal can preserve or slightly improve recognition on Diving-48, improve recognition under aggressive selection on K400, and remain insufficient to recover full-video performance on SSv2, where boundary-state and spatial-relation context is necessary.

2 Related Work

2.1 Video datasets and temporal annotation

Video action recognition has been shaped by large class-labeled datasets that define both the recognition targets and the evaluation protocols used by modern recognizers. SSv2 emphasizes object interactions and visual common-sense transformations; K400 covers a broad set of human actions in short web videos; and Diving-48 stresses fine-grained motion differences in an athletic domain [7, 12, 16]. These datasets provide the class labels on which this thesis builds, but their supervision is video-level or clip-level: they identify what action is present, not which short temporal segments carry the evidence needed to recognize it.

Large video-language corpora provide a different form of supervision. HowTo100M scales weak video-language learning through narrated instructional videos; InternVid and Panda-70M expand this direction with large-scale video-text data and teacher-generated captions [21, 30, 5]. These corpora are valuable for pretraining and multimodal representation learning, but their annotations are primarily descriptive. A caption describes what happens in a clip; it does not identify which short segments are most useful for recognizing a target action class.

Temporal grounding and dense captioning datasets localize events more directly. ActivityNet Captions provides temporally localized event descriptions; DiDeMo localizes moments from natural-language queries; QVHighlights combines query-based moment retrieval with highlight detection; and MAD grounds language in long movie videos using audio descriptions [14, 9, 15, 27]. This line of work is close to MM because it attaches language or relevance judgments to temporal regions. QVHighlights is the closest point of comparison because it includes saliency scores, but those scores are query-conditioned highlights rather than action-conditioned evidence labels for a known recognition class. MM instead asks which segments of a class-labeled action video carry evidence for recognizing the target action.

Temporal action localization provides another nearby comparison. Datasets and benchmarks such as THUMOS and ActivityNet, together with methods such as Structured Segment Networks, ask models to identify when action instances occur in untrimmed videos [11, 8, 33]. These tasks localize action intervals, but an action interval can still contain setup, repeated motion, distractors, and aftermath. MM instead assigns action-conditioned importance scores to short segments, asking which parts of a video carry class-discriminative evidence rather than merely where an action begins and ends.

MM therefore differs from existing video datasets in the object it annotates. It is not a captioning corpus, a query-grounding benchmark, or a temporal action localization dataset. It adds a reusable temporal evidence layer on top of class-labeled action videos: dense, action-conditioned segment labels over the full video timeline.

2.2 Frame, segment, and token selection for video recognition

A separate line of work studies how to reduce computation by selecting only part of a video for recognition. AdaFrame adaptively chooses frames on a per-video basis for fast video recognition, while SCSampler learns to sample salient clips from long videos and reports both accuracy and efficiency gains [31, 13]. OCSampler further compresses a video into a single informative clip using an instance-specific sampling policy [17]. These methods show that temporal selection can matter, but they treat selection as a learned policy inside a recognition system. They do not provide a reusable corpus of segment-level evidence labels that can be used independently of a particular selector architecture.

Recent efficient-video methods also operate at the token level. STTS selects informative tokens in both spatial and temporal dimensions for efficient video transformers, formulating token selection as a ranking problem over video tokens [29]. EVAD uses spatiotemporal token dropout and context refinement for efficient video action detection [4]. More general vision-transformer efficiency methods, such as DynamicViT and Token Merging, prune or merge visual tokens inside the model computation graph [23, 3]. These approaches reduce computation by changing how a model processes its tokens. MM instead supplies external temporal evidence labels at the segment level. The distinction is between learning a selector and releasing a reusable selection signal.

The recognizers used in this thesis are not themselves contributions. VideoMAE and V-JEPA 2 provide the frozen recognition backbones used to test whether MM labels change recognition behavior under controlled sampling [28, 1]. Holding the classifier fixed is important: it makes the temporal sampling strategy, not recognizer training, the primary intervention. In this sense, MM complements selection and pruning methods. Prior work asks how to select frames, clips, or tokens; MM provides large-scale action-conditioned labels that can evaluate or supervise selection independently of any particular selector or recognizer.

2.3 Vision-language models as annotators and judges

MM relies on a vision-language model oracle rather than human segment-level annotation. The primary oracle is Qwen3-VL-32B, a dense variant in the Qwen3-VL family that supports image and video inputs, long multimodal context, and temporal grounding [2]. The cross-oracle study compares this primary signal against two proprietary frontier VLMs, GPT-5.5 via Azure ChatAPI and Gemini 3.1 Pro, and one open-weights comparator, InternVL3-38B. InternVL3-38B provides an archival, open-weights multimodal reference point for the cross-oracle analysis [35].

Using foundation models as scalable evaluators is now common in language model evaluation. G-Eval uses GPT-4 with task rubrics to evaluate natural language generation, while MT-Bench and Chatbot Arena study LLM-as-judge evaluation and document both its usefulness and its biases [18, 34]. This literature motivates the general idea of model-generated judgments, but most of it concerns text quality, preference judgments, or open-ended assistant responses. MM extends the model-as-annotator idea to a different setting: temporally localized evidence labels for video action recognition.

Video-language benchmarks make clear why such labels should be treated as pseudo-labels rather than ground truth. Video-MME evaluates multimodal LLMs across short, medium, and long videos with multimodal inputs, and EgoSchema tests very long-form video question answering with substantial temporal reasoning demands [6, 20]. These benchmarks show that video-capable VLMs can perform multimodal video reasoning, while also exposing persistent weaknesses in temporal localization, long-context understanding, and fine-grained event reasoning. Accordingly, this thesis does not assume oracle correctness. I evaluate MM labels operationally: by comparing importance-led selection against an importance-inverted control, by measuring downstream recognition, and by checking cross-oracle agreement and downstream recognition consistency across multiple VLMs.

This thesis therefore treats VLM outputs as pseudo-labels rather than human ground truth. Because human temporal-importance judgments are not available at this scale, the immediate question is not whether the labels perfectly match human annotations, but whether they are consistent enough and useful enough to support controlled recognition experiments. Human verification of segment-level importance is reserved as a limitation and future-work direction.

2.4 Importance, saliency, and counterfactual evidence

The term "importance" has several meanings in adjacent literatures. Generic saliency asks what is visually conspicuous; model-specific attribution asks what affected one trained model's prediction; and action-conditioned evidence asks what helps recognize a target action class. MM targets the third notion.

Visual saliency and attribution methods estimate which parts of an input affect a trained model's prediction. Grad-CAM localizes class-discriminative image regions using gradients, while RISE estimates black-box importance maps by randomized input masking [26, 22]. LIME and SHAP provide model-agnostic local explanations and additive feature-attribution frameworks, respectively [24, 19]. Occlusion-based analysis similarly probes importance by removing parts of the input and observing changes in prediction [32]. These methods are important predecessors, but they are usually post hoc and model-specific: they explain a particular model's prediction on a particular input.

Removal-based evaluation has also been used to test whether attribution rankings are meaningful. RISE popularized insertion and deletion curves for evaluating black-box visual explanations; ROAR evaluates interpretability methods by removing features and retraining; and ROAD proposes a remove-and-debias strategy to reduce confounds in perturbation-based attribution evaluation [22, 10, 25]. MM borrows the logic of these evaluations: if an ordering is meaningful, recognition should degrade differently when high-ranked versus low-ranked evidence is removed or inserted. In this thesis, however, the ranked objects are not pixels or image regions but short temporal video segments.

The greedy-removal baseline used in this thesis connects MM to counterfactual explanation: one can try to identify a minimum sufficient set by repeatedly removing segments and asking whether the action remains recognizable. At corpus scale, however, iterative removal is expensive and unstable. The direct-scoring pipeline used for MM instead asks the oracle to score all segments in a single call. The Results section compares these modes directly, showing that direct scoring matches greedy removal within its observed noise floor while using substantially fewer oracle calls. MM therefore differs from post hoc explanation by producing reusable, action-conditioned temporal evidence labels before training or evaluating a downstream selector.

3 Methodology

3.1 Approach overview

The MM pipeline has two stages: corpus construction and downstream validation. In the construction stage, each source video is paired with its known action label, partitioned into short temporal segments, and submitted to a Qwen3-VL-32B oracle. The oracle returns segment-level importance scores, an oracle-designated kept set, and video-level precheck metadata. These outputs define the MM pseudo-labels: a continuous importance weight for every segment and a binary VLM-important / VLM-designated-filler label.

In the validation stage, I hold video action classifiers fixed and vary only the temporal input sampling strategy. The core evaluation protocol uses an α-parameterized variable-density sampler: the positive condition samples VLM-important segments densely and VLM-designated filler sparsely, while the matched negative control inverts this rule. This design tests whether recognition changes are attributable to the VLM importance signal rather than to variable-density sampling alone.

The remaining protocols test robustness and operating-point sensitivity. Threshold and budget sweeps evaluate the retained continuous scores rather than the binary kept set; insertion/deletion

curves compare the VLM ordering against random, temporal, motion, and anti-VLM orderings; temporal-order shuffling tests whether selected inputs rely on segment order; cross-oracle validation measures how stable the pseudo-labels are across four video-capable VLMs; and oracle-call accounting compares direct scoring against iterative greedy removal.

Figure 2 summarizes the construction stage.

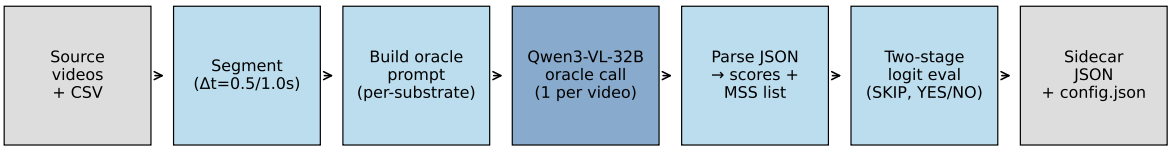

Figure 2: The MM construction pipeline. Source videos and their class labels are segmented on a fixed grid ($\Delta t = 0.5$ or 1.0 s), paired with a dataset-specific direct-scoring prompt, and scored by Qwen3-VL-32B in one oracle call per video. The parsed response yields per-segment scores and a kept set, the two-stage logit rule yields precheck metadata, and the result is written as one sidecar JSON per video.

3.2 Datasets and task structure

MM builds on three established video action-recognition datasets that stress different temporal structures (Table 1): SSv2, K400, and Diving-48. SSv2 contains object-interaction classes in which boundary states and spatial relations often matter: recognizing a manipulation may require knowing where an object began, how it moved, and where it ended. K400 contains broad human activities that are often visible through much of the clip, so the discriminative evidence is frequently distributed rather than confined to a short transition. Diving-48 contains fine-grained athletic classes in which the discriminative content is concentrated in the dive motion itself.

This diversity is important for evaluating temporal importance. If VLM importance labels are useful only when evidence is concentrated, the effect should be clearest on Diving-48. If selection can remove irrelevant setup or background while retaining action evidence, it should help on K400. If many labels depend on temporal boundary states, selection may remain informative while still failing to recover full-video performance, as expected on SSv2. The three datasets therefore provide a stress test for whether VLM-derived importance is generally informative and whether it is useful under fixed recognizers and controlled sampling.

Figure 3 shows example classes from each dataset and illustrates the three temporal regimes: SSv2 templates hinge on object state and relation, K400 classes are often recognizable from activity context, and Diving-48 classes differ only in fine-grained motion attributes.

3.3 Meaningful Moments label construction

Meaningful moments are defined counterfactually. A segment matters for a labeled action to the extent that the action can no longer be identified without it, and the object of interest is the smallest set of segments that remains sufficient for an oracle to recognize the labeled action. Formally, for a video v with label c segmented as $(s_0, \ldots, s_{T-1})$, the target is a minimum sufficient subset

$$K^\star \in \arg\min_{K \preceq v} |K| \quad \text{s.t.} \quad q(K, c) > \tfrac{1}{2},$$

Table 1: Task structure of the three action-recognition datasets used in MM. The datasets differ in how class-discriminative evidence is distributed over time. Video counts are source-video annotation attempts over the splits used here; median durations are computed over the precheck-passed corpus.

Dataset	Videos	Classes	Med. dur. (s)	Temporal structure	Main role in this thesis
SSv2	220,847	174	3.75	Object interactions and boundary states	Tests relation- and state-dependent actions
K400	298,337	400	10.00	Broad activities often visible throughout	Tests aggressive action-focused selection
Diving-48	16,997	48	4.10	Fine-grained motion concentrated in the dive	Tests temporally concentrated evidence

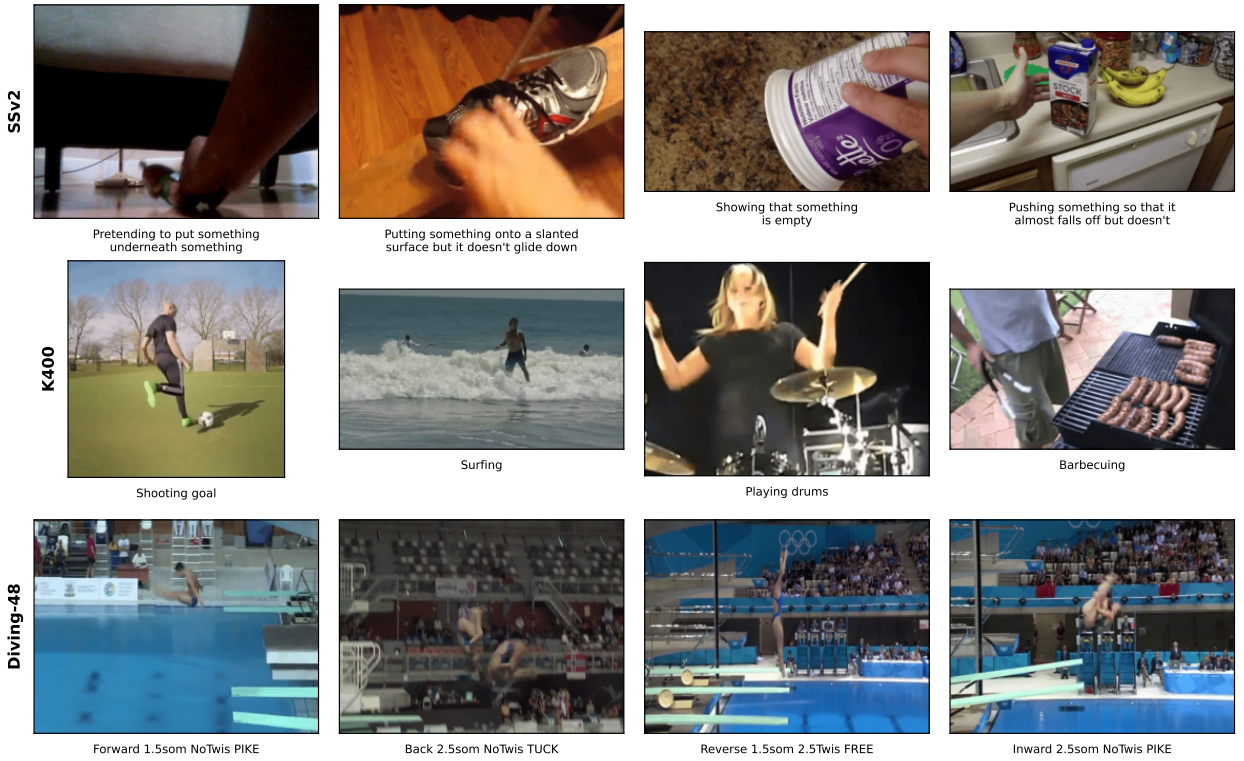

Figure 3: Example classes from the three datasets. SSv2 templates depend on object states and spatial relations; K400 classes are broad human activities; Diving-48 classes share the pool context and differ only in takeoff group, somersault count, twist count, and flight position.

where $K \preceq v$ ranges over ordered subsequences of segments and $q(K, c) = P(\text{YES} \mid \neg\text{SKIP})$ is the oracle's affirmation probability that action c remains identifiable from K, computed from decision-token logits exactly as in Section 3.3.4. Greedy removal operationalizes this definition directly (formal procedure in Appendix A.2): starting from the full video, segments are removed one at a time for as long as the oracle still affirms that the action is identifiable from what remains, and the segments that survive form a minimal sufficient subset. Greedy removal, however, is expensive

(each iteration queries every remaining candidate, about 25 oracle calls per eight-segment video in the pilots) and unstable across repeated runs. The production corpus therefore uses direct scoring, a single oracle call that returns per-segment importance scores and an explicit kept set, treated as a one-call approximation of the same counterfactual target. Whether this approximation is faithful is an empirical question: Section 4.6 shows that direct scoring matches greedy removal within greedy's own run-to-run noise floor while using roughly 25× fewer oracle calls.

Figure 4 traces a real greedy-removal run from an early pilot on an SSv2 video labeled *putting a banana that can't roll onto a slanted surface, so it stays where it is.* Six of the eight segments are removed before the oracle refuses to discard either remaining segment. Direct scoring on the same video keeps four segments instead, a concrete instance of the mode-level disagreement quantified in Section 4.6.

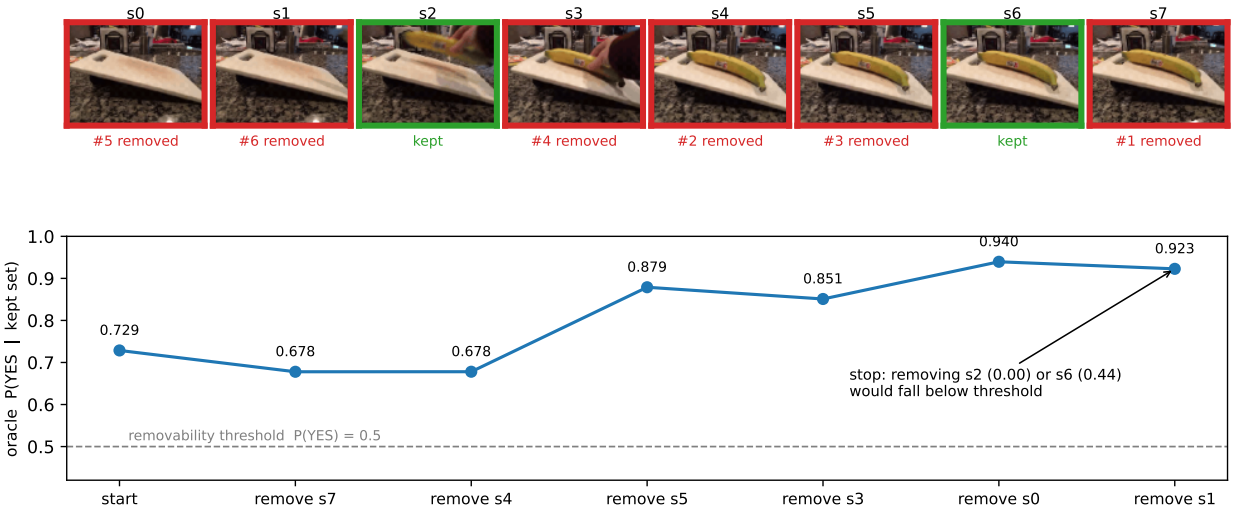

Figure 4: A real greedy-removal trace (SSv2 video 69417, February pilot). Top: the eight 0.5 s segments; green marks the surviving kept set $\{s_2, s_6\}$ and red marks removed segments with their removal order. Bottom: the oracle's two-way $P(\text{YES})$ after each removal. The run stops when removing either remaining segment would drop $P(\text{YES})$ below the 0.5 removability threshold (s_2: 0.00, s_6: 0.44). Direct scoring on the same video keeps $\{s_2, s_3, s_4, s_5\}$.

3.3.1 Oracle and dataset-specific prompts

The primary MM annotations are generated by Qwen3-VL-32B using a direct-scoring prompt. Each oracle call receives the source video, the known action label, and a fixed temporal segmentation of the video. The prompt asks the oracle to score each segment according to how much that segment helps recognize the given action class. The action label is therefore part of the annotation target: MM labels are action-conditioned temporal evidence labels, not generic saliency labels.

The production prompts are dataset-specific. The common prompt structure asks for action-conditioned evidence, but the wording is adapted to the dataset: SSv2 prompts emphasize object-state changes and spatial relations, K400 prompts emphasize broad activity evidence, and Diving-48 prompts emphasize fine-grained diving phases. Appendix A gives the full prompt templates, output schemas, and generation settings.

The direct-scoring response contains three kinds of information: a video-level precheck decision, per-segment importance scores, and an oracle-designated kept set. The direct-scoring versus greedy-

removal comparison is treated as an empirical result in Section 4.6, not as an assumption in the construction procedure.

3.3.2 Temporal segmentation

The basic annotation unit is a temporal segment. For a video of duration D, I partition the timeline into $\lceil D/\Delta t \rceil$ exhaustive segments. Segment i starts at $i\Delta t$ and ends at $\min((i+1)\Delta t, D)$. Segment indices are zero-based in the stored records, and the final segment is clipped to the video duration when D is not an exact multiple of Δt.

I use $\Delta t = 0.5$ seconds for SSv2 and Diving-48, and $\Delta t = 1.0$ second for K400 (Table 2). The shorter grid is used for SSv2 and Diving-48 because their discriminative evidence can occur over brief state changes or motion phases. K400 clips are longer and more activity-centric, so a one-second grid reduces oracle payload while preserving the temporal granularity needed for broad activity evidence. Appendix C collects additional segmentation and protocol details.

Table 2: Temporal segmentation used for MM annotation. Mean segment counts are computed over the precheck-passed headline corpus.

Dataset	Segment duration Δt	Mean segments / video
SSv2	0.5 s	8.0
K400	1.0 s	10.1
Diving-48	0.5 s	9.3

3.3.3 Importance scoring and binary label derivation

The oracle returns two related but distinct segment-level signals. First, it assigns each segment an integer importance score on a 0–100 scale. I normalize this score to a continuous weight $w_i \in [0, 1]$ by dividing by 100. The continuous weight is retained as a ranking signal and is used directly in the threshold, budget, and insertion/deletion protocols. Figure 5 shows the resulting weight distributions: all three datasets are strongly bimodal, with a low-importance band, a high-importance band, and relatively sparse intermediate mass.

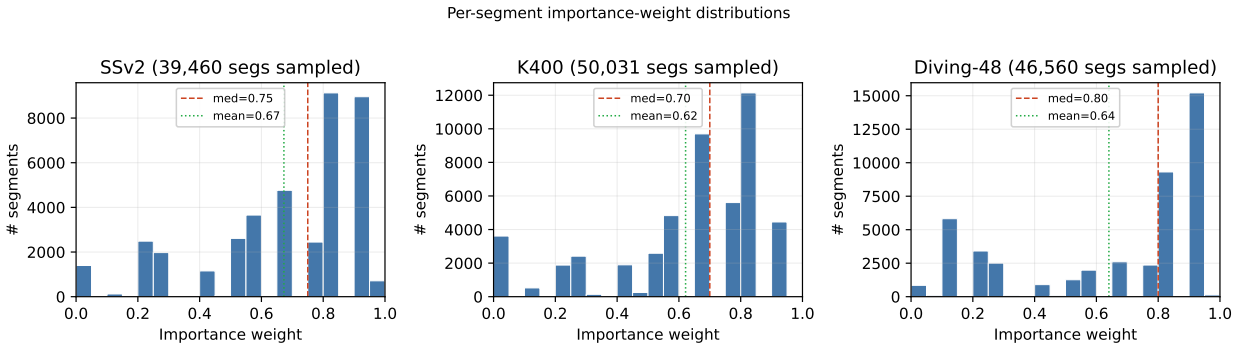

Figure 5: Per-segment importance-weight distributions over sampled production segments (about 40–50K segments per dataset). All three datasets are bimodal, with a low-importance band, a high-importance band, and sparse intermediate mass. Vertical lines mark the median and mean (SSv2 0.75/0.67, K400 0.70/0.62, Diving-48 0.80/0.64).

The scores also have a characteristic temporal signature. Figure 6 maps importance onto normalized clip position: every dataset concentrates importance mid-clip and depresses it at the edges, with Diving-48 the most sharply concentrated (kept fraction 0.99 at its peak versus 0.00 in the first and last bins), K400 the flattest, and SSv2 falling steeply in the final fifth of the clip.

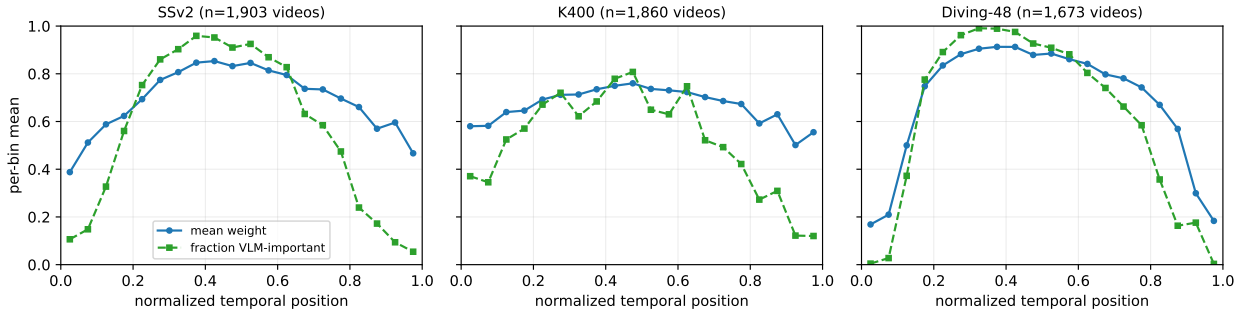

Figure 6: Where importance concentrates over time. Per-bin mean continuous weight and fraction labeled VLM-important by normalized temporal position (20 bins; about 2,000 sampled precheck-passed train videos per dataset, seed 42). Kept fraction peaks mid-clip on all three datasets (SSv2 0.96 at position 0.38, K400 0.81 at 0.47, Diving-48 0.99 at 0.33) and falls toward the clip boundaries (first/last bins: SSv2 0.11/0.05, K400 0.37/0.12, Diving-48 0.00/0.00).

Second, the oracle returns an explicit kept-set field, named `minimum_sufficient_set` in the raw output. In the production corpus, the released binary label is derived from this oracle-stated kept set. Segments in the kept set are labeled VLM-important; all other segments are labeled VLM-designated filler. If the kept-set field is omitted, the parser falls back to the inclusive threshold rule $w_i \geq 0.5$.

Although the prompt instructs the oracle to populate the kept set with segments scoring at least 50, the kept-set field and the continuous score field are not always identical in practice. I therefore treat the binary label and the continuous weight as distinct oracle outputs. This distinction is important for the evaluation: the α-sweep uses the production binary labels, while the threshold and budget sweeps test whether the retained continuous weights provide better operating points.

The parser includes a neighbor-interpolation fallback for omitted segment scores. If both adjacent segments have scores, the missing value is set to their mean; if only one adjacent segment has a score, that value is used; if no adjacent score is available, the value is set to zero. This fallback fired for only one segment across the production corpus, and the affected segment does not appear in any headline evaluation pool. In practice, every headline-evaluation segment carries an oracle-provided score.

3.3.4 Precheck, logit-derived quality metadata, and filtering

Each direct-scoring response also contains video-level precheck metadata. The precheck asks whether the labeled action is visible and scorable in the video. It returns one of three decisions: YES, NO, or SKIP. NO indicates that the oracle judged the labeled action not visibly present or not scorable; SKIP indicates that the input was ambiguous, corrupted, or otherwise unsuitable.

I compute the precheck quantities from decision-token logits rather than from model-written confidence text. Let $\ell_{\text{YES}}, \ell_{\text{NO}}, \ell_{\text{SKIP}}$ be the logits assigned to the YES, NO, and SKIP decision tokens. I compute

$$P(\text{SKIP}) = \frac{\exp(\ell_{\text{SKIP}})}{\exp(\ell_{\text{YES}}) + \exp(\ell_{\text{NO}}) + \exp(\ell_{\text{SKIP}})}$$

and

$$P(\text{YES} \mid \neg\text{SKIP}) = \sigma(\ell_{\text{YES}} - \ell_{\text{NO}}).$$

A video passes precheck when $P(\text{SKIP}) \leq 0.5$ and $P(\text{YES} \mid \neg\text{SKIP}) > 0.5$.

These probabilities are video-level visibility and scorability metadata. They are not calibrated estimates of human correctness, and they are not segment-level importance scores. Precheck-failed sidecars still retain segment scores, but the headline MM corpus statistics and all headline evaluation pools use the precheck-passed subset. Appendix B expands the corpus-quality details.

3.4 Meaningful Moments corpus statistics and release format

3.4.1 Corpus populations

I distinguish four corpus populations. An *annotation attempt* is any source video submitted to the oracle. A *scored record* is a video for which the oracle returned segment scores, regardless of precheck status. A *precheck-passed record* is a scored record that passes the logit-derived precheck rule above. A *headline evaluation pool* is a pinned subset of precheck-passed records used for recognition experiments.

The MM corpus consists of 499,299 precheck-passed videos and 4,576,082 per-segment importance scores across 622 action classes. The underlying sidecars contain 4,912,308 scored segment records when precheck-failed videos are included. I report the precheck-passed subset as the corpus because it is the subset used for corpus statistics and evaluation.

3.4.2 Annotation outcomes and corpus composition

Table 3 reports production annotation outcomes by dataset. Parse failures are separated from precheck decisions. Production parse failures are zero across all three datasets; filtering is dominated by precheck-NO decisions, especially on Diving-48.

Table 3: Production annotation outcomes by dataset. Parse failures are separated from precheck decisions. Precheck-NO means the oracle judged the labeled action not visibly present; precheck-SKIP means the oracle flagged the input as ambiguous, corrupted, or otherwise unsuitable.

Dataset	Attempts	Parse fail	Precheck-NO	Precheck-SKIP	Precheck-pass
SSv2	220,847	0	10,715	290	209,842
K400	298,337	0	23,264	6	275,067
Diving-48	16,997	0	2,577	30	14,390
Total	536,181	0	36,556	326	499,299

The corpus label space totals 622 action classes: 174 SSv2 classes, 400 K400 classes, and the 48-class Diving-48 label space. Diving-48 defines 48 classes, but one of them has no videos in either the train or validation indices used here, so the corpus represents 47 of the 48 defined classes and the headline evaluation pool likewise covers 47. Figure 7 shows the per-class label coverage on the train split, and Figure 8 shows that MM coverage tracks per-class source-video counts at parity for every represented class. Appendix B reports class-coverage summaries and per-class count distributions.

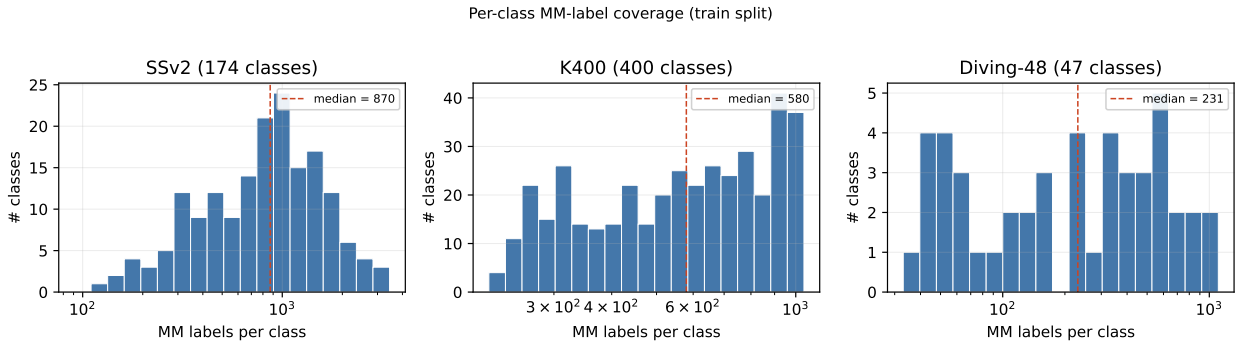

Figure 7: Per-class MM-label coverage on the train split. Median labels per class: SSv2 870, K400 580, Diving-48 231 (over the 47 represented classes).

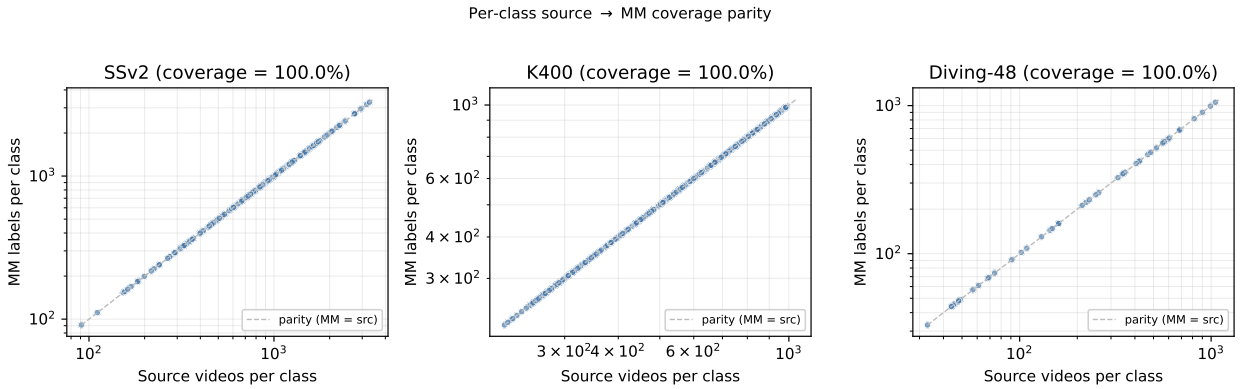

Figure 8: Per-class source-to-MM coverage parity. MM labels track per-class source-video counts on the parity line for every represented class in all three datasets.

3.4.3 Stored label fields

Each released segment record (Table 4) contains the source video identifier, dataset, split, action label, zero-based segment index, start and end timestamps, continuous weight, binary label, precheck metadata, oracle identifier, and prompt version. In the published release (`https://huggingface.co/datasets/ckeith/meaningful-moments`, tag `v1.0`) these fields are carried by one record per video with a nested per-segment list, shipped in two parallel forms: Parquet tables (one configuration per dataset, one split per source split, loadable directly with `datasets.load_dataset`) and the canonical per-video JSON sidecars packaged as deterministic tar shards. The `dataset` and `split` fields are realized as the Parquet configuration/split names rather than per-record columns, and the oracle identifier and prompt version are pinned once per annotation run in a released `config.json` (model revision hash, prompt SHA-256) rather than repeated per record. The release is designed to be usable without local filesystem paths or raw sidecar structure.

The field `frequency` is omitted from the release because it duplicates `weight` in direct-scoring mode (and, in the cross-oracle supplement, can carry stale pre-recovery values). `video_path` is rewritten to a substrate-relative path (`SSv2/...`, `k400/...`, `diving48/...`) so that no local filesystem layout is released while the join to source videos remains trivial. Raw oracle text, rationales, logits, and kept-set provenance are released in the per-video JSON sidecars; the Parquet tables mirror the sidecars field-for-field except for the verbatim `raw_output` string, which is recoverable

Table 4: Core MM fields. The core table stores the segment-level labels and the metadata needed to reproduce the headline precheck-passed subset.

Field	Meaning
`video_id`	Source video identifier.
`dataset`	Source dataset: SSv2, K400, or Diving-48.
`split`	Source split when available.
`action_label`	Known action label used to condition scoring.
`segment_index`	Zero-based segment index.
`start_s`, `end_s`	Segment start and end timestamps in seconds.
`weight`	Continuous importance score $w_i \in [0, 1]$.
`label`	Binary label: important or unimportant.
`precheck_passed`	Whether the video passed the precheck rule.
`precheck_decision`	YES, NO, or SKIP precheck decision.
`logit_confidence`	$P(\text{YES} \mid \neg\text{SKIP})$.
`logit_p_skip`	$P(\text{SKIP})$.
`oracle_model`	Oracle model identifier.
`prompt_version`	Prompt template/version identifier.

from the sidecar archives.

3.5 Core evaluation protocol

3.5.1 Unit of analysis and notation

Let v denote a source video and c its known action label. After temporal segmentation, v is represented as a sequence of segments $(s_0, \ldots, s_{T-1})$, where each segment s_i has start and end timestamps. The oracle assigns each segment a continuous weight $w_i \in [0, 1]$ and a binary label $\ell_i \in \{\text{VLM-important}, \text{VLM-designated filler}\}$. Evaluation operates at the video level: for each sampling condition, the segment labels and weights define a sampled recognizer input, and a frozen classifier produces a video-level prediction.

The key methodological requirement is to isolate the VLM importance signal from the sampling mechanism. To do this, the positive and negative conditions use the same frame budget, the same recognizer, the same video pool, and the same preprocessing. They differ only in whether the sampler treats VLM-important or VLM-designated-filler segments as dense regions.

3.5.2 Variable-density sampling and the α axis

The core protocol uses an α-parameterized variable-density sampler. For each segment, the sampler assigns a density d_i. The effective sampling mass for a segment is its duration multiplied by its density. Frames are then allocated across segments by largest-remainder rounding so that the resulting input contains the recognizer's required number of frames.

For the positive condition, called Importance-Led FF, the density rule is

$$d_i = \begin{cases} 1, & \ell_i = \text{VLM-important}, \\ \alpha, & \ell_i = \text{VLM-designated filler}. \end{cases}$$

For the negative control, called Anti-Importance FF, the rule is inverted:

$$d_i = \begin{cases} \alpha, & \ell_i = \text{VLM-important}, \\ 1, & \ell_i = \text{VLM-designated filler}. \end{cases}$$

Both arms share the same variable-density machinery; only the importance ordering is inverted.

The production α grid contains interior values only:

$$\alpha \in \{0.05, 0.10, 0.20, 0.25, 0.30, 0.40, 0.50, 0.60, 0.70, 0.75, 0.80, 0.90\}.$$

I report the endpoint conditions separately (Section 3.5.4). Sampling is deterministic for a given sidecar, condition, and α, except for protocols explicitly defined to use randomness, such as random insertion/deletion orderings and temporal-order shuffling.

3.5.3 Positive condition and matched negative control

The matched negative control is necessary because variable-density sampling can change recognizer behavior even when the importance signal is uninformative. For example, changing the temporal density may alter which phases of a video are represented, or it may change the effective temporal stride of the recognizer input. A positive condition alone could not distinguish VLM-signal gains from sampling-mechanism gains.

The Anti-Importance FF condition controls for this confound by using the same density floor, frame budget, and frame allocation rule as Importance-Led FF, but with the importance ordering inverted. If gains were caused purely by variable-density sampling, the positive and negative-control curves should behave similarly. A gap between the two curves at matched α indicates that the VLM ordering carries task-relevant signal. Figure 9 illustrates both arms on a real eight-segment SSv2 example at $\alpha = 0.25$ with a sixteen-frame budget: the positive arm concentrates frames on VLM-important segments, while the inverted control concentrates the same budget on VLM-designated filler.

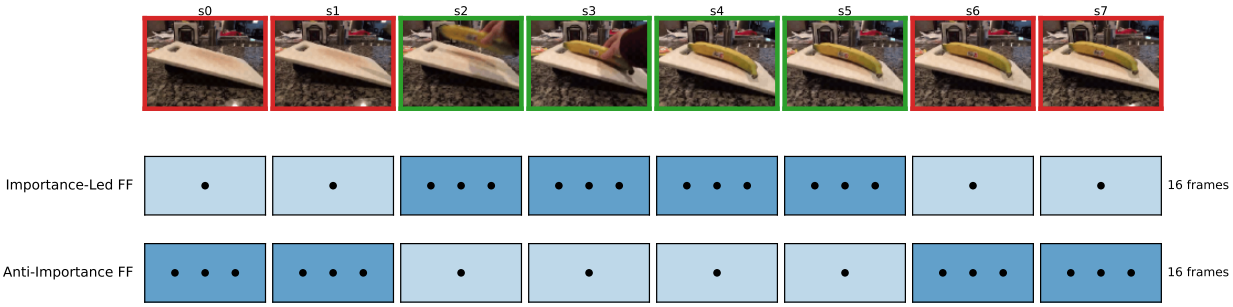

Figure 9: Variable-density sampling at $\alpha = 0.25$ on a real eight-segment SSv2 example with a sixteen-frame budget (the same video as Figure 4, here with its production direct-scoring labels, which keep s_2 through s_5). Cell shading shows the per-segment density d_i and dots show the frames allocated by largest-remainder rounding. Importance-Led FF concentrates frames on VLM-important segments; the inverted control concentrates the same budget on VLM-designated filler.

3.5.4 Endpoint conditions and reporting conventions

Although the production α grid covers only interior values, I report two endpoint conventions to make the α axis interpretable. The $\alpha = 1$ reference is the `full` condition, which samples uniformly

from the full video. This is not literally an α-sweep grid point; it is the shared uniform-reference condition for both the positive and negative arms.

Hard-cut conditions represent the $\alpha = 0$ endpoint. The positive endpoint, `vlm-selected` or Keep-Important, keeps the VLM-important segments and drops the filler; the negative endpoint, `lowest-evidence` or Keep-Filler, inverts that rule. Both use a cut-and-concatenate protocol followed by resampling, rather than a density rule with $d_i = 0$ over the full timeline, and I report them as endpoint-equivalent conditions because they express the limiting positive and inverted selection rules.

A separate continuous-score endpoint appears in the threshold and budget sweeps. For example, `vlm-score-threshold-t0` keeps every segment with nonzero retained weight and therefore has different semantics from the binary hard-cut endpoint. I keep these endpoint conventions separate throughout the Results section.

3.6 Evaluation setup

3.6.1 Evaluation pools

All headline recognition experiments use pinned evaluation pools so that comparisons are paired by video. SSv2 and K400 use stratified evaluation CSVs sampled with seed 42 from precheck-passed records. Diving-48 uses the full validation split after precheck filtering. Insertion/deletion curves use separate 500-video stratified subsets (Table 5).

Table 5: Evaluation pools used for headline and auxiliary experiments. All pools are restricted to precheck-passed records.

Dataset	CSV	Videos	Classes	Sampling rule
SSv2	`eval_2k_stratified.csv`	2,080	174	Seed 42, per-class cap 12
K400	`eval_2k_stratified.csv`	2,000	400	Seed 42, per-class cap 5
Diving-48	`eval_full.csv`	1,970	47	Full validation split
I/D subsets	`id_500_stratified.csv`	500 each	varies	Stratified from evaluation pools

All sampling conditions within a dataset are evaluated on the same pinned video IDs when paired statistics are reported. This allows paired bootstrap intervals and paired condition-versus-reference tests to be computed on matched examples.

3.6.2 Recognizers

The recognizers (Table 6) are dataset-specific checkpoints, but they are frozen during all MM evaluations: no gradients are taken and no model weights are updated. Only the temporal input selection changes. This design makes the temporal sampler the experimental intervention.

SSv2 also has a step=4 protocol check corresponding to the paper-style clip span. Under that setting, the variable-density pool can saturate on short videos, so step=2 is used as the primary SSv2 α-sweep protocol. Some auxiliary diagnostics use HuggingFace V-JEPA 2 ports rather than the official paper checkpoints; those count as diagnostic runs and do not define the headline recognizer protocol.

Table 6: Headline recognizers. All are frozen during MM evaluation.

Dataset	Recognizer	Checkpoint	Input protocol
SSv2	V-JEPA 2 official	`ssv2-vitl-16x2x3.pt`	16 frames, 2×3 views, step=2 primary
K400	VideoMAE-L	`MCG-NJU/videomae-large-finetuned-kinetics`	16-frame single clip
Diving-48	V-JEPA 2 official	`diving48-vitl-256.pt`	32 frames, 4×3 views, step=2

3.6.3 Statistical testing

All headline contrasts are paired by video whenever possible. For the headline contrast family, I compute paired bootstrap confidence intervals with $B = 10{,}000$ resamples using the percentile method. Auxiliary paired-stat reports use $B = 1{,}000$ resamples unless otherwise stated. Displayed confidence intervals are raw 95% intervals.

For paired condition-versus-reference tests, I use McNemar's test with Yates continuity correction. For the family of $k = 8$ primary headline contrasts, I apply Bonferroni correction when interpreting statistical significance. Holm step-down correction is reported only as an auxiliary reference in appendix analyses. If any condition fails to produce a valid sampled input for a video, that video is excluded from the matched subset for the corresponding paired contrast.

3.7 Auxiliary validation protocols

3.7.1 Threshold and budget sweeps

The α-sweep uses the production binary labels because they define the cleanest positive/inverted-control comparison. I separately evaluate the continuous weights with threshold and budget sweeps.

The score-threshold sweep keeps segments with $w_i \geq T/100$, for $T \in \{0, 10, 20, \ldots, 100\}$. The threshold is inclusive. If no segment satisfies a high threshold, the selector falls back to keeping the single highest-weight segment so that the recognizer receives a valid input.

The budget sweep ranks segments by continuous weight and keeps the highest-weight segments until the retained duration reaches the target budget fraction. I evaluate $f \in \{10, 20, \ldots, 90\}$. Both sweep families use the same cut-and-concatenate kept-set protocol followed by recognizer-specific resampling.

3.7.2 Insertion/deletion curves

Insertion/deletion curves evaluate whether the VLM weight ordering is meaningful across a range of retained fractions. I rank segments by five orderings: VLM weight descending, VLM weight ascending (anti-VLM), deterministic random order, chronological order, and optical-flow motion magnitude. For each ordering, I sweep the retained fraction over $\{10, 20, \ldots, 90\}$, add common endpoints, and compute the trapezoidal area under the resulting top-1 and top-5 curves.

The primary comparison is each ordering's AUC relative to the random ordering on the same 500-video stratified pool. This provides a recognizer-level test of whether the VLM ordering carries signal beyond content-agnostic or non-VLM baselines.

3.7.3 Temporal-order shuffling

The temporal-order shuffling probe tests whether a condition relies on the chronological order of its retained segments. For a given kept-set condition, I evaluate the recognizer twice: once with retained segments in chronological order and once with the retained segment order randomly permuted. The segment contents are not altered; only their order is changed.

I report shuffle damage as the aligned top-1 difference between the unshuffled and shuffled inputs. I also compare differential damage between `vlm-selected` and uniform kept-set inputs. This probe does not measure general importance; it measures temporal-order sensitivity under the recognizer and input protocol.

3.7.4 Cross-oracle validation

The primary corpus uses Qwen3-VL-32B, so I run a separate cross-oracle validation study to bound oracle dependence. The final four-oracle study uses Qwen3-VL-32B, InternVL3-38B, Gemini 3.1 Pro, and GPT-5.5 via Azure ChatAPI. All four oracles score the same shared videos with the generic direct-scoring prompt. The joint-precheck-pass subset keeps only videos for which all four oracles produced a parseable sidecar and passed precheck, yielding $n = 369$ videos.

I compute segment-level agreement metrics, including Spearman correlation on aligned weight vectors and Jaccard / Set-F1 overlap on binary kept sets. I also evaluate downstream recognition trends under labels produced by different oracles. The cross-oracle study is not a human-ground-truth validation; it is a reliability and robustness check for VLM-derived pseudo-labels.

3.7.5 Oracle-call accounting

Direct scoring uses about one finalized oracle response per source-video annotation attempt. That response contains precheck metadata, segment scores, and the oracle-designated kept set. Greedy removal is iterative: it repeatedly removes candidate segments and asks whether the action remains recognizable. Its worst-case call count grows quadratically with the number of segments, although the observed pilot average is lower because the procedure terminates early and the video shortens after removals.

For corpus-scale accounting, I report oracle-call counts rather than dollar cost. Direct scoring required about 0.54 million oracle calls at MM scale; greedy removal would have required an estimated 13.4 million. The Results section evaluates whether this roughly 25× reduction sacrifices downstream signal.

4 Results

4.1 Overview: importance is informative but dataset-dependent

Across all three datasets, the matched positive and negative arms separate decisively at the hard-cut endpoint: keeping VLM-important segments beats keeping VLM-designated filler by +29.95 pp on Diving-48, +5.25 pp on K400, and +9.04 pp on SSv2. The importance signal is therefore informative everywhere. Its downstream usefulness, however, depends on dataset structure. Table 7 reports each dataset's best VLM-guided operating point against `full`: high-keep selection preserves or slightly improves recognition on Diving-48, aggressive selection improves recognition on K400, and no tested operating point recovers full-video performance on SSv2. The subsections below analyze the matched α-sweeps, the hard-cut endpoints, and the continuous-score threshold and budget

sweeps behind these operating points. Table 8 collects the paired-bootstrap confidence intervals for the eight primary contrasts; the statistical conventions are those of Section 3.6.3.

Table 7: Headline finding: VLM-derived temporal importance is informative across all three datasets, but its downstream usefulness depends on dataset structure. Each row reports the dataset's best VLM-guided operating point; the subsequent subsections analyze matched α-sweeps, hard-cut endpoints, and continuous-score threshold and budget sweeps in full.

Dataset	Best VLM-guided result vs `full`	Interpretation
Diving-48	+0.76 pp ($T = 0$)	High-keep selection preserves or slightly improves recognition
K400	+2.05 pp ($T = 70$)	Aggressive selection improves recognition
SSv2	−2.12 pp ($f = 70\%$)	Best selection remains below full-video performance

Table 8: Paired-bootstrap confidence intervals on the eight primary contrasts from both sweeps. Δ is mean(condition) minus mean(`full`) over paired sidecars; CIs are percentile intervals from $B = 10{,}000$ paired bootstrap resamples. "$*$" marks intervals that exclude 0 at the raw 95% level.

Dataset	Contrast	n	Δ (pp)	95% CI
D48	α-sweep: Imp-Led $\alpha = 0.70$ vs `full`	1970	+0.56	$[-0.10, +1.27]$
SSv2	α-sweep: `vlm-selected` vs `full`	2080	-3.32^*	$[-4.90, -1.73]$
K400	α-sweep: `vlm-selected` vs `full`	2000	$+1.45^*$	$[+0.35, +2.55]$
D48	$\alpha = 0$: $T = 0$ vs `full`	1970	$+0.76^*$	$[+0.05, +1.47]$
SSv2	$\alpha = 0$: $T = 50$ vs `full`	2080	-2.40^*	$[-3.75, -1.01]$
K400	$\alpha = 0$: $T = 70$ vs `full`	2000	$+2.05^*$	$[+1.00, +3.10]$
D48	$\alpha = 0$: $f = 90\%$ vs `full`	1970	+0.05	$[-0.66, +0.76]$
K400	$\alpha = 0$: $f = 30\%$ vs `full`	2000	$+1.80^*$	$[+0.65, +2.90]$

Every SSv2 and K400 contrast is significant at the raw 95% level, and all of them survive Bonferroni correction over the $k = 8$ family (family-wise $\alpha = 0.05/8 \approx 0.006$): the SSv2 kept-set deficit (-3.32 and -2.40 pp) and the K400 selection gains ($+1.45$, $+2.05$, and $+1.80$ pp) are robust. The Diving-48 contrasts are not: the $\alpha = 0.70$ peak ($[-0.10, +1.27]$) and the $T = 0$ win ($[+0.05, +1.47]$) do not survive correction, and the $f = 90\%$ contrast was not significant to begin with. The Diving-48 headline should therefore be read as selection matching `full`, not significantly beating it.

4.2 Alpha-sweep results across datasets

4.2.1 Diving-48: fast-forward works when evidence is temporally concentrated

Diving-48 is the dataset where the variable-density mechanism works as intended. Table 9 and Figure 10 show the full sweep: the two arms agree near $\alpha = 1$ and fan out as $\alpha \to 0$, which is the signature of the importance signal rather than the sampling mechanism driving the gap. Importance-Led FF is robust across the axis, staying within ± 0.6 pp of `full` for all $\alpha \in [0.5, 1.0]$ and peaking slightly above `full` at $\alpha = 0.7$ (88.38%, +0.56 pp; the CI $[-0.10, +1.27]$ overlaps zero, so the peak is suggestive rather than significant). The inverted control collapses monotonically, from

88.38% at $\alpha = 0.9$ to 52.13% at the hard cut, giving a +29.95 pp endpoint gap that variable-density sampling alone cannot explain.

The mechanism is consistent with the dataset's structure. Diving-48 classes are temporally concentrated athletic actions whose discriminative content is the dive itself, surrounded by setup and water-splash frames that carry little class information. The oracle tags the dive as important and the surroundings as filler, so Importance-Led FF concentrates compute exactly where the evidence is, and the inverted control concentrates it exactly where the evidence is not.

Table 9: Diving-48 α-sweep paired against the inverted control (V-JEPA 2 paper checkpoint, 4×3 multi-clip, $n = 1970$). The gap column (Imp-Led minus Anti-Imp top-1) is small at high α, where both arms converge to `full`, and large at low α, where the density rule is fully expressed.

	Importance-Led FF		Anti-Importance FF		
α	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)	Gap top-1 (pp)
1.00	`full`: 87.82 / 99.09 (shared endpoint)				0.00
0.90	88.27	98.93	88.38	98.98	−0.11
0.80	88.27	98.98	88.27	99.04	0.00
0.70	**88.38**	98.98	88.17	99.04	0.21
0.60	87.82	98.93	86.35	98.83	1.47
0.50	88.07	99.09	84.01	98.48	4.06
0.40	86.90	98.83	79.19	97.77	7.71
0.30	85.08	98.22	66.80	94.16	18.28
0.20	85.18	97.97	66.55	94.26	18.63
0.10	83.60	97.77	58.32	88.53	25.28
$\alpha = 0$ *hard-cut endpoints (Keep-Important vs Keep-Filler)*					
0.00	82.08	97.41	52.13	83.76	**+29.95**

4.2.2 K400: hard cuts improve action-focused sampling

On K400 the interior of the α axis is uninformative and the hard cut is where the action is. Table 10 and Figure 11 show that every interior gap satisfies $|\text{gap}| \leq 0.5$ pp: reducing density on a small fraction of low-importance frames neither helps nor hurts. At the hard-cut endpoint, however, Keep-Important is the best K400 condition outright at 85.40%, beating `full` by +1.45 pp ([+0.35, +2.55], significant after Bonferroni) and Keep-Filler by +5.25 pp.

The pattern again matches the dataset's structure. K400 classes are broad activities that are visible through most of the clip, so a uniform sample already covers the action and there is little filler for the density rule to skip, which produces the flat interior. The hard cut does something different: it discards setup and camera-pan segments entirely and concentrates the sixteen-frame budget on footage where the oracle confidently saw the action, which is why `vlm-selected` beats `full`. The Keep-Filler control fails for the symmetric reason: it keeps the distractors and discards the action.

4.2.3 SSv2: importance is informative but full context still wins

SSv2 separates the importance signal from the sampling mechanism most sharply. Table 11 and Figure 12 report the primary `frame_step=2` protocol (the step=4 paper protocol saturates the

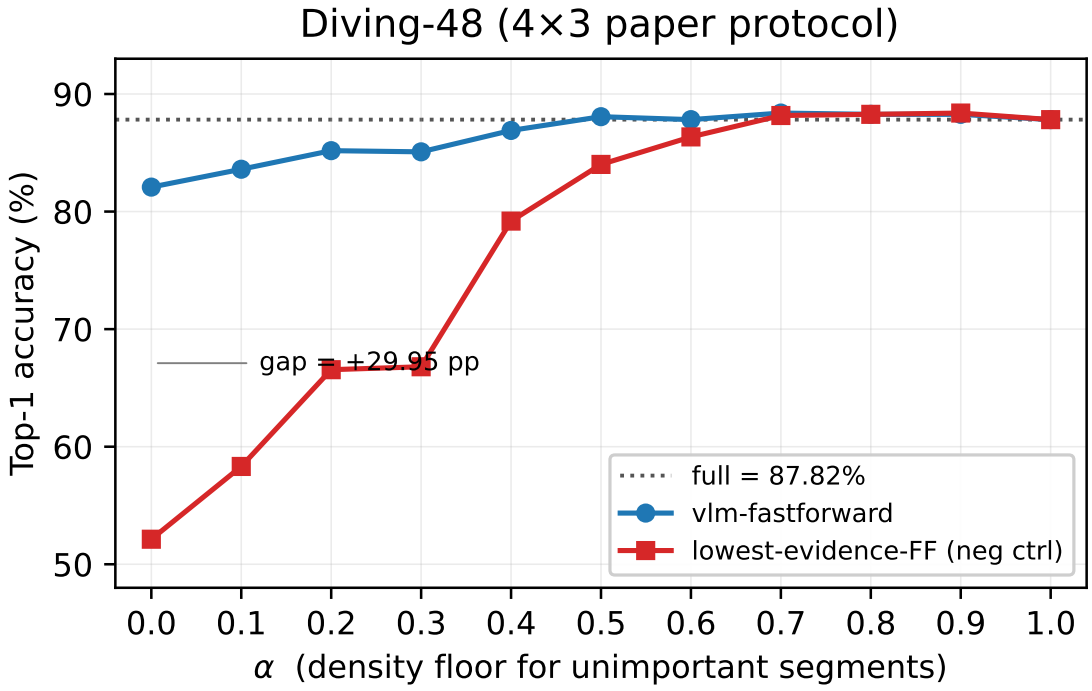

Figure 10: Diving-48 top-1 vs α. The two arms agree near $\alpha = 1$ and fan out as $\alpha \to 0$. Importance-Led FF peaks at $\alpha = 0.7$ slightly above `full`; the inverted control collapses by about 30 pp at the hard-cut endpoint.

Table 10: K400 α-sweep paired against the inverted control (VideoMAE-L, single 16-frame clip, $n = 2000$; the paired aligned denominator after dropping 21 selector failures is $n_{\text{paired}} = 1979$). Interior gaps are negligible; only the hard-cut endpoint separates the arms.

	Importance-Led FF		Anti-Importance FF		
α	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)	Gap top-1 (pp)
1.00	`full`: 83.95 / 96.90 (shared endpoint)				0.00
0.90	83.45	96.35	83.30	95.95	0.15
0.80	83.30	96.45	83.45	96.20	−0.15
0.70	83.25	96.50	83.35	96.20	−0.10
0.60	83.30	96.55	83.55	96.25	−0.25
0.50	83.25	97.05	83.70	96.30	−0.45
0.40	83.35	97.00	83.80	96.30	−0.45
0.30	83.50	97.10	83.75	96.35	−0.25
0.20	83.90	96.85	83.55	96.75	0.35
0.10	83.95	96.85	83.45	96.70	0.50
$\alpha = 0$ *hard-cut endpoints*					
0.00	**85.40**	**97.55**	80.15	94.15	**+5.25**

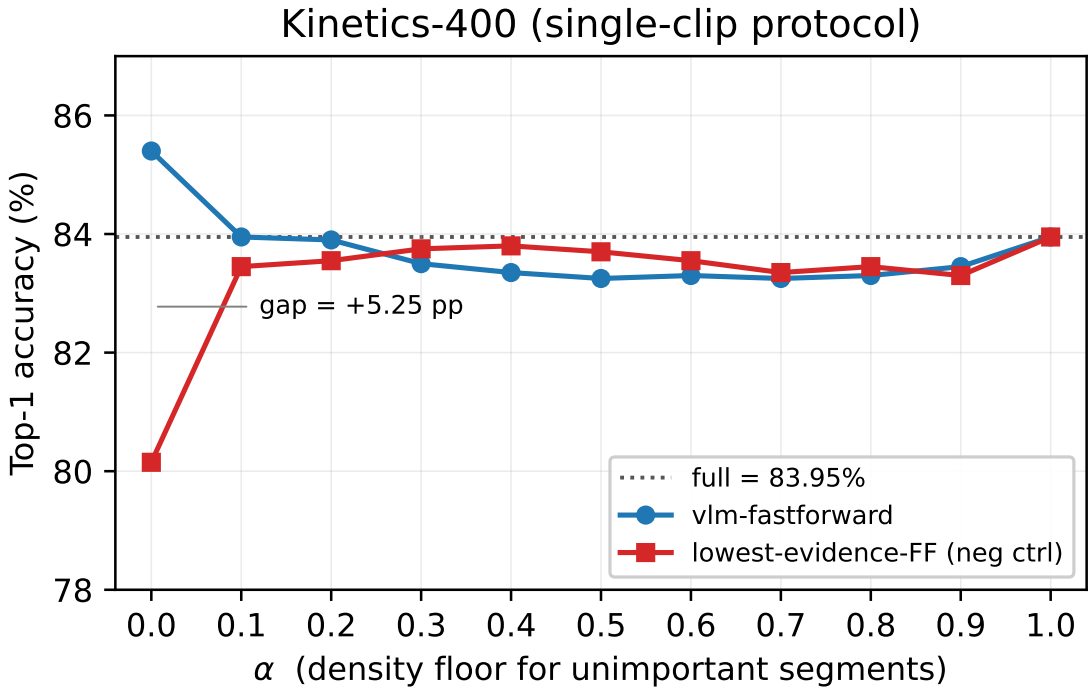

Figure 11: K400 top-1 vs α. Interior curves are flat and overlap; the hard-cut endpoint is where the arms separate, with Keep-Important (`vlm-selected`) the best K400 condition.

variable-density pool on short SSv2 videos and is reported only as an endpoint sanity check; see Appendix C.6). Three observations follow. First, Importance-Led FF never exceeds `full` at any α; its best interior point is 65.53% at $\alpha \in \{0.1, 0.2, 0.3\}$, still 0.24 pp below `full`. Second, the interior gaps are small and run in the *wrong* direction: the arms tie at $\alpha = 0.9$, and at every interior α below it the inverted control is marginally ahead (gaps between -0.62 and -1.45 pp), and at $\alpha \in \{0.1, 0.2, 0.3\}$ it even edges the step=2 `full` baseline (66.68 versus 65.77), a protocol artifact rather than a control win: the step=2 reference sits about 4 pp below its step=4 counterpart (69.76), and interior variable-density inputs alter the effective temporal stride. Third, the hard-cut endpoint nevertheless gives a clean positive gap of +9.04 pp (and +11.77 pp under the step=4 sanity check, where `full` reaches 69.76%). The importance signal carries real information on SSv2; the fast-forward mechanism simply does not surface it.

The misalignment explains the reversed interior. SSv2 classes are cause-effect manipulations discriminated by initial versus final object state, that is, by frames near the action boundaries. The oracle tags the middle of the motion as important and the near-still boundary frames as filler, so Importance-Led FF sends compute toward mid-action frames while the inverted control happens to sample the boundary frames the recognizer wants, landing marginally ahead at matched α. The hard cut removes this confound: Keep-Important still contains the boundary content at the edges of each kept segment, while Keep-Filler retains only brief filler segments with no temporal continuity, and the endpoint gap turns cleanly positive.

Label preservation under closed-set classification. As an independent check that does not involve the recognizer, the same oracle re-classified the SSv2 test set (n = 25,647) over the full

Table 11: SSv2 α-sweep paired against the inverted control (V-JEPA 2 paper checkpoint, 2×3 multi-clip, `frame_step=2`, $n = 2080$). Interior gaps are small and negative; only the hard-cut endpoint shows the positive direction.

	Importance-Led FF		Anti-Importance FF		
α	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)	Gap top-1 (pp)
1.00	`full`: 65.77 / 92.98 (shared endpoint)				0.00
0.90	59.86	88.94	59.86	88.94	0.00
0.80	61.15	89.81	62.60	90.82	−1.45
0.70	61.44	89.95	62.50	90.82	−1.06
0.60	63.89	91.83	64.62	90.96	−0.73
0.50	63.89	91.83	64.62	90.96	−0.73
0.40	64.04	91.97	64.66	90.82	−0.62
0.30	65.53	92.60	66.68	91.92	−1.15
0.20	65.53	92.60	66.68	91.92	−1.15
0.10	65.53	92.60	66.68	91.92	−1.15
$\alpha = 0$ *hard-cut endpoints*					
0.00	62.45	91.11	53.41	81.20	**+9.04**

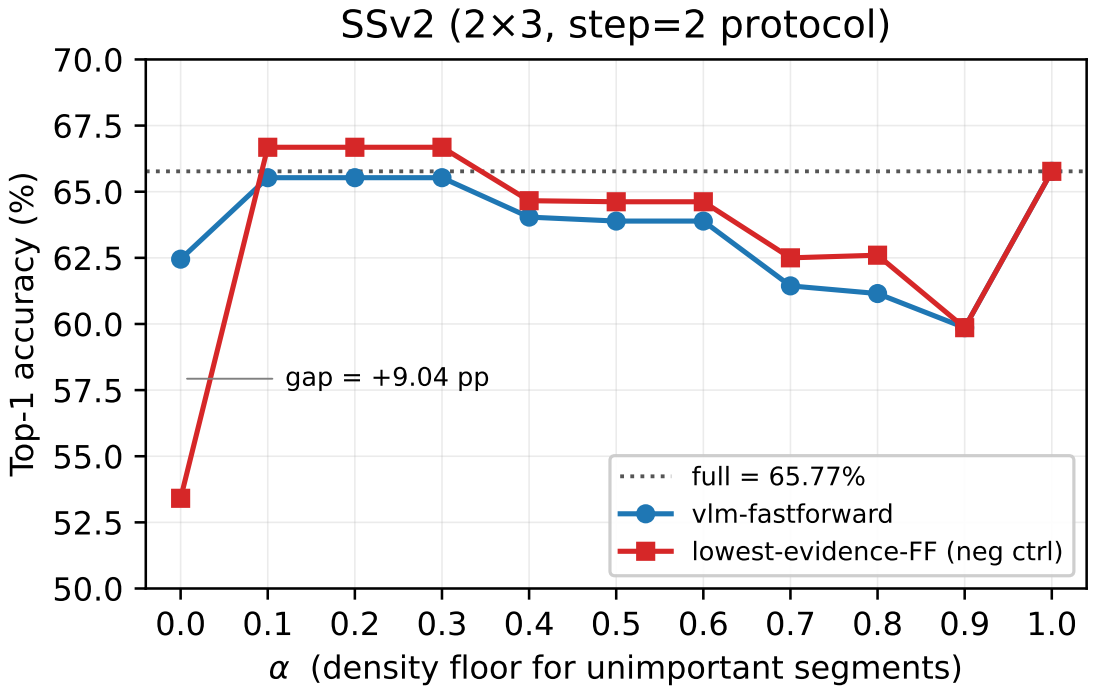

Figure 12: SSv2 top-1 vs α at `frame_step=2`. Interior curves overlap within about 1 pp, with the inverted control marginally above; only the hard-cut endpoint shows the canonical positive direction.

174-template closed set, once on the full video and once on the MSS-cut. Top-1 is 17.18% for full versus 14.70% for the cut (−2.48 pp; top-5 34.01% versus 30.35%), with 62.86% paired-prediction

agreement: the cut preserves about 85% of full-video top-1 from the oracle's own perspective. The aggregate loss is not uniform. Restricting to the 172 templates with at least 10 videos, the median per-template delta is exactly 0 pp and 67% of templates sit within ±2 pp of full; the loss concentrates in a minority of boundary-state-discriminated templates, while motion-discriminated templates can gain (Appendix E.3 gives the per-template breakdown).

4.3 Continuous scores outperform oracle-designated kept sets

The oracle returns both a continuous weight and a categorical kept set, and the two are not equally useful. Sweeping a score threshold T (keep $w_i \geq T/100$) and a duration budget f (keep the top-f% of duration by weight) over the retained continuous weights shows that the oracle-designated kept set is dominated by other operating points on all three datasets (Table 12, Figure 13; full decile tables in Appendix E.5).

Three different shapes emerge, one per dataset. On Diving-48 the threshold axis is endpoint-optimal: $T = 0$, which keeps every segment with nonzero weight, reaches 88.58% (+0.76 pp over `full` and +6.50 pp over the kept set). Because $T = 0$ and `full` are both keep-everything conditions under different inference protocols, this gap is best read as the kept-set protocol matching the uniform protocol, not as a selection gain (cf. Section 3.5.4). On SSv2 both axes have interior optima ($T = 50$ at 63.37%, $f = 70$ at 63.65%): some selection beats none, aggressive selection is worse, and the whole sweep stays below `full`. On K400 both axes peak above `full` in the aggressive regime ($T = 70$ at 86.00%, $f = 30$ at 85.75%), adding +0.60 pp over the already-winning categorical cut.

Two further observations hold everywhere. First, the categorical kept set is never the best operating point: the oracle is strictly more selective than its own scoring rule prescribes, omitting from the kept set roughly 12% of segments that score at or above the stated 50/100 inclusion bar (measured on Diving-48), and the recognizer prefers those segments back. Second, the lowest-weight segments are not redundant: forcing $T = 100$ or $f = 10$ collapses accuracy below even the Keep-Filler control on Diving-48 and SSv2, so discarding the low-weight tail entirely is more destructive than inverting the importance signal.

Table 12: Threshold and budget sweep optima per dataset. Three different shapes emerge: endpoint-optimal on Diving-48, interior-optimal on SSv2, and aggressive-optimal on K400.

Dataset	`full`	`vlm-selected`	Best T	Best f	Best vs `full`
Diving-48	87.82	82.08	$T = 0$: 88.58	$f = 90$: 87.87	+0.76
SSv2	65.77	62.45	$T = 50$: 63.37	$f = 70$: 63.65	−2.12
K400	83.95	85.40	$T = 70$: 86.00	$f = 30$: 85.75	+2.05

4.4 Mechanism probes

4.4.1 Insertion/deletion curves validate the importance ordering

The insertion/deletion analysis tests whether the importance *ordering* is meaningful across the whole retained-fraction axis rather than at a single operating point, using the five orderings of Section 3.7.2 on the $n = 500$ stratified subsets (Figure 14, Tables 13 and 14). Three findings. First, inverting the importance signal hurts everywhere: `anti-vlm` is significantly below `random` on all three datasets (−0.098 AUC on SSv2, −0.032 on K400, −0.070 on D48, all $p < 10^{-3}$). Second, `vlm` beats `random` significantly on top-5 on all three datasets, while on top-1 it is significant only

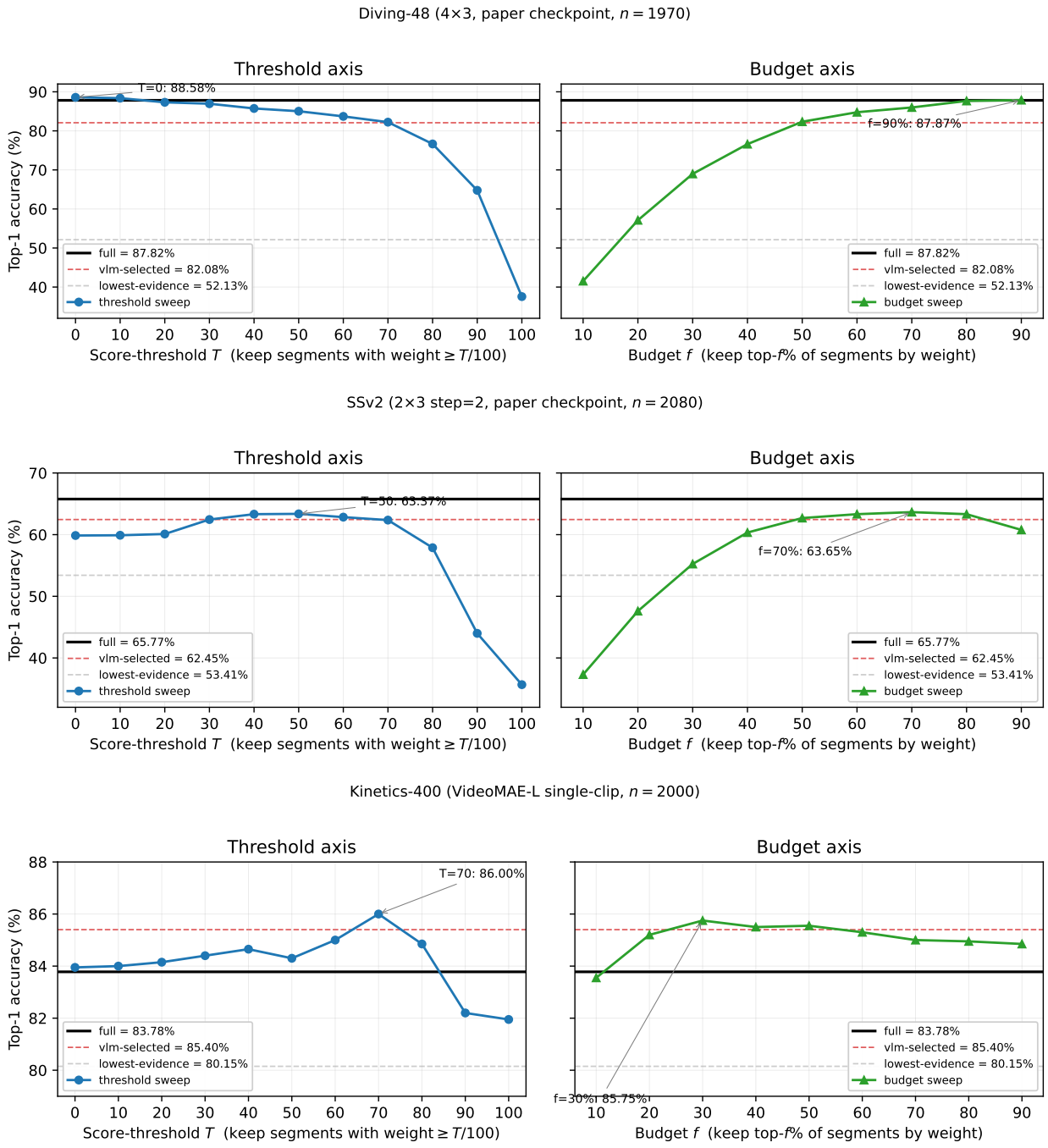

Figure 13: Score-threshold (left) and budget (right) sweeps per dataset (top to bottom: Diving-48, SSv2, K400). Reference lines mark `full`, `vlm-selected`, and `lowest-evidence`. The three datasets produce three different shapes: endpoint-optimal, interior-optimal, and aggressive-optimal.

on Diving-48 (+0.054); on SSv2 and K400 the ordering helps reach the right class neighborhood more than it nails the exact label. Third, the content-agnostic baselines behave dataset-specifically: `temporal` is below `random` on SSv2 and K400 but above it on Diving-48, whose dives follow a canonical phase sequence, and `motion` is competitive throughout, edging `vlm` on SSv2. These runs

use the HF-port recognizers, so absolute values are not comparable to the paper-checkpoint tables; the ordering-minus-random comparison at fixed recognizer is the meaningful quantity.

Table 13: Mean trapezoidal AUC of the insertion/deletion curve, top-1 and top-5 ($n = 500$ paired per dataset). Bold marks the highest mean per (dataset, metric).

	Metric	vlm	motion	random	temporal	anti-vlm
SSv2 (V-JEPA 2 2×3)	Top-1	0.5761	**0.5927**	0.5763	0.4937	0.4787
	Top-5	0.8326	**0.8396**	0.8176	0.7368	0.7242
K400 (VideoMAE-L)	Top-1	**0.8107**	0.7947	0.8029	0.7937	0.7711
	Top-5	**0.9239**	0.9083	0.9109	0.9077	0.8831
Diving-48 (V-JEPA 2 HF-port)	Top-1	**0.3480**	0.3402	0.2942	0.3132	0.2244
	Top-5	**0.7271**	0.6951	0.6595	0.6583	0.5295

Table 14: ΔAUC versus the `random` ordering, paired bootstrap CIs ($B = 10{,}000$, $n = 500$ per dataset). "$*$" marks intervals that exclude 0 at 95%.

Dataset	Ordering	Top-1 ΔAUC	95% CI	p
SSv2	vlm	-0.0002	$[-0.014, +0.014]$	0.985
SSv2	motion	$+0.0164^*$	$[+0.003, +0.030]$	0.017
SSv2	temporal	-0.0826^*	$[-0.100, -0.065]$	$< 10^{-3}$
SSv2	anti-vlm	-0.0976^*	$[-0.114, -0.082]$	$< 10^{-3}$
K400	vlm	$+0.0078$	$[-0.001, +0.017]$	0.077
K400	motion	-0.0082	$[-0.017, +0.0002]$	0.059
K400	temporal	-0.0092^*	$[-0.018, -0.0004]$	0.037
K400	anti-vlm	-0.0318^*	$[-0.043, -0.021]$	$< 10^{-3}$
D48	vlm	$+0.0538^*$	$[+0.037, +0.071]$	$< 10^{-3}$
D48	motion	$+0.0460^*$	$[+0.032, +0.060]$	$< 10^{-3}$
D48	temporal	$+0.0190^*$	$[+0.003, +0.034]$	0.011
D48	anti-vlm	-0.0698^*	$[-0.085, -0.055]$	$< 10^{-3}$
Top-5 ΔAUC, vlm vs random				
SSv2	vlm	$+0.0150^*$	$[+0.004, +0.026]$	0.006
K400	vlm	$+0.0130^*$	$[+0.007, +0.020]$	$< 10^{-3}$
D48	vlm	$+0.0676^*$	$[+0.053, +0.083]$	$< 10^{-3}$

4.4.2 Temporal-order reliance is dataset-specific

Shuffling the temporal order of the kept segments probes what kind of signal each condition relies on (Section 3.7.3; Table 15). The pattern is dataset-specific rather than selection-specific. On SSv2 both conditions are strongly order-reliant and `uniform` more so than `vlm-selected` (differential +2.21 pp, CI $[-2.40, +4.81]$, direction consistent with the boundary-state story, though the interval includes zero). On K400 neither condition is order-reliant (both deltas under 1 pp, n.s.), consistent with activity-throughout-clip semantics. On Diving-48 the sign reverses significantly:

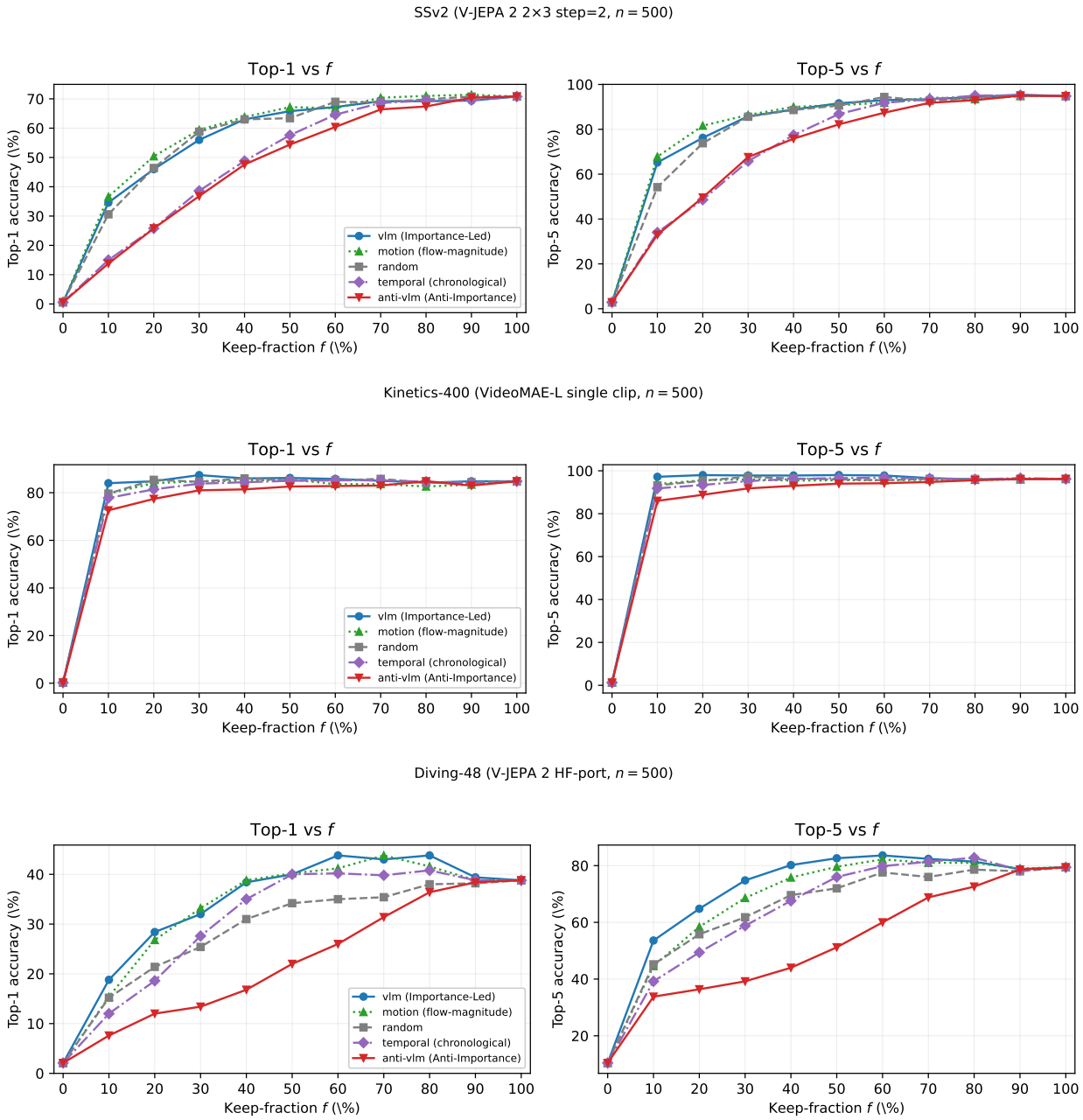

Figure 14: Insertion/deletion curves on the $n = 500$ stratified subsets (top to bottom: SSv2, K400, Diving-48; left top-1, right top-5). `anti-vlm` sits below `random` everywhere; Diving-48 shows the cleanest `vlm` separation; `motion` is a strong content-aware baseline.

`vlm-selected` is *more* order-reliant than `uniform` (differential -5.59 pp, $[-8.09, -3.15]$, paired on the 1,681 videos where both conditions evaluate), because the kept set concentrates on the dive whose counted rotations are destroyed by shuffling, while uniform keeps approach and entry frames whose information is less order-encoded.

Table 15: Shuffle damage on top-1 accuracy. $\Delta_{\text{shuf}} = \text{top-1}_{\text{unshuffled}} - \text{top-1}_{\text{shuffled}}$; higher means more temporal-order reliant. CIs are 95% paired bootstrap; p is McNemar's test. The D48 `uniform` row covers the 1,681 videos where the uniform selector produces a valid pair (Section 3.6.3).

Dataset	Condition	n	unshuf	shuf	Δ_{shuf} (CI)	p
SSv2	`vlm-selected`	2080	62.31	47.45	14.86 $[12.74, 17.07]$	6×10^{-41}
SSv2	`uniform`	2080	65.96	48.89	17.07 $[15.05, 19.13]$	5×10^{-56}
K400	`vlm-selected`	2000	85.40	84.95	0.45 $[-0.45, 1.40]$	0.41
K400	`uniform`	2000	85.15	84.55	0.60 $[-0.30, 1.50]$	0.22
D48	`vlm-selected`	1970	41.32	35.03	6.29 $[4.67, 7.92]$	2×10^{-14}
D48	`uniform`	1681	36.11	34.32	1.78 $[0.00, 3.57]$	0.056

4.5 Cross-oracle validation bounds pseudo-label reliability

The corpus has a single primary oracle, so a four-oracle study bounds how oracle-specific the labels are. On the $n = 369$ joint-precheck-pass sample (Section 3.7.4; Appendix F), the three closed-weights frontier models (Qwen3-VL-32B, Gemini 3.1 Pro, GPT-5.5) cluster at pairwise Spearman 0.49–0.61 on per-segment importance, while InternVL3-38B sits apart at 0.35–0.37 against all three. Kept-set agreement is moderate everywhere (Set-F1 0.55–0.74): the signal is real and reproducible, but a sizeable fraction of segment-level decisions are genuinely model-dependent, which bounds the reliability of any single-oracle corpus (Table 25).

The more load-bearing result is downstream: substituting any of the other oracles as the label source preserves the qualitative recognition ordering. In the per-oracle recognition evaluation (Table 26), `vlm-selected` beats `uniform` and `lowest-evidence` on Diving-48 and K400 for every oracle, and Gemini, the strongest source on all three datasets, reproduces the same qualitative ordering as the production Qwen labels. Moderate segment-level disagreement therefore washes out at the level of the recognition conclusions this thesis draws.

4.6 Direct scoring reduces oracle calls while matching greedy within its noise floor

The conceptual target of Section 3.3 is greedy counterfactual removal; the production corpus uses direct scoring. Three pilot results on the same 101 SSv2 videos justify the substitution. First, greedy mode is itself unstable: four independent greedy runs with identical prompt, model, and parameters agree pairwise at only mean Jaccard 0.146 on kept sets. The apparently stable rank signal (mean Spearman 0.711) is an artifact of tie handling: greedy weights are coarse and heavily tied, and under standard tie-corrected Spearman the agreement drops to about 0.17 (Appendix A.2), so the ordering is no more reliable than the kept set. Second, the pooled greedy-to-direct kept-set agreement is 0.144, the same magnitude as greedy's own noise floor: switching modes moves the kept set no more than re-running greedy would, while reducing measured oracle calls from roughly 25 per video to 1. At production scale this is the difference between the released corpus ($\sim$ 0.54M calls) and an estimated $\sim$ 13.4M calls for greedy, which at fixed budget would have supported only about 21K labeled videos.

Third, the comparison conflates extraction mode with prompt template, and a controlled decomposition shows the mode is the load-bearing variable (Table 16). Two direct-scoring runs that differ only in prompt agree at Jaccard 0.827 and Spearman 0.861; either direct run agrees with greedy at only 0.21 Jaccard with near-zero weight correlation. Changing the prompt within direct

mode moves the kept set far less than changing the mode. One residual difference matters for the corpus: direct mode keeps about 3.8× more segments per video than greedy (4.42 versus 1.17 on the pilot), partly a prompt effect, which yields a richer per-video positive set for the kept-set protocol. Full pilot tables and the prompt texts are in Appendix A.

Table 16: Mode-versus-prompt decomposition on the same SSv2 pilot videos. Within direct mode (last row), changing the prompt moves the kept set far less than the across-mode comparison: the greedy-to-direct gap is a mode effect, not a prompt effect.

Pairing	n	Mean Jaccard	Mean Spearman
Greedy ↔ direct (generic prompt)	~ 95	0.208	−0.028
Greedy ↔ direct (SSv2 prompt)	~ 95	0.221	−0.011
Direct (generic) ↔ direct (SSv2)	~ 95	0.827	+0.861

4.7 Summary of empirical findings

The empirical picture is consistent across every probe. The importance signal is informative on all three datasets: the inverted control loses decisively at the hard cut, anti-importance orderings lose to random everywhere, and four oracles reproduce the downstream ordering. Its usefulness is conditional on dataset structure: selection matches `full` on Diving-48, beats it under aggressive selection on K400, and remains below it on SSv2, where boundary-state evidence resists the kept-set protocol. The continuous weights dominate the categorical kept set at every operating point, and direct scoring delivers the corpus at roughly 25× fewer oracle calls than the greedy procedure it approximates. A complementary diagnostic, selection-aware adaptation of the recognition head under frozen encoders, is reported in Appendix E.1. The Discussion interprets these regularities.

5 Discussion

5.1 What does temporal importance mean?

The intuition behind Meaningful Moments is that some moments in a video carry the evidence for an action and most do not. The results let me state the operational version of that intuition precisely: an MM importance score measures how much a segment contributes to a verification-oriented VLM's ability to confirm the labeled action. It is counterfactual in origin (Section 3.3), action-conditioned by construction, and approximated in practice by a single direct-scoring call.

This definition captures something real and is not reducible to cheaper notions of importance. Inverting it hurts recognition on every dataset, both at the hard cut and integrated over the whole retained-fraction axis, so the scores are not noise. It is also not simply motion: optical-flow ordering is a strong baseline but behaves differently across datasets, and the VLM signal carries finer class information on top-5 separation. What the definition does not promise is identity with *recognizer* importance. The SSv2 results show the gap most clearly: the oracle's verification evidence concentrates in mid-action motion, while a frozen recognizer discriminates SSv2 classes by boundary states, and the misalignment produces the reversed interior gaps of Section 4.2.3. MM importance therefore means "what a VLM needs to verify the labeled action," which overlaps with, but is not the same as, "what a recognizer needs to classify the video." The dataset dependence documented in Section 4 is the geometry of that overlap.

5.2 Why dataset structure determines whether selection helps

The three datasets produce three different selection outcomes from one underlying rule: selection helps when the evidence the oracle marks coincides with the evidence the recognizer uses, and when that evidence is concentrated enough to be worth isolating. Two conditions matter, alignment and concentration, and the datasets realize three different combinations of them.

Diving-48 satisfies both. Its classes are stereotyped temporal arcs whose discriminative content is the dive itself, the oracle and the recognizer agree on where that content is, and the surrounding footage is genuinely dispensable. Selection therefore preserves recognition, and the shuffle reversal of Section 4.4.2 confirms the mechanism: the kept set concentrates rotation content whose order is the class, so it has the most to lose under shuffling. K400 satisfies alignment but not concentration. Its classes are appearance-cued activities visible through most of the clip, so there is no hidden evidence for selection to find; the interior of the α axis is flat and shuffling costs nothing. Selection still wins at the hard cut, but for a different reason: with a fixed sixteen-frame budget, discarding ambiguous footage concentrates the budget on unambiguous action evidence. The gain comes from removing distractors, not from finding evidence. SSv2 satisfies concentration but not alignment. Its classes are discriminated by boundary states and spatial relations, while the oracle marks mid-action motion instead; the corpus-level temporal profile (Figure 6) makes this visible, with SSv2 importance depressed precisely in the boundary fifths of the clip. The best selection operating point stays below `full` while the per-template breakdown splits exactly along this line: boundary-discriminated templates lose under the cut and motion-discriminated templates gain.

The practical reading is that dataset difficulty is not the relevant variable. What determines whether MM-guided selection helps is where a dataset's class-discriminative evidence lives relative to what a verification-oriented oracle marks, and that property can be reasoned about before running a recognizer.

5.3 What oracle disagreement means for pseudo-label reliability

The cross-oracle numbers support a calibrated reading of what MM labels are. Frontier-model pairs agree at Spearman 0.49–0.61: far above chance under permutation nulls, far below ceiling. A sizeable fraction of segment-level decisions are genuinely model-dependent, and kept-set agreement is lower still because the binary boundary inherits each oracle's own thresholding behavior. A single-oracle MM label should therefore be treated as one competent annotator's judgment rather than as ground truth, which is exactly the framing the Limitations section adopts. That InternVL3-38B sits apart at 0.35–0.37 against all three closed-weights models suggests the importance prior is tied to model family and capability tier rather than being an artifact of any one pair.

The downstream consistency result locates where this noise matters. Per video and per segment, oracles disagree; aggregated over an evaluation pool, every oracle reproduces the same recognition ordering, with the same conditions winning on the same datasets. Segment-level disagreement washes out at the level where MM is intended to be used, namely corpus-scale training signals and pooled evaluation statistics. The reliability unit for MM is the aggregate, and at that unit the labels are robust to the choice of oracle. Per-segment identity is the wrong unit to demand from current VLMs, and applications that need it should expect to ensemble oracles or add human verification.

5.4 What MM can be used for

The only downstream task evaluated in this thesis is controlled-sampling video action recognition under frozen classifiers. The corpus, however, was built to be reusable, and its three label layers

(binary kept sets, continuous weights, and precheck metadata) match the input requirements of several lines of follow-on work. None of the following is evaluated here; each is future work.

- **Training temporal selectors.** The binary labels are direct supervision for segment-policy learners in the frame- and token-selection family [31, 13]: a selector trained against MM targets inherits the oracle's evidence judgments without any oracle call at inference time.
- **Encoder adaptation under MM-cut inputs.** Recognizers can be adapted to the selected-input distribution rather than evaluated frozen on it. The frozen-encoder head-adaptation diagnostic in Appendix E.1 is a first instance and suggests that much of the off-the-shelf kept-set deficit is closeable domain shift.
- **Sampling policies and reward modeling.** The continuous weights provide a dense per-segment reward for supervised or reinforcement-learned video-selection agents, and the precheck probabilities provide a video-level gating signal.
- **Long-video recognition, summarization, and retrieval.** Action-conditioned segment evidence acts as an index: for long inputs, MM-style scores can prioritize which spans to decode, summarize, or embed.

5.5 Implications for VLM-as-oracle pipelines

The production pipeline yields four pieces of operational guidance for anyone using a VLM as a large-scale annotator. First, measure the extraction mode's own test-retest noise before optimizing anything else. The decisive fact in Section 4.6 was not that direct scoring agreed with greedy removal, but that greedy agreed with itself at only 0.146 Jaccard: once the expensive mode's noise floor is known, a 25× cheaper single-call mode that matches it within that floor is strictly preferable, and the relevant budget currency is oracle calls, which tracks wall-clock and token cost almost linearly. Second, treat extraction mode and prompt template as separable design axes and expect mode to dominate. Within direct mode, swapping prompts preserved kept-set agreement at Jaccard 0.83; across modes the agreement was 0.21. Prompt engineering tunes the signal; mode choice defines it. Third, ask the oracle for continuous scores and derive categorical decisions downstream. The oracle's own kept-set field was dominated by simple thresholds on its retained weights on all three datasets (Section 4.3), so the scores are the asset and the categorical field is a convenience. Fourth, decision-token logits are free quality metadata: the two-stage precheck rule of Section 3.3.4 produced corpus-scale video-level scorability flags with zero parse failures and no separate filtering model, at the cost of requiring a backend that exposes token likelihoods (Appendix F.5).

6 Limitations

No human temporal-importance ground truth. Meaningful Moments has no human-annotated temporal-importance ground truth. The cross-oracle agreement study (Section 4.5) therefore measures consistency among VLM annotators rather than correctness against human judgment, and inter-oracle agreement bounds labeling reliability only from above. The downstream-recognition consistency analysis partially mitigates this, because the importance ordering carries recognition signal across multiple oracles rather than the primary one alone, but a human-calibrated segment-level importance benchmark is left to future work.

Single-oracle primary annotation. All headline corpus statistics and recognition results derive from a single primary oracle, Qwen3-VL-32B. The four-oracle study ($n = 369$ joint-precheck-pass videos) shows that three frontier-class VLMs cluster at pairwise Spearman 0.49–0.61 on the importance signal, which bounds primary-oracle dependence but does not remove it: a different primary oracle would shift the exact per-segment scores even where the aggregate recognition trends are consistent. Producing the corpus from a multi-oracle ensemble rather than a single annotator is left to future work.

Action-label conditioning. The oracle scores each video conditioned on its ground-truth action label, so Meaningful Moments is a label-conditioned importance signal rather than an open-ended one. A pipeline without ground-truth labels would require either an upstream classification step or an unconditioned scoring prompt. Conditioning on the label was a deliberate choice that sharpens the importance signal for a known class, but it may bias the oracle toward *verifying* that the labeled action is present rather than *discovering* which moments are salient.

Prompt-defined binary kept-set semantics. The binary `important/filler` partition is an operational convention inherited from the oracle prompt and its raw kept-set field, not a human-calibrated decision boundary. The threshold and budget sweeps (Section 4.3) partially test sensitivity to this choice by evaluating the retained continuous weights directly, and they show that the categorical cut is dominated by other operating points on all three datasets, which is evidence that the binary boundary is a convenience rather than an optimum.

Recognizer dependence. The recognition results are conditional on the specific frozen recognizers used: V-JEPA 2 paper checkpoints for SSv2 and Diving-48, and VideoMAE-Large for K400. Per-recognizer accuracies, and the size of the gap between `vlm-selected` and `full`, can shift under different backbone families (for example TimeSformer or MViT) or different probe-training regimes. The dataset-dependence pattern is the robust claim; the precise per-condition margins should be read as backbone-specific, and replication across recognizer families is left to future work.

Dataset-specific precheck failures and coverage bias. Videos that fail the oracle's video-level precheck are excluded from the corpus and from every headline evaluation pool. Full-corpus precheck-fail rates, which combine precheck-NO ("action not visibly present") with the small precheck-SKIP fraction, vary by dataset: 4.98% on SSv2, 7.80% on K400, and 15.34% on Diving-48. The corpus and all reported results are thus conditional on the oracle's scorability judgment, and the higher Diving-48 rate means its eval pool is the most heavily filtered. Which classes precheck-NO concentrates in, and whether that filtering biases the per-class findings, is left as future work.

Licensing and access constraints. Meaningful Moments releases per-segment importance scores and pseudo-labels, not video content. The source videos for SSv2, K400, and Diving-48 remain under their respective dataset licenses: the MM labels are released under CC-BY-4.0, but downstream users must obtain the underlying videos separately under each dataset's original terms. This keeps redistribution compliant at the cost of the corpus not being self-contained.

7 Conclusion and future work

7.1 Summary

This thesis introduced Meaningful Moments, a corpus of VLM-generated temporal importance labels covering 4,576,082 per-segment scores on 499,299 precheck-passed videos across the SSv2, K400, and Diving-48 label spaces, together with a controlled evaluation framework for testing whether such labels help video action recognition. The central claim was conditional from the start: VLM-derived temporal importance labels are useful, but only when the VLM's notion of importance aligns with the class-discriminative evidence the recognizer needs.

The evidence supports both halves of that claim. The importance signal is informative on all three datasets: inverting it collapses recognition at the hard cut, anti-importance orderings lose to random across the whole retained-fraction axis, and four independent oracles reproduce the same downstream ordering. Its usefulness follows the dataset map: on Diving-48, where evidence is concentrated and aligned, selection preserves full-video recognition; on K400, where evidence is diffuse but redundant, aggressive selection improves on uniform sampling by removing distractors; on SSv2, where classes are discriminated by boundary states the oracle does not prioritize, the signal is real but selection does not recover full-video performance. Along the way, the pipeline results showed that the oracle's continuous scores dominate its own categorical kept set and that single-call direct scoring matches iterative greedy removal within greedy's own noise floor at roughly 25× fewer oracle calls, which is what made corpus-scale annotation feasible.

7.2 Future work

Each item below addresses the correspondingly ordered limitation in Section 6; the seventh limitation, licensing and access, is addressed by the released and versioned v1.0 label distribution rather than by future work.

1. **Human-calibrated importance benchmark** (no human ground truth). Collect human segment-importance annotations on a stratified MM subsample and measure human-oracle alignment, turning the cross-oracle upper bound into a calibrated accuracy estimate.
2. **Multi-oracle ensemble labels** (single primary oracle). Produce an ensemble label layer from the frontier band, using per-segment score averaging and agreement-weighted kept sets, and test whether ensembling improves downstream selection.
3. **Label-free scoring** (action-label conditioning). Add an unconditioned or upstream-classified scoring variant and measure how far importance shifts from verifying a given class toward discovering salient action content.
4. **Learned kept-set boundaries** (prompt-defined binary semantics). Replace the oracle's categorical kept set with thresholds learned per class or per dataset over the continuous weights, which Section 4.3 suggests is strictly better.
5. **Multi-backbone replication** (recognizer dependence). Re-run the headline protocol under additional recognizer families and probe-training regimes to separate substrate effects from backbone effects.
6. **Precheck bias analysis** (precheck coverage bias). Characterize which classes precheck failures concentrate in, especially on Diving-48, and evaluate recovery strategies for scorable-but-rejected videos.

References

[1] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Rongyao Fang, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Li Ying Meng, Xuancheng Ren, Xin-yi Ren, Sibo Song, Yu-chen Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Wang, Yihe Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Botao Zheng, Humen Zhong, Jingren Zhou, Fanxi Zhou, Yuanzhi Zhu, and Keming Zhu. Qwen3-vl technical report. *ArXiv*, abs/2511.21631, 2025.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023.

[4] Lei Chen, Zhan Tong, Yibing Song, Gangshan Wu, and Limin Wang. Efficient video action detection with token dropout and context refinement. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10354–10365, 2023.

[5] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-Wei Chao, Byung Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, and Sergey Tulyakov. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13320–13331, 06 2024.

[6] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis, 2025.

[7] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851, 2017.

[8] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, 2015.

[9] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. Localizing moments in video with natural language. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5804–5813, 2017.

[10] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[11] Haroon Idrees, Amir R. Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, February 2017.

[12] Will Kay, João Carreira, Karen Simonyan, Brian Hu Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Apostol Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017.

[13] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6231–6241, 2019.

[14] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[15] Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11846–11858. Curran Associates, Inc., 2021.

[16] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[17] Jintao Lin, Haodong Duan, Kai Chen, Dahua Lin, and Limin Wang. Ocsampler: Compressing videos to one clip with single-step sampling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13884–13893, 2022.

[18] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023.

[19] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[20] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46212–46244. Curran Associates, Inc., 2023.

[21] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2630–2640, 2019.

[22] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *ArXiv*, abs/1806.07421, 2018.

[23] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[25] Yao Rong, Tobias Leemann, Vadim Borisov, Gjergji Kasneci, and Enkelejda Kasneci. A consistent and efficient evaluation strategy for attribution methods. *arXiv preprint arXiv:2202.00449*, 2022.

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[27] Mattia Soldan, Alejandro Pardo, Juan León Alcázar, Fabian Caba Heilbron, Chen Zhao, Silvio Giancola, and Bernard Ghanem. Mad: A scalable dataset for language grounding in videos from movie audio descriptions. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5016–5025, 2022.

[28] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training, 2022.

[29] Junke Wang, Xitong Yang, Hengduo Li, Li Liu, Zuxuan Wu, and Yu-Gang Jiang. Efficient video transformers with spatial-temporal token selection, 2022.

[30] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *ArXiv*, abs/2307.06942, 2023.

[31] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis. Adaframe: Adaptive frame selection for fast video recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1278–1287, 2019.

[32] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[33] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2933–2942, 2017.

[34] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[35] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025.

A Prompt Templates and Oracle Configuration

This appendix reproduces the oracle prompts, the response schema the parser expects, a worked parsed example, and the model identifiers behind the four oracles used in this work.

A.1 Direct-scoring prompt family

The production corpus is annotated entirely with the direct-scoring prompt family. A single oracle call returns a video-level precheck decision (YES/NO/SKIP), a per-segment importance score in $[0, 100]$, a coarse phase label per segment, and an explicit minimum-sufficient-set list. Each dataset uses a variant that shares this structure but tailors the class vocabulary and the phase taxonomy to the substrate. The generic, SSv2, K400, and Diving-48 variants are reproduced verbatim below from `oracle/scripts/mss/prompts/direct_scoring{,_ssv2,_k400,_diving48}.md`.

Generic (`direct_scoring`).

```
---
name: direct_scoring
version: "3.1"
description: Single-query segment importance scoring for general video action recognition. Outputs per-segment
      importance scores and minimum sufficient set without iterative removal.
variables:
  - action_label: The action class label
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
You are annotating which temporal segments of a video are needed to recognize a specific action.

ACTION: {action_label}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_label}"?

Be INCLUSIVE. Answer YES if:
- You see body movements, objects, or environments consistent with the described action
- The overall trajectory of events matches (even if some frames are blurry or ambiguous)
- You can piece together from multiple frames that this action is happening or has happened

Answer NO only if:
- The video clearly shows a DIFFERENT action than the label
- The relevant objects, actors, or motions are absent across the entire video
- The video is entirely unrelated content

Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.
```

```
STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would the action
      become harder to recognize?"

PHASE ANALYSIS -- score each segment by what it contributes:
  a) EXECUTION: The defining motion of the action (the swing, the step, the strum, the throw, the lift) --
       usually the highest-value content.
  b) DISAMBIGUATION: Frames that distinguish this action from a near-neighbor. If multiple actions involve
       similar setup or environment, the segments that uniquely identify THIS action are critical.
  c) CONTACT / INITIATION: The moment force is applied, an object is grasped, or the action begins.
  d) RESULT / COMPLETION: End state that confirms the action occurred.
  e) CONTINUATION / TRAJECTORY: Sustained motion showing direction, speed, or manner -- important for
       repetitive or continuous actions.
  f) SETUP / APPROACH: The actor positions for the action; useful but often redundant.

Not every action exhibits every phase. Continuous actions (e.g. sustained activities) may be dominated by
     EXECUTION/CONTINUATION; discrete actions may have a clear CONTACT -> EXECUTION -> RESULT arc.

POSITION INDEPENDENCE: Score by visual content, not segment position. Early segments are not inherently more
     important. Late segments showing results, completion, or disambiguation can be just as critical.

ACTOR + MOTION OVER SCENE: Segments where the action is VISIBLY being performed by an actor outrank segments
     that show only the setting, equipment, or aftermath without the action happening.

MINIMUM SEGMENT GUIDANCE (rough, action-dependent):
- Brief discrete events (single contact, single throw, single step): 2-3 segments
- Two-phase actions (approach -> execute, execute -> result): 3-4 segments
- Continuous / repetitive activities (running, swimming, dancing, playing an instrument): 3-5 segments
     capturing characteristic motion across the clip
- Cause-and-effect chains (action causes a downstream visible effect): 4-5 segments
- Multi-stage activities with distinct phases: 4-6 segments

Important segments may appear anywhere -- beginning, middle, or end.

Err toward INCLUDING a segment if you're unsure -- it's worse to drop a segment that matters than to keep one
     that doesn't.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the action unrecognizable or ambiguous
- 50-79: This segment adds meaningful evidence (shows a distinct phase or disambiguates)
- 20-49: Mildly helpful but redundant with adjacent segments
- 0-19: Filler, static pause, scene-only with no action visible, or no relevant content

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what you see happening in the video, max 30 words>",
  "segments": [
    {{
      "segment_id": <1-based index>,
      "time_range": "<start_s>-<end_s>",
      "importance": <0-100>,
      "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
      "reason": "<max 10 words>"
    }}
  ],
  "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
  "rationale": "<why these segments are needed together to recognize the action, 2-3 sentences>"
}}

You MUST include ALL {n_segments} segments in the "segments" array -- one entry per segment, no omissions.
```

Something-Something v2 (`direct_scoring_ssv2`).

```
---
name: direct_scoring_ssv2
version: "1.0"
description: Single-query segment importance scoring for Something-Something V2 videos. Outputs per-segment
     importance scores and minimum sufficient set without iterative removal.
variables:
  - action_template: The SSv2 action label with specific objects
  - placeholders: The objects involved (comma-separated)
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
You are annotating which temporal segments of a video are needed to recognize a specific action.

ACTION: {action_template}
OBJECT(S): {placeholders}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_template}"?

Be INCLUSIVE. Answer YES if:
- You see objects and body movements consistent with the described action
- The overall trajectory of events matches (even if some frames are blurry or ambiguous)
- You can piece together from multiple frames that this action is happening or has happened

Answer NO only if:
- The video clearly shows a DIFFERENT action (e.g., label says "pushing" but you see "pulling")
- The relevant objects are completely absent from the entire video
- The video is entirely unrelated content

Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.

STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would the action
      become harder to recognize?"

IMPORTANT: Most SSv2 actions require MULTIPLE temporal phases to distinguish from similar actions. Think about
      what makes THIS action different from related ones:

PHASE ANALYSIS -- consider which segments capture:
  a) EXECUTION: The core motion (rotation, translation, deformation, release) -- can appear anywhere in the
       video
  b) DISAMBIGUATION: Any segment that distinguishes this action from a similar one
     - "pushing left to right" vs "pushing right to left" -> need enough frames to confirm direction
     - "throwing" vs "dropping" -> need the release moment
     - "pretending to X" vs actually doing X -> need to see the non-completion
  c) RESULT/COMPLETION: End state confirming the action occurred (or did NOT occur for "pretending")
  d) CONTACT/INITIATION: Moment of grip, touch, or force application
  e) SETUP/APPROACH: Hand reaches toward object, positions for action
  f) CONTINUATION/TRAJECTORY: Sustained motion showing direction, speed, manner

POSITION INDEPENDENCE: Score each segment based on its visual content, NOT its position. Early segments are
     not inherently more important. Late segments showing results, completion, or disambiguation are equally
     critical.

MINIMUM SEGMENT GUIDANCE:
- Simple static displays ("showing X behind Y"): 1-2 segments
- Single discrete motions ("picking X up", "turning X over"): 2-3 segments
- Two-phase actions ("putting X into Y", "dropping X"): 3-4 segments
- Sustained/continuous actions ("spinning", "rolling"): 3-5 segments
- Cause-and-effect chains ("pushing X so it falls"): 4-5 segments
- Intentional non-completion ("pretending to X"): 4-6 segments (need context showing the absence)

Important segments may appear anywhere -- beginning, middle, or end.
```

```
Err toward INCLUDING a segment if you're unsure -- it's worse to drop a segment that matters than to keep one
     that doesn't.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the action unrecognizable or ambiguous
- 50-79: This segment adds meaningful evidence (shows a distinct phase or disambiguates)
- 20-49: Mildly helpful but redundant with adjacent segments
- 0-19: Filler, static pause, or no relevant content

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what you see happening in the video, max 30 words>",
  "segments": [
    {{
      "segment_id": <1-based index>,
      "time_range": "<start_s>-<end_s>",
      "importance": <0-100>,
      "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
      "reason": "<max 10 words>"
    }}
  ],
  "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
  "rationale": "<why these segments are needed together to recognize the action, 2-3 sentences>"
}}

You MUST include ALL {n_segments} segments in the "segments" array -- one entry per segment, no omissions.
```

Kinetics-400 (`direct_scoring_k400`).

```
---
name: direct_scoring_k400
version: "1.1"
description: Single-query segment importance scoring for Kinetics-400 videos. Outputs per-segment importance
     scores and minimum sufficient set without iterative removal.
variables:
  - action_label: The K400 action class (e.g., 'playing basketball', 'swimming')
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
You are annotating which temporal segments of a video are needed to recognize a specific action.

ACTION: {action_label}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_label}"?

Be INCLUSIVE. Answer YES if:
- You see an actor and motion consistent with "{action_label}"
- The overall trajectory of events matches (even if some frames are blurry, occluded, or partially out of
     frame)
- You can piece together from multiple frames that this action is happening or has happened

Answer NO only if:
- The video clearly shows a DIFFERENT action than the label
- You see only a scene, setting, or equipment with no actor performing the action
- The relevant actor or motion is absent throughout

Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.

STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
```

```
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would the action
      become harder to recognize?"

IMPORTANT: Many K400 classes are continuous or repetitive. Think about which segments capture the
     characteristic motion vs. which are repetition with no new information.

PHASE ANALYSIS -- score each segment by what it contributes:
  a) EXECUTION: The defining motion of the action -- dribbling/shooting (basketball), strumming/fingering (
       guitar), kicking (soccer), strokes (swimming), chopping/stirring (cooking), step patterns (dance).
       USUALLY the highest-value segments.
  b) CONTINUATION / TRAJECTORY: Sustained or repeated motion showing manner, rhythm, or speed. For continuous
       activities (running, swimming, playing an instrument), MULTIPLE segments of repeated execution carry
       signal -- they confirm this is the activity, not a single momentary motion.
  c) DISAMBIGUATION: Frames that distinguish "{action_label}" from a near-neighbor class. Many K400 classes
       share scenes/equipment (e.g. tennis vs badminton, jogging vs running, playing guitar vs playing bass) --
        segments that uniquely identify THIS class are critical.
  d) CONTACT / KEY EVENT: The moment of impact, release, or peak action (a serve, a swing, a jump, a release).
  e) RESULT: A visible outcome that confirms the action occurred (ball entering hoop, food landing on plate).
  f) SETUP / APPROACH: Actor positions or prepares; usually lower-value, but can matter for rapid actions.

POSITION INDEPENDENCE: Score by visual content, not position. Early segments are not inherently more important.
      Late segments showing the activity in progress can be just as critical.

ACTOR + MOTION OVER SCENE: Segments where the action is VISIBLY being performed by an actor outrank segments
     that show only the venue, equipment, or aftermath. A segment of only a basketball court (no players) or
     only a kitchen (no cooking) deserves a low score even though the scene is on-class.

MINIMUM SEGMENT GUIDANCE (rough, action-dependent):
- Continuous activities sustained throughout the clip (swimming, running, playing an instrument, dancing): 3-5
      segments capturing execution at different points in the clip
- Repetitive sports actions (dribbling, lifting weights, jumping rope): 3-5 segments showing the repeated
     cycle
- Discrete sports moments (a single throw, kick, dunk, dive): 3-4 segments around the contact / release
- Multi-step daily activities (cooking a dish, assembling something): 4-6 segments covering distinct sub-
     actions
- Brief expressive actions (shaking hands, hugging, sneezing): 2-3 segments around the contact / peak

Important segments may appear anywhere -- beginning, middle, or end.

Err toward INCLUDING a segment if you're unsure -- it's worse to drop a segment that matters than to keep one
     that doesn't.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the action unrecognizable or ambiguous
- 50-79: Meaningful evidence -- shows a distinct phase, repetition cycle, or disambiguates from a near class
- 20-49: Mildly helpful but redundant with adjacent execution segments
- 0-19: Filler, static pause, scene-only with no action visible, or no relevant content

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what you see happening in the video, max 30 words>",
  "segments": [
    {{
      "segment_id": <1-based index>,
      "time_range": "<start_s>-<end_s>",
      "importance": <0-100>,
      "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
      "reason": "<max 10 words>"
    }}
  ],
  "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
  "rationale": "<why these segments are needed together to recognize the action, 2-3 sentences>"
}}

You MUST include ALL {n_segments} segments in the "segments" array -- one entry per segment, no omissions.
```

Diving-48 (direct_scoring_diving48).

```
---
name: direct_scoring_diving48
version: "1.0"
description: Single-query segment importance scoring for Diving-48 videos. Encodes FINA-rule diving phases (
     approach/takeoff/flight/entry) and the 4-attribute label structure (takeoff group x somersaults x twists
     x body position).
variables:
  - action_label: Diving-48 action label (e.g., 'forward 3.5 somersaults in pike position')
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
You are annotating which temporal segments of a video are needed to recognize a specific competitive dive.

ACTION: {action_label}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

This is a competitive dive, scored under FINA rules. The action label encodes a
4-attribute classification:

- Takeoff group: forward, back, reverse, inward, or armstand. The takeoff group
  determines the diver's facing direction and the relationship between body
  orientation and rotation direction.
- Number of somersaults: in half-rotation increments (1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5).
- Number of twists: in half-rotation increments (0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5).
- Body position during flight: tuck (knees to chest), pike (legs straight,
  hip fold), straight/layout (fully extended), or free (combination position).

A standard dive proceeds in four phases (FINA-defined judging components):
  APPROACH: setup at the end of the board (running for forward springboard,
            standing for back/inward, armstand for platform armstand dives).
  TAKEOFF: push off the board or platform. Sets the angular momentum and
            initial rotation direction; brief but discriminative for the takeoff
            group.
  FLIGHT: somersault and twist rotations executed in the announced body
            position. This phase carries most of the class-discriminative
            information: the number of somersaults, the number of twists, and
            the body position are all observed here.
  ENTRY: head-first or feet-first water entry. Confirms the announced body
            position transitioning to a vertical line for scoring.

Discriminating between Diving-48 classes typically requires counting somersaults
and twists during FLIGHT, identifying the body position during FLIGHT, and
identifying the takeoff group from the diver's orientation during TAKEOFF.
ENTRY is also important for confirming body position. APPROACH is largely
shared across many classes and rarely class-discriminative.

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_label}"?

Be INCLUSIVE. Answer YES if:
- You can see a diver executing a dive with takeoff group, rotation count, body position,
  and twist count broadly consistent with the label
- Motion blur during fast rotation is normal -- count rotations from peak-to-peak
  body orientation changes rather than expecting crisp frames
- Some attributes may be hard to verify exactly (e.g., 4 vs 4.5 somersaults can be
  visually similar) -- accept reasonable consistency, not exact pixel-level proof

Answer NO only if:
- The takeoff group is wrong (e.g., a forward takeoff for a "back" label)
- The body position is clearly wrong (e.g., visibly straight/layout for a "tuck" label,
  or vice-versa) throughout FLIGHT
- The dive is missing entirely (no diver visible, water-only footage, or unrelated content)
- The somersault count is off by >= 1 full somersault from the label (e.g., a single
  somersault for a "3.5 somersault" label)
```

```
Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.

STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would
the dive become harder to recognize as '{action_label}' specifically?"

DIVING PHASE -> SEGMENT PHASE MAPPING (use these phase labels in the output JSON):
  APPROACH segments -> "setup" (low priority unless armstand or unusual stance)
  TAKEOFF segments -> "contact" (HIGH priority -- encodes takeoff group)
  FLIGHT segments -> "execution" (HIGHEST priority -- encodes somersaults, twists, body position)
                     or "continuation" if the rotation is sustained across many segments
  ENTRY segments -> "result" (HIGH priority -- confirms body position and rotation completion)

Use "disambiguation" for segments that uniquely separate this dive from a
near-neighbor class (e.g., the moment that distinguishes 3 vs 3.5 somersaults
by the entry orientation, or pike vs tuck by visible knee position mid-flight).

PRIORITY GUIDANCE FOR DIVING:
- FLIGHT segments are almost always HIGH (60-100). The number of somersaults and
  twists, and the body position, can only be verified during flight.
- TAKEOFF is brief (often 1 segment) but HIGH (70-95). It anchors the takeoff group.
  Drop it only if the takeoff group is unambiguously visible elsewhere.
- ENTRY segments are MEDIUM-HIGH (50-80). The entry confirms body-position transition
  to vertical and is essential for verifying the announced body position.
- APPROACH segments are usually LOW (10-30) for forward/back/reverse/inward dives where
  the approach is generic. EXCEPTION: armstand dives -- the armstand setup IS the
  takeoff group signature, so APPROACH for armstand is HIGH.
- Empty pre-dive footage (diver not yet on the platform, just water, post-splash water
  rings) is FILLER (0-15).

POSITION INDEPENDENCE: Score by visual content, not segment position. The dive may
start late or end early in the clip -- find the actual TAKEOFF -> FLIGHT -> ENTRY span
and score those highest regardless of where they fall in the timeline.

MINIMUM SEGMENT GUIDANCE FOR DIVING:
- Most dives need 3-5 keep segments: 1 takeoff + 2-3 flight + 1 entry
- High-rotation dives (3.5, 4, 4.5 somersaults) may need 4-6 flight segments to
  count rotations confidently
- Multi-twist dives (1.5, 2, 2.5+ twists) similarly benefit from extra flight
  coverage to count twist axes
- Armstand dives need APPROACH segments (the armstand) plus TAKEOFF + FLIGHT + ENTRY,
  so 4-6 keep segments total

Err toward INCLUDING a segment if it shows any portion of the rotating body in flight
or the entry -- it's worse to drop a flight segment that disambiguates somersault count
than to keep one that's slightly redundant.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the dive's class unrecognizable or ambiguous
  (e.g., the only flight segment showing a particular rotation, the takeoff that
  identifies the takeoff group, the entry that confirms body position).
- 50-79: Meaningful evidence -- a flight segment among several, the entry, or a
  disambiguating frame between two near-neighbor classes.
- 20-49: Mildly helpful but redundant with adjacent flight segments showing the
  same rotation phase.
- 0-19: Pre-dive footage, post-splash water-only frames, or other filler with no
  diver in flight.

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what dive you see, including takeoff group, rotation count, twist count, body position;
      max 30 words>",
  "segments": [
    {{
```

```
    "segment_id": <1-based index>,
    "time_range": "<start_s>-<end_s>",
    "importance": <0-100>,
    "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
    "reason": "<max 10 words>"
   }}
 ],
 "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
 "rationale": "<why these segments are needed together to recognize this specific dive class -- call out
      which segments verify takeoff group, somersault count, twist count, and body position; 2-3 sentences>"
}}

You MUST include ALL {n_segments} segments in the "segments" array -- one entry per segment, no omissions.
```

A.2 Greedy-removal prompt family

The greedy-removal mode is used only for the mode-comparison ablations and never for the production corpus. It poses a single counterfactual sufficiency question per oracle call: given the currently retained segments, is the labeled action still identifiable? Iterating this question while greedily removing segments yields the minimal sufficient subset, at a cost of one oracle call per candidate per iteration (about 25 calls per video in the pilots) rather than the single call that direct scoring needs. The three variants (`mss_verification.md` and its `_ssv2` and `_k400` variants, under `oracle/scripts/mss/prompts/`) are reproduced verbatim below; there is no Diving-48 greedy variant, and greedy runs on Diving-48 fall back to the generic template.

Formal procedure. Writing $q(K) = P(\text{YES} \mid \neg\text{SKIP})$ for the oracle's affirmation probability on the kept list K (decision-token logits as in Section 3.3.4), each iteration queries every remaining candidate and removes the least-damaging removable segment:

$$\text{Removable}(K) = \{\, s_i \in K : q(K \setminus \{s_i\}) > \tfrac{1}{2} \,\},$$
$$s^\star = \operatorname*{arg\,max}_{s_i \in \text{Removable}(K)} q(K \setminus \{s_i\}),$$
$$\text{Greedy}(K) = \begin{cases} K, & |K| = 1 \text{ or } \text{Removable}(K) = \emptyset, \\ \text{Greedy}(K \setminus \{s^\star\}), & \text{otherwise,} \end{cases}$$

starting from $K = (s_0, \ldots, s_{T-1})$. The arg max form is equivalent to removing the segment with the smallest confidence drop $q(K) - q(K \setminus \{s_i\})$ (the deterministic "score" removal strategy), and querying every candidate per iteration is what produces the quadratic worst-case call count of Section 3.7.5.

Generic (`mss_verification`).

```
---
name: mss_identification_ssv2
version: "4.0"
description: Identifies the minimum set of segments needed to recognize a Something-Something V2 action in a
     video.
variables:
  - action_template: The SSv2 action label with specific objects
  - placeholders: The objects involved (comma-separated)
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
‘‘‘
```

```
You are annotating which temporal segments of a video are needed to recognize a specific action.

ACTION: {action_template}
OBJECT(S): {placeholders}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_template}"?

Be INCLUSIVE. Answer YES if:
- You see objects and body movements consistent with the described action
- The overall trajectory of events matches (even if some frames are blurry or ambiguous)
- You can piece together from multiple frames that this action is happening or has happened

Answer NO only if:
- The video clearly shows a DIFFERENT action (e.g., label says "pushing" but you see "pulling")
- The relevant objects are completely absent from the entire video
- The video is entirely unrelated content

Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.

STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would the action
      become harder to recognize?"

IMPORTANT: Most SSv2 actions require MULTIPLE temporal phases to distinguish from similar actions. Think about
      what makes THIS action different from related ones:

PHASE ANALYSIS -- consider which segments capture:
  a) SETUP/APPROACH: Hand reaches toward object, positions for action
  b) CONTACT/INITIATION: Moment of grip, touch, or force application
  c) EXECUTION: The core motion (rotation, translation, deformation, release)
  d) CONTINUATION/TRAJECTORY: Sustained motion showing direction, speed, manner
  e) RESULT/COMPLETION: End state confirming the action occurred (or did NOT occur for "pretending")
  f) DISAMBIGUATION: Any segment that distinguishes this action from a similar one
     - "pushing left to right" vs "pushing right to left" -> need enough frames to confirm direction
     - "throwing" vs "dropping" -> need the release moment
     - "pretending to X" vs actually doing X -> need to see the non-completion

MINIMUM SEGMENT GUIDANCE:
- Simple static displays ("showing X behind Y"): 1-2 segments
- Single discrete motions ("picking X up", "turning X over"): 2-3 segments
- Two-phase actions ("putting X into Y", "dropping X"): 3-4 segments
- Sustained/continuous actions ("spinning", "rolling"): 3-5 segments
- Cause-and-effect chains ("pushing X so it falls"): 4-5 segments
- Intentional non-completion ("pretending to X"): 4-6 segments (need context showing the absence)

Err toward INCLUDING a segment if you're unsure -- it's worse to drop a segment that matters than to keep one
     that doesn't.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the action unrecognizable or ambiguous
- 50-79: This segment adds meaningful evidence (shows a distinct phase or disambiguates)
- 20-49: Mildly helpful but redundant with adjacent segments
- 0-19: Filler, static pause, or no relevant content

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what you see happening in the video, max 30 words>",
  "segments": [
    {{
      "segment_id": <1-based index>,
      "time_range": "<start_s>-<end_s>",
```

```
     "importance": <0-100>,
     "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
     "reason": "<max 10 words>"
    }}
  ],
  "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
  "rationale": "<why these segments are needed together to recognize the action, 2-3 sentences>"
}}

Only include segments scoring >= 20 in the "segments" array (omit pure filler).
```

Something-Something v2 (`mss_verification_ssv2`).

```
---
name: mss_identification_ssv2
version: "4.0"
description: Identifies the minimum set of segments needed to recognize a Something-Something V2 action in a
     video.
variables:
  - action_template: The SSv2 action label with specific objects
  - placeholders: The objects involved (comma-separated)
  - n_segments: Total number of segments in the video
  - duration: Video duration in seconds
  - segment_duration: Length of each segment in seconds
---
‘‘‘
‘‘‘
You are annotating which temporal segments of a video are needed to recognize a specific action.

ACTION: {action_template}
OBJECT(S): {placeholders}
VIDEO DURATION: {duration}s ({n_segments} segments of {segment_duration}s each, indexed 1 to {n_segments})

STEP 1 -- ACTION PRESENCE CHECK:
Watch the full video. Does it plausibly depict "{action_template}"?

Be INCLUSIVE. Answer YES if:
- You see objects and body movements consistent with the described action
- The overall trajectory of events matches (even if some frames are blurry or ambiguous)
- You can piece together from multiple frames that this action is happening or has happened

Answer NO only if:
- The video clearly shows a DIFFERENT action (e.g., label says "pushing" but you see "pulling")
- The relevant objects are completely absent from the entire video
- The video is entirely unrelated content

Answer SKIP only if the video is corrupted, black, or completely unreadable.

If NO or SKIP, output the JSON below and stop.

STEP 2 -- SEGMENT IMPORTANCE ANNOTATION:
For each {segment_duration}s segment (1 to {n_segments}), decide: "If I removed this segment, would the action
      become harder to recognize?"

IMPORTANT: Most SSv2 actions require MULTIPLE temporal phases to distinguish from similar actions. Think about
      what makes THIS action different from related ones:

PHASE ANALYSIS -- consider which segments capture:
  a) SETUP/APPROACH: Hand reaches toward object, positions for action
  b) CONTACT/INITIATION: Moment of grip, touch, or force application
  c) EXECUTION: The core motion (rotation, translation, deformation, release)
  d) CONTINUATION/TRAJECTORY: Sustained motion showing direction, speed, manner
  e) RESULT/COMPLETION: End state confirming the action occurred (or did NOT occur for "pretending")
  f) DISAMBIGUATION: Any segment that distinguishes this action from a similar one
     - "pushing left to right" vs "pushing right to left" -> need enough frames to confirm direction
     - "throwing" vs "dropping" -> need the release moment
```

```
    - "pretending to X" vs actually doing X -> need to see the non-completion

MINIMUM SEGMENT GUIDANCE:
- Simple static displays ("showing X behind Y"): 1-2 segments
- Single discrete motions ("picking X up", "turning X over"): 2-3 segments
- Two-phase actions ("putting X into Y", "dropping X"): 3-4 segments
- Sustained/continuous actions ("spinning", "rolling"): 3-5 segments
- Cause-and-effect chains ("pushing X so it falls"): 4-5 segments
- Intentional non-completion ("pretending to X"): 4-6 segments (need context showing the absence)

Err toward INCLUDING a segment if you're unsure -- it's worse to drop a segment that matters than to keep one
    that doesn't.

SCORING EACH SEGMENT (0-100):
- 80-100: Removing this segment would make the action unrecognizable or ambiguous
- 50-79: This segment adds meaningful evidence (shows a distinct phase or disambiguates)
- 20-49: Mildly helpful but redundant with adjacent segments
- 0-19: Filler, static pause, or no relevant content

OUTPUT (JSON only, no markdown, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "action_summary": "<what you see happening in the video, max 30 words>",
  "segments": [
    {{
      "segment_id": <1-based index>,
      "time_range": "<start_s>-<end_s>",
      "importance": <0-100>,
      "phase": "setup" | "contact" | "execution" | "continuation" | "result" | "disambiguation",
      "reason": "<max 10 words>"
    }}
  ],
  "minimum_sufficient_set": [<list of segment_ids scoring >= 50>],
  "rationale": "<why these segments are needed together to recognize the action, 2-3 sentences>"
}}

Only include segments scoring >= 20 in the "segments" array (omit pure filler).
```

Kinetics-400 (`mss_verification_k400`).

```
---
name: mss_verification_k400
version: "3.0"
description: Verifies whether a video contains sufficient visual evidence for a Kinetics-400 action class.
    Requires visible action execution, not just scene recognition.
variables:
  - action_label: The K400 action class (e.g., 'playing basketball', 'swimming')
---

You are verifying whether a video shows a specific action from the Kinetics-400 dataset.

ACTION CLASS: {action_label}

ABOUT KINETICS-400:
This dataset contains 400 human action classes covering:
- Sports (playing basketball, swimming, skateboarding)
- Daily activities (cooking, cleaning, eating)
- Interactions (shaking hands, hugging, fighting)
- Instrument playing (playing guitar, playing piano)
- Dance and movement (dancing, yoga, stretching)

Actions are typically 10 seconds long and focus on the MAIN activity being performed.

TASK: Determine if this video shows someone performing "{action_label}".
```

```
CRITICAL REQUIREMENT -- VISIBLE ACTION:
You must SEE the action happening in the video frames. Recognizing a scene or setting where the action COULD
     happen is NOT sufficient. The action itself must be visible.

Ask yourself: "Can I see someone DOING '{action_label}', or do I just see a setting where it might happen?"
- A basketball court with no players moving is NOT "playing basketball"
- A kitchen with no cooking activity is NOT "cooking"
- A guitar visible but not being played is NOT "playing guitar"

KEY JUDGMENT CRITERIA:
1. MAIN ACTION identification:
   - What is the PRIMARY activity VISIBLE in the video?
   - Ignore brief/incidental actions, focus on the dominant activity
   - The person performing the action must be visible AND actively doing it

2. ACTION EXECUTION (not just scene):
   - The defining motion/activity of "{action_label}" must be VISIBLE across frames
   - Seeing just a setting, equipment, or result state is NOT sufficient
   - "playing basketball" = you must see dribbling, shooting, passing -- not just a court
   - "cooking" = you must see food preparation happening -- not just a kitchen

3. REASONABLE MATCHING:
   - Accept natural variations of the action class
   - "playing basketball" = dribbling, shooting, passing a basketball
   - "cooking" = chopping, stirring, frying in kitchen context

4. MASKED OR PARTIAL VIDEO:
   - If parts of the video are masked, blurred, or removed, judge from the VISIBLE portions
   - But the action itself must still be recognizable in the remaining frames
   - If the remaining frames show only the setting/scene without the action happening, choose NO

CONFIDENCE SCALE (only applies when decision is YES):
- 0.8-1.0: Action is clearly and unambiguously visible -- you can see it happening
- 0.5-0.8: Action is recognizable but some details are obscured or uncertain
- 0.3-0.5: Action appears to be happening but evidence is limited or partially visible

DECISION:
- YES: The action "{action_label}" is VISIBLY being performed -- you can see the motion/activity happening in
     the video frames.
- NO: The action is NOT visible -- you see only a scene/setting, only the result, or a DIFFERENT action
     entirely.
- SKIP: Video is corrupted, completely black, entirely unrelated content, or so degraded that NO action of ANY
      kind can be identified. SKIP is a last resort.

OUTPUT (JSON only, no other text):
{{
  "decision": "YES" | "NO" | "SKIP",
  "confidence": <0.0-1.0>,
  "evidence": "<what action you observed happening, max 20 words>",
  "rationale": "<explain: can you see the action being performed? what motion/activity is visible? 2-3
       sentences>"
}}
```

Greedy pilot tables. Tables 17 and 18 give the full pilot data behind Section 4.6: pairwise stability across four independent greedy runs of the same prompt on the same 101 SSv2 videos, and each greedy arm paired against the production direct-scoring run on the same sample.

A.3 Oracle response schema and parsed example

Each direct-scoring call returns a single JSON object. The parser expects the following top-level fields:

- `decision`: one of `YES`/`NO`/`SKIP`, the video-level precheck verdict (is the labeled action visibly

Table 17: Intra-greedy stability across four independent greedy runs (same prompt, same 101 SSv2 videos, identical parameters). Jaccard is computed on per-video kept sets (videos where both arms keep nothing are excluded); Spearman on per-segment weights aligned by index, averaged per video, using sort-order ranks *without* tie correction; on the greedy arms' coarse weight values a tie-corrected Spearman is substantially lower (≈ 0.17), so the two conventions are not interchangeable. Exact recomputation: `scripts/mode_prompt_agreement.py`.

Pair	Mean Jaccard	Mean Spearman
Arm A ↔ Arm B	0.149	0.661
Arm A ↔ Arm C	0.212	0.712
Arm A ↔ Arm D	0.075	0.742
Arm B ↔ Arm C	0.179	0.666
Arm B ↔ Arm D	0.201	0.748
Arm C ↔ Arm D	0.062	0.738
Mean intra-greedy	**0.146**	**0.711**

Table 18: Each greedy arm paired against the production direct-scoring run on the same videos. "kept" is mean kept segments per video (greedy / direct); "calls" is mean oracle calls per video.

Arm	n	Jaccard	Spearman	kept (G/D)	calls (G/D)	speedup
A (Feb 03)	94	0.173	−0.012	1.67 / 4.42	20.9 / 1.0	20.7×
B (Feb 10)	97	0.133	0.003	0.82 / 4.42	31.9 / 1.0	31.6×
C (Feb 11)	94	0.190	−0.012	1.61 / 4.42	27.8 / 1.0	27.6×
D (Feb 12)	96	0.081	−0.021	0.55 / 4.42	20.4 / 1.0	20.2×
Pooled	381	**0.144**	−0.010	1.17 / 4.42	∼25 / 1.0	∼ 25×

present and scorable?).

- `confidence`: the oracle's self-reported confidence in $[0, 1]$.
- `action_summary`: a free-text description of the observed action.
- `segments`: a list, one entry per fixed-duration segment, each with `segment_id`, `time_range`, `importance` (0–100), `phase` (one of setup/contact/execution/result), and a short `reason`.
- `minimum_sufficient_set`: the list of `segment_id`s the oracle judges jointly sufficient to identify the action.
- `rationale`: a free-text justification for the chosen set.

The release pipeline derives a per-segment continuous `weight` (`importance`/100) and a binary `label` (`important` if the segment is in `minimum_sufficient_set`, else `unimportant`) from this object, and stores the raw two-stage logits separately (`logit_yes`, `logit_no`, `logit_skip`, plus the derived `logit_confidence` and `logit_p_skip`). A representative parsed response, abbreviated to three of seven segments, is shown below for the Diving-48 clip `8qRmKunCjtY_00000` ("backward dive (no somersault) in pike position"):

```
{
  "decision": "YES",
  "confidence": 0.95,
```

```
  "action_summary": "Backward dive with no somersaults, pike position throughout flight, feet-first entry.
      Takeoff group is backward, body position clearly pike, no twists observed.",
  "segments": [
    {"segment_id": 1, "time_range": "0.0-0.5", "importance": 30, "phase": "setup", "reason": "Diver standing
        on board, preparing for backward takeoff"},
    ...,
    {"segment_id": 3, "time_range": "1.0-1.5", "importance": 95, "phase": "execution", "reason": "Peak flight,
         pike position clearly visible, no somersaults"},
    ...,
    {"segment_id": 7, "time_range": "3.0-3.5", "importance": 15, "phase": "result", "reason": "Water surface
        only, no diver visible"}
  ],
  "minimum_sufficient_set": [2, 3, 4, 5],
  "rationale": "Segment 2 confirms backward takeoff group. Segments 3-4 show pike body position with no
      somersaults during flight. Segment 5 confirms feet-first entry, validating pike position. Together,
      these segments uniquely identify the dive class."
}
```


For this call the recorded two-stage logits were `logit_yes` = 42.75, `logit_no` = 35.5, `logit_skip` = 21.5, giving $P(\text{SKIP}) \approx 0$ and a binary $P(\text{YES} \mid \neg\text{SKIP}) = 0.9993$, so the video cleanly passes precheck.

A.4 Oracle backends and model identifiers

Five interchangeable backends implement the shared `OracleInterface`. Each accepts the same prompt and returns the same parsed schema; they differ only in how logits are recovered (raw Hugging Face logits versus API logprobs). Table 19 lists the backends and the four canonical model identities.

Table 19: Oracle backends and the four canonical model identities used in this work. The production corpus uses Qwen3-VL-32B; the other three are cross-oracle comparators.

Backend module	Transport	Models / identifiers
`qwen_oracle.py`	local HF	Qwen3-VL-32B (`Qwen/Qwen3-VL-32B-Instruct`)
`internvl3_oracle.py`	local HF (slurm)	InternVL3-38B (`OpenGVLab/InternVL3-38B`)
`gemini_oracle.py`	Google GenAI	Gemini 3.1 Pro (also via Dartmouth ChatAPI)
`dartmouth_oracle.py`	Dartmouth API	Gemini family (remote, no GPU)
`azure_oracle.py`	Azure ChatAPI	GPT-5.5

B Corpus Construction, Quality Metadata, and Release Format

This appendix expands the corpus summary in the Methodology with full-corpus population counts, the precheck-decomposed annotation outcomes, the released schema, and source-video licensing.

B.1 Corpus populations

Across SSv2, K400, and Diving-48, the production pipeline made 536,181 source-video annotation attempts and produced 499,299 precheck-passed videos. These yield 4,576,082 per-segment importance scores on precheck-passed videos, and 4,912,308 total segment-scoring records when precheck-failed sidecars are included. Per-dataset counts appear in the Methodology annotation-outcomes table (Table 3).

B.2 Annotation outcomes with precheck decomposition

Table 20 decomposes the failed-annotation population into three categories: parse failures (malformed JSON), precheck-NO (the oracle judges the labeled action not visibly present), and precheck-SKIP (the oracle declines as ambiguous or corrupted). Parse failures are absent across all three datasets because Qwen3-VL-32B's structured JSON output is highly reliable; the failed population is therefore almost entirely precheck-NO, with a small precheck-SKIP tail.

Table 20: Annotation outcomes by failure category, full-corpus measured. The parse-fail column is zero everywhere; precheck-NO dominates the failed population, and precheck-SKIP is a small tail.

Dataset	Parse-fail	Precheck-NO	Precheck-SKIP
SSv2	0	10,715	290
K400	0	23,264	6
Diving-48	0	2,577	30

B.3 Precheck failure exemplars

The two failure modes are qualitatively distinct. A precheck-NO verdict records `decision=NO` when the oracle cannot see the labeled action in the video at all; the stored `evidence` field then describes what is visible instead, the `raw_output` snippet captures the model's reasoning, and the two-stage logits show $P(\text{YES} \mid \neg\text{SKIP})$ below 0.5. A precheck-SKIP verdict records `decision=SKIP` when the clip is too corrupted, ambiguous, or short to score, and is driven by a dominant $P(\text{SKIP})$ in the three-way softmax. Representative precheck-failed sidecars for each dataset are retained in the corpus under the production run directories and carry the full `evidence`, `raw_output`, and logit fields for inspection; they are excluded from all headline evaluation pools (see Limitations, "Dataset-specific precheck failures and coverage bias"). Figure 15 shows three representative precheck-NO videos; for each, the oracle's verbatim `evidence` field explains the rejection:

- K400 `Ne5V4b5hxEk_000040_000050`, labeled *driving tractor* ($P(\text{YES} \mid \neg\text{SKIP}) = 0.000$): "The video shows close-up views of a red tractor engine and parts, including hoses, valves, and a black grille with '562' label. No person is visible, and no driving motion is shown."
- K400 `jKB-yb7NUrI_000021_000031`, labeled *trimming or shaving beard* ($P(\text{YES} \mid \neg\text{SKIP}) = 0.000$): "Band performing on stage; guitarist playing red electric guitar, drummer visible, singer with microphone. No beard trimming or shaving observed."
- Diving-48 `3PLiUG_DuC8_00204`, labeled *forward 2.5 somersaults with 2 twists in pike position* ($P(\text{YES} \mid \neg\text{SKIP}) = 0.000$, $P(\text{SKIP}) = 0.003$): "Video shows only the pool ceiling and water surface; no diver is visible performing a dive. No takeoff, flight, or entry phases are observable."

B.4 Released schema

The unit of release is one canonical JSON sidecar per source video, published at `https://huggingface.co/datasets/ckeith/meaningful-moments` (tag `v1.0`). Top-level fields: `video_id`, `action_label`, `video_path` (substrate-relative), `timestamp`, `summary` (the oracle's one-paragraph description), `label_counts` (per-video important/unimportant tallies), `segments`, `mss_result`, and `elapsed_s`, plus substrate metadata where the source provides it (SSv2 `template` and

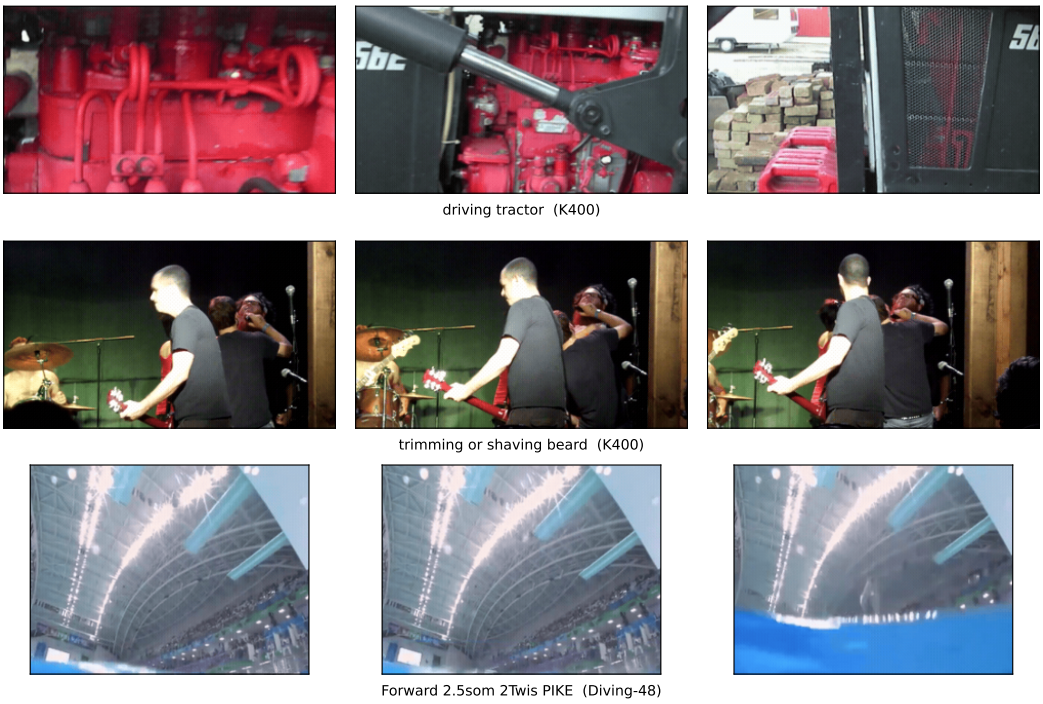

Figure 15: Representative precheck-failed videos, three stills each. Top: K400 *driving tractor*. Middle: K400 *trimming or shaving beard*. Bottom: Diving-48 *Forward 2.5som 2Twis PIKE*. In each case the labeled action is not visibly identifiable, so the video fails precheck and is excluded from the corpus and the evaluation pools.

`placeholders`; Diving-48 `class_id` and `raw_class_name`). Each entry of `segments` carries `index` (zero-based), `time_range`, `start_s`, `end_s`, `label`, `weight`, `phase`, and `reason`. `mss_result` carries `kept_indices`, `total_oracle_calls`, `precheck_passed`, `precheck_vote_yes`, and the full `precheck_responses` (`decision`, `confidence`, `evidence`, `rationale`, verbatim `raw_output`, the raw logits `logit_yes`/`logit_no`/`logit_skip`, and the derived `logit_confidence` $= P(\text{YES} \mid \neg\text{SKIP})$ and `logit_p_skip` $= P(\text{SKIP})$). The sidecars are packaged as deterministic tar shards; parallel Parquet tables (one configuration per dataset, one split per source split) mirror the sidecars field-for-field except for the verbatim `raw_output` string, with the `dataset` and `split` identities realized as the Parquet configuration and split names. The oracle identity and prompt version are pinned once per annotation run in a released `config.json` (model revision hash, prompt SHA-256). One internal field is dropped everywhere: `frequency`, a duplicate of `weight` in direct-scoring mode that additionally carries stale pre-recovery values in part of the cross-oracle supplement.

B.5 Release pipeline stored-field manifest

For implementers, the release pipeline (`scripts/hf_release/` in the code repository) writes exactly the fields of Section B.4. The per-segment `phase`, `reason`, and `time_range` are recovered from the oracle's verbatim `raw_output` at release time via the one-based `segment_id` join, with the join asserted per segment by the invariant `weight` = `importance`/100; videos whose raw output cannot be parsed (almost exclusively precheck-failed extraction-error records) degrade to null `phase`/`reason` and remain in the release as failure records. Recovery exceeds 99.95% of precheck-passed videos on every production run. The two pipeline-specific conventions worth restating are that `frequency`

is never written to the release (it equals `weight` in direct-scoring mode), and that `video_path` is rewritten to a substrate-relative path so no local filesystem layout is released while the `video_id` join to source videos stays trivial.

B.6 Source-video licensing and access

The Meaningful Moments labels are released under CC-BY-4.0, but the source videos are not redistributed and remain under their original terms: SSv2 under CC-BY-NC, K400 under CC-BY-4.0, and Diving-48 as research-only. Downstream users obtain the videos separately under each dataset's terms and join them to the released labels by `video_id`.

The release is hosted at `https://huggingface.co/datasets/ckeith/meaningful-moments` and tagged `v1.0`; loading with `revision="v1.0"` pins the exact published bytes. Alongside the Parquet tables and sidecar shards, the repository carries per-run `config.json` provenance (model revision, prompt SHA-256), the pinned evaluation-pool CSVs of Appendix C.1, the four scoring prompts verbatim, a Croissant metadata file (`croissant.json`), a SHA-256 manifest (`sha256sums.txt`) covering every sidecar and released file, and the four-oracle agreement supplement (`supplement/cross_oracle/`, 600-video pilot pool per oracle). Future re-extractions will be released as sibling subsets under new semver tags; existing labels are never overwritten. Issues are tracked on the dataset repository's discussion page.

C Evaluation Protocol Details

This appendix gives the evaluation pools, sampling grids, frame-allocation rule, and two protocol clarifications referenced from the main text.

C.1 Evaluation pools

The headline recognizer-eval pools are pinned and stratified per class at seed 42:

- SSv2 `eval_2k_stratified.csv`: $n = 2080$, 174 classes, cap 12 per class, restricted to precheck-passed videos.
- K400 `eval_2k_stratified.csv`: $n = 2000$, 400 classes, cap 5 per class.
- Diving-48 `eval_full.csv`: $n = 1970$, 47 classes (one of the 48 official classes has zero validation videos); no pinned `.meta.json` sidecar.
- Insertion/deletion subsets `id_500_stratified.csv`: $n = 500$ each, seed 42, sub-sampled from the corresponding eval pool.

C.2 Alpha grid and endpoint conventions

The production α-sweep uses the grid $\{0.05, 0.10, 0.20, 0.25, 0.30, 0.40, 0.50, 0.60, 0.70, 0.75, 0.80, 0.90\}$ (stored as percentages in `FASTFORWARD_ALPHA_PERCENTS` in `selectors.py`). The endpoints $\alpha = 0$ and $\alpha = 1$ are deliberately not in this grid because they use different inference paths. The $\alpha = 1$ endpoint is reported via `full` (uniform sampling across the original timeline). The $\alpha = 0$ endpoint is reported via `vlm-selected` (Keep-Important) and `lowest-evidence` (Keep-Filler), which build a cut-and-concatenated kept-set clip and resample from it rather than following the density-rule path.

C.3 Frame allocation pseudocode

Under the density rule, each segment k is assigned a density $d_k = 1$ if $\texttt{label}_k = \texttt{important}$ else α, and an effective weight $w_k = (\texttt{end\_s}_k - \texttt{start\_s}_k) \cdot d_k$. Frames are then allocated in proportion to w_k using largest-remainder rounding so that the total kept frames equal the recognizer's `frames_per_clip`:

```
def allocate_frames(segments, alpha, frames_per_clip):
    d = [1.0 if s.label == "important" else alpha for s in segments]
    w = [(s.end_s - s.start_s) * dk for s, dk in zip(segments, d)]
    raw = [frames_per_clip * wk / sum(w) for wk in w] # fractional
    alloc = [round(r) for r in raw]
    # repair rounding so the total is exactly frames_per_clip: hand the
    # residual to the largest fractional remainders (smallest, when
    # frames must be removed)
    diff = frames_per_clip - sum(alloc)
    order = argsort([r - a for r, a in zip(raw, alloc)], desc=(diff > 0))
    for i in range(abs(diff)):
        alloc[order[i]] += 1 if diff > 0 else -1
    ensure_min_one_frame(alloc) # every non-empty segment keeps a frame
    return alloc # frames per segment
```

Within each segment the allocated frames are sampled uniformly from that segment's decoded frames, and the per-segment frame lists are concatenated in time order. The reference implementation is `adapter.py:_decode_weighted_kept_frames_pyav`.

C.4 Threshold and budget sweep grids

The score-threshold sweep uses $T \in \{0, 10, 20, \ldots, 100\}$ and keeps segments with $\texttt{weight} \geq T/100$. The budget sweep uses $f \in \{10, 20, \ldots, 90\}$, ranks segments by descending weight, and accumulates the highest-weight segments until the retained duration reaches $(f/100) \cdot$ (total pool duration). Both grids are defined in `selectors.py` (`SCORE_THRESHOLD_PERCENTS` and `ID_FRACTION_PERCENTS`, stored as percentages).

C.5 Insertion/deletion curve grid

The insertion/deletion analysis sweeps the retained fraction over $\{10, 20, \ldots, 90\}$ for five orderings: `vlm` (weight descending), `anti-vlm` (ascending), `random`, `temporal` (chronological), and `motion` (cached optical-flow magnitude descending). The random ordering is deterministic per video, seeded by $\texttt{sha256}(\texttt{run\_seed} = 42 \,\|\, \texttt{video\_id} \,\|\, \texttt{condition})$. Per-video trapezoidal AUC on top-1 and top-5 is integrated over the swept fractions plus the common endpoints, and ΔAUC against the random ordering is tested with a paired bootstrap ($B = 10{,}000$) on the $n = 500$ stratified subset per dataset.

C.6 SSv2 step=2 vs step=4 protocol clarification

The V-JEPA 2 paper protocol uses a clip span of 16 frames at step 4, i.e. 64 source frames per clip. Most SSv2 videos are shorter than this, which causes the multi-clip evaluator to collapse its clips into overlapping windows; interior α values then produce near-identical inputs and the α axis cannot move. The production α-sweep therefore uses step 2 (32 source frames per clip), which lets the axis vary. The step-4 paper protocol is reported only as a sanity check at the $\alpha = 0$ endpoint, where clip collapse does not matter.

C.7 Diving-48 id_500 metadata-label bug

The sidecar `data/csvs/diving48/id_500_stratified.csv.meta.json` records `dataset: ssv2`, a label bug originating in the `DATASET_DEFAULTS` table of `scripts/sample_id_subset.py`. The

pool itself is correctly stratified from the Diving-48 `eval_full.csv` (verified by inspecting the row `video_id`s); only the metadata tag is wrong. This is cosmetic and is slated to be corrected before release.

D Statistical Testing Details

The statistical procedures are anchored against `oracle/scripts/eval_paired_stats.py` and `scripts/recompute_exp5_cis.py`.

D.1 Paired bootstrap and confidence intervals

Headline contrasts (Table 7 and the headline-CI table) use $B = 10{,}000$ bootstrap resamples via `scripts/recompute_exp5_cis.py`; the auxiliary paired-statistics reports throughout the appendix use the default $B = 1000$. Intervals are percentile bootstrap intervals (`np.percentile(boot_means, [2.5, 97.5])`) and are reported as raw 95% intervals, not Bonferroni-adjusted.

D.2 McNemar test

Paired accuracy contrasts use a hand-implemented McNemar test with Yates continuity correction. With b_{10} and b_{01} the discordant-pair counts,

$$\chi^2 = \frac{\left(|b_{10} - b_{01}| - 1\right)^2}{b_{10} + b_{01}}, \qquad \mathrm{df} = 1,$$

and the p-value is computed as $\mathrm{erfc}\left(\sqrt{\chi^2/2}\right)$. This is the continuity-corrected (not exact-binomial) form, and the prose refers to it as "McNemar's test with Yates continuity correction".

D.3 Multiple-comparison correction

Bonferroni correction is applied to the headline family of $k = 8$ primary contrasts, giving a family-wise threshold $\alpha = 0.05/8 = 0.00625$. The correction governs significance interpretation only; the reported confidence intervals remain raw 95% intervals. A Holm step-down procedure is used only for the auxiliary consistency contrasts in the cross-oracle tables (`cross_oracle/consistency_tables.py`), and is not the correction used for the headline results.

D.4 Matched-video subset logic

Every paired contrast (condition a versus condition b) drops any video for which either condition's selector failed (`selector_failed=True` in the sidecar), so the matched n for a contrast is at most the eval-pool n. For example, the K400 `vlm-selected` versus `full` contrast on the $n = 2000$ pool has a matched $n_{\mathrm{paired}} = 1979$ after dropping 21 selector failures.

E Auxiliary Results and Diagnostic Probes

These diagnostics do not sit on the main contribution list but inform the central story. They are framed as decompositions and checks, not as recognizer-improvement claims.

E.1 Selection-aware adaptation under frozen encoders

This probe asks how much of the off-the-shelf `vlm-selected` deficit is domain shift (closeable by adapting to the MM-cut input distribution) versus irrecoverable information loss (the residual after adaptation). A linear head is trained on a frozen encoder and frozen attentive pooler over MM-cut inputs, matching the eval input distribution, so any residual gap isolates information loss from domain shift. On Diving-48, `vlm-selected` top-1 rises from 42.18% off-the-shelf to 74.77% after adaptation (+32.6 pp), and `vlm-selected` beats `full` within both arms (+4.77 pp off-the-shelf, +3.70 pp adapted; within-recognizer comparisons, not absolute state-of-the-art claims). On SSv2, adaptation roughly halves the kept-set deficit, from −3.32 pp off-the-shelf to −1.58 pp adapted. The reading differs by dataset: the large Diving-48 drop at the MM-cut input distribution is almost entirely domain shift the head absorbs (there is no deficit against `full` in either arm), while on SSv2 about half the kept-set deficit is domain shift, leaving a smaller irrecoverable residual.

E.2 SSv2 closed-set classification check

As an independent check that does not involve the recognizer head, Qwen3-VL-32B itself performs closed-set classification over the 174 SSv2 templates, comparing the full video against the MSS-cut on $n = 25{,}647$ SSv2 test videos. Top-1 is 17.18% for the full video versus 14.70% for the MSS-cut, with 62.86% paired-prediction agreement. From the oracle's own perspective the MSS-cut preserves roughly 85% of full-video top-1. Data: `pseudo_labels/classify_ssv2/eval_20260423_005053/`.

E.3 Per-class diagnostics and SSv2 per-template breakdown

A per-class delta breakdown shows which classes MM helps or hurts on each dataset (Figure 16). For SSv2 specifically, the closed-set classification deficit concentrates in boundary-state-discriminated templates (where the discriminative evidence is the before/after state at the clip boundaries, which the cut-and-concatenate repack disturbs), while motion-discriminated templates can gain.

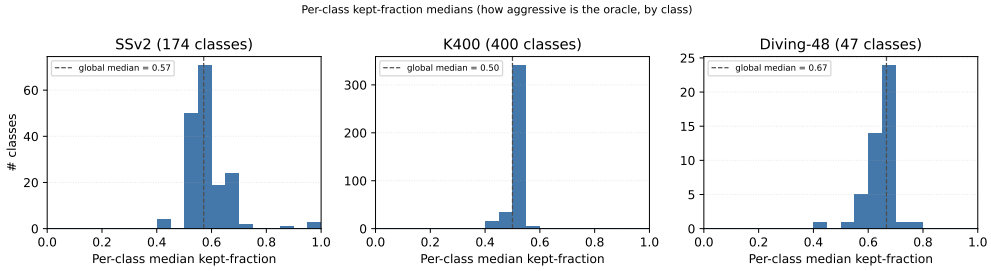

Figure 16: Per-class kept-fraction distribution. Classes vary widely in how much of their video MM retains, which tracks the per-class recognition deltas.

Table 21 gives the extremes of the SSv2 per-template closed-set breakdown referenced in the main text. Over the 172 templates with at least 10 videos, the median per-template delta is 0 pp, 116 templates (67%) sit within ±2 pp of full, 26 (15%) lose more than 5 pp, and 4 (2%) gain more than 5 pp. The worst losses are boundary-state-discriminated templates; the largest gains are motion-discriminated templates whose evidence concentrates mid-action.

Table 21: Largest per-template losses and gains under MSS-cut closed-set classification (SSv2 test, templates with $n \geq 10$). Positive $\Delta_{\text{top-1}}$ means full beats MSS-cut.

Template	n	Full %	MSS-cut %	$\Delta_{\text{top-1}}$ (pp)
Largest losses (MSS-cut hurts)				
Removing something, revealing something behind	77	36.36	15.58	+20.78
Dropping something next to something	106	36.79	16.04	+20.75
Putting something on a surface	492	51.83	31.10	+20.73
Laying something on the table on its side, not upright	145	35.86	16.55	+19.31
Closing something	184	45.65	27.17	+18.48
Largest gains (MSS-cut helps)				
Something falling like a feather or paper	358	13.97	22.35	−8.38
Tearing something into two pieces	581	36.32	41.65	−5.34
Pouring something out of something	119	31.09	35.29	−4.20
Pouring something onto something	78	14.10	17.95	−3.85
Throwing something against something	88	1.14	4.55	−3.41

E.4 Temporal coverage and clustering diagnostics

For each (video, condition) the analysis computes the kept-set range, coverage ratio, and Gini coefficient of the inter-segment gaps. The diagnostic question is whether clustered (low-coverage) MM selections explain the kept-set deficit. They do not: the per-video correlation between the coverage gap (`uniform` minus `vlm-selected`) and the recognition gap has $|r| < 0.1$ on SSv2, so coverage geometry alone does not account for the deficit. Output: `pseudo_labels/classifier_eval/temporal_coverage_analysis/`.

E.5 Extended threshold and budget tables

Tables 22–24 give the full decile sweeps behind the $\alpha = 0$ continuous-score results. Each table pairs the score-threshold axis (T, keep `weight` $\geq T/100$) with the budget axis (f, keep the top-f% of duration by weight). The headline operating points quoted in the main text are the bolded top-1 cells; bold marks each column's maximum.

E.6 Qualitative gallery

Figures 17–22 show representative wins and losses per dataset: clips where MM keeps the class-discriminative moments (wins) and clips where it discards information the recognizer needs (losses). Thumbnails are color-coded green for kept segments and red for dropped segments.

F Cross-Oracle Validation Details

This appendix expands the cross-oracle headline numbers: the four-oracle identity, the joint-precheck derivation, pairwise segment-level agreement, and the downstream-recognition consistency that is the load-bearing check.

Table 22: Diving-48 score-threshold (left) and budget (right) sweeps, paper-checkpoint 4×3, $n = 1970$. Reference: `full` 87.82 / 99.09; `vlm-selected` 82.08 / 97.41; `lowest-evidence` 52.13 / 83.76.

T	Top-1 (%)	Top-5 (%)
0	**88.58**	**98.73**
10	88.38	98.58
20	87.31	98.63
30	86.95	98.43
40	85.74	98.12
50	85.03	97.87
60	83.71	97.41
70	82.23	96.45
80	76.65	95.03
90	64.77	89.34
100	37.56	75.79

f (%)	Top-1 (%)	Top-5 (%)
10	41.52	78.32
20	57.11	87.21
30	68.98	93.10
40	76.60	97.36
50	82.34	98.48
60	84.77	98.83
70	85.99	98.88
80	87.66	**98.98**
90	**87.87**	98.88

Table 23: SSv2 score-threshold (left) and budget (right) sweeps, paper-checkpoint 2×3 step=2, $n = 2080$. Reference: `full` 65.77 / 92.98; `vlm-selected` 62.45 / 91.11; `lowest-evidence` 53.41 / 81.20. Both axes show an interior optimum.

T	Top-1 (%)	Top-5 (%)
0	59.86	88.94
10	59.90	88.89
20	60.10	88.94
30	62.45	91.01
40	63.32	91.25
50	**63.37**	91.06
60	62.84	91.49
70	62.36	**91.63**
80	57.88	87.07
90	43.99	73.56
100	35.67	65.58

f (%)	Top-1 (%)	Top-5 (%)
10	37.31	67.69
20	47.60	77.98
30	55.24	85.38
40	60.34	89.42
50	62.69	90.72
60	63.32	91.54
70	**63.65**	**91.68**
80	63.32	90.72
90	60.77	89.66

F.1 Final four oracles versus the earlier coverage pilot

The final four-oracle agreement study uses Qwen3-VL-32B, InternVL3-38B, Gemini 3.1 Pro, and GPT-5.5 (via Azure ChatAPI). An earlier coverage pilot ($n = 200$ per dataset) additionally evaluated InternVL3-78B. That 78B variant belongs to the earlier pilot only and is not part of the headline four-oracle agreement numbers; the distinction is called out here to avoid conflating the two studies.

F.2 Joint-precheck-pass derivation (n=369)

The shared evaluation set is derived by intersecting the sidecar `video_id` sets across all four runs and then dropping any video for which any run records `precheck_passed` as `False` (per `oracle_agreement_eval.py`). Of the 600-video pilot pool, this yields $n = 369$ jointly scorable videos.

Table 24: K400 score-threshold (left) and budget (right) sweeps, VideoMAE-L single clip, $n = 2000$. Reference: `full` 83.95 / 96.90; `vlm-selected` 85.40 / 97.55; `lowest-evidence` 80.15 / 94.15. Both axes peak above `full`.

T	Top-1 (%)	Top-5 (%)
0	83.95	96.90
10	84.00	96.90
20	84.15	96.90
30	84.40	97.20
40	84.65	97.40
50	84.30	97.60
60	85.00	97.55
70	**86.00**	**97.75**
80	84.85	97.40
90	82.20	96.70
100	81.95	96.60

f (%)	Top-1 (%)	Top-5 (%)
10	83.55	97.15
20	85.20	97.50
30	**85.75**	97.55
40	85.50	97.45
50	85.55	**97.75**
60	85.30	97.65
70	85.00	97.40
80	84.95	97.20
90	84.85	96.85

F.3 Pairwise agreement: Spearman, Jaccard, Set-F1

Table 25 reports all six pairwise agreements on the per-segment importance signal. Spearman is computed per video on aligned per-segment weight vectors then averaged; Jaccard and Set-F1 are computed on the kept set (`label=important`); keep-ratio MAE is the absolute deviation between each oracle's per-video kept fraction. The three closed-weights frontier models (Qwen, Gemini, GPT-5.5) cluster at Spearman 0.49–0.61, while InternVL3-38B sits apart at 0.35–0.37 against all three.

Table 25: Pairwise agreement on per-segment importance across all six oracle pairs, paired-bootstrap means over $n = 369$ joint-precheck-pass videos (Spearman n slightly lower, 311–366 per pair, due to per-pair empty-segment exclusions). 95% CIs: Spearman ±0.04–0.05, Jaccard ±0.02–0.03, Set-F1 ±0.02, keep-ratio MAE ±0.015–0.02. Source: `eval_complete_gemini_20260520_003221/`.

Pair	Spearman	Jaccard	Set F1	Keep-ratio MAE
Qwen as anchor				
Qwen ↔ GPT-5.5	0.584	0.620	0.744	0.170
Qwen ↔ Gemini	0.494	0.469	0.583	0.201
Qwen ↔ InternVL3-38B	0.370	0.555	0.680	0.130
Cross-comparator pairs				
Gemini ↔ GPT-5.5	0.610	0.515	0.624	0.257
InternVL3-38B ↔ GPT-5.5	0.359	0.586	0.717	0.175
InternVL3-38B ↔ Gemini	0.354	0.432	0.553	0.260

F.4 Downstream recognition consistency per oracle

The more load-bearing question is whether substituting a different oracle as the label source would change the downstream recognition story. Table 26 reports per-condition top-1 for each (oracle, dataset) cell on the shared pilot subset, holding the recognizer fixed within each substrate.

Figure 17: Diving-48 qualitative wins.

The `full` column is identical across oracles by construction. The qualitative ordering is robust: `vlm-selected` beats `uniform` and `lowest-evidence` on Diving-48 and K400 for every oracle, and the strongest `vlm-selected` sources (Qwen and Gemini) agree on every substrate, despite the moderate segment-level agreement of Table 25.

F.5 API logprob coverage

The two-stage precheck confidence depends on recovering token-level likelihoods, and the four backends differ in what they expose. GPT-5.5 uses the OpenAI-style `logprobs=True` request

Figure 18: Diving-48 qualitative losses.

with `top_logprobs=5` (the default `logprobs_top_k`) in `azure_oracle.py`, falling back automatically to text-only parsing if a deployment does not expose them. Gemini 3.1 Pro uses the SDK `logprobs_result` (chosen plus top candidates, `gemini_oracle.py`). Qwen3-VL-32B uses raw Hugging Face logits via `output_scores=True` (`qwen_oracle.py`). InternVL3-38B's chat interface does not surface per-token generation scores, so its `logit_yes/logit_no/logit_skip` fields are recorded as `None` and its `logit_confidence` falls back to the model's self-reported confidence (`internvl3_oracle.py`); this is a property of the backend, not a parsing failure. The per-segment importance scores that drive the agreement metrics come from each oracle's parsed JSON output for all four backends, so the agreement numbers do not depend on logprob availability.

Figure 19: Kinetics-400 qualitative wins. Four dropped-segment thumbnails in the fourth example are masked to black because the source video carries an overlaid expletive.

G Reproducibility Manifest

A single-page manifest for reproducing the corpus and the evaluations.

G.1 Repository state

The full annotation, evaluation, statistics, and plotting pipeline is released as a public code repository alongside the Meaningful Moments labels: `https://github.com/camkeith/meaningful-moments`

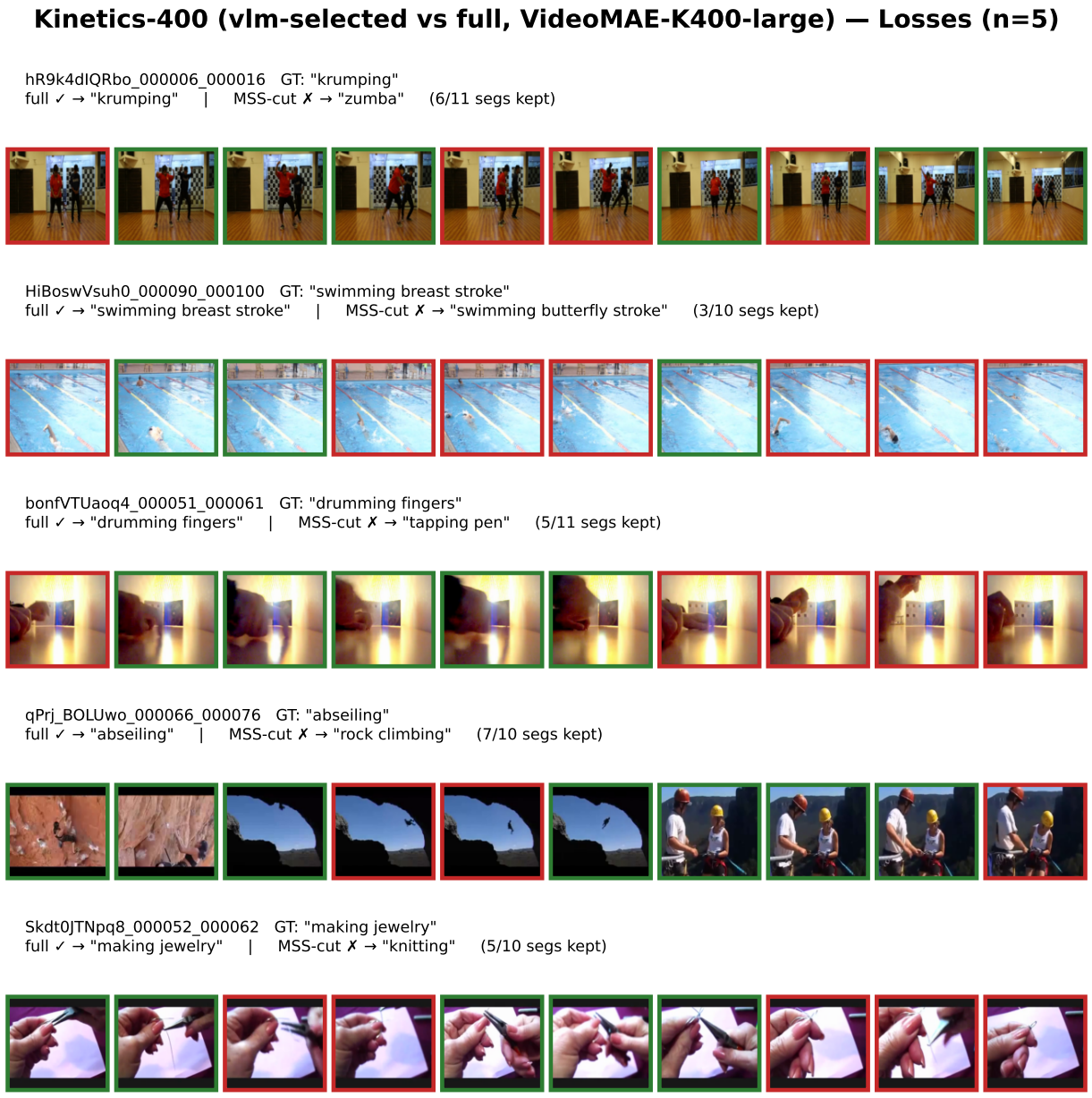

Figure 20: Kinetics-400 qualitative losses.

(MIT), tag `v1.0` at commit `d2448de`; the dataset card pins the same repository URL and tag. All code and data paths in this manifest are relative to the repository root.

G.2 Environment

The pipeline runs on Python 3.12.3 and ffmpeg 6.1.1; exact Python dependencies are pinned in `oracle/requirements.txt`. Key dependencies:

```
torch>=2.0
transformers>=4.40
accelerate>=0.26
```

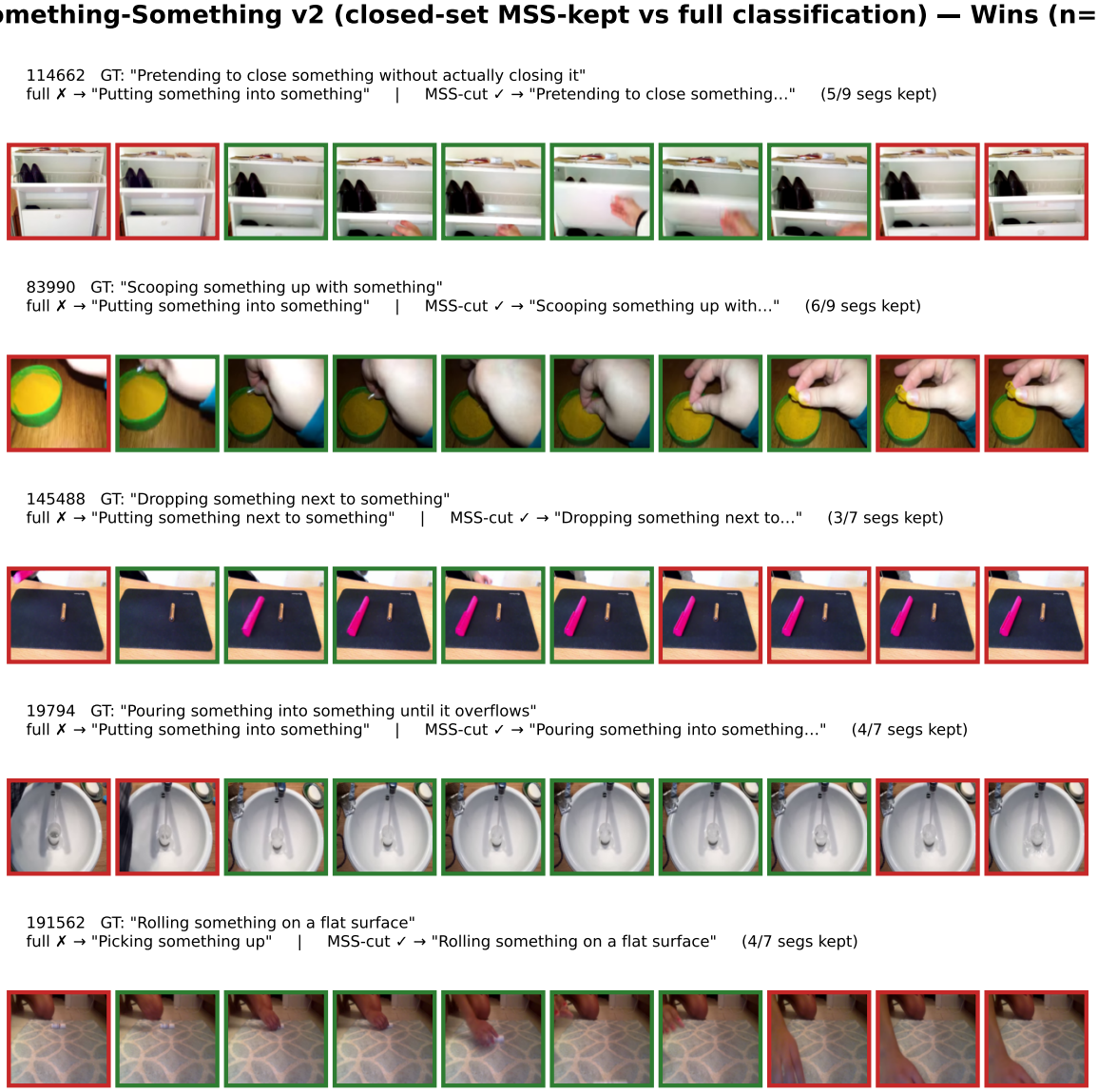

Figure 21: Something-Something v2 qualitative wins.

```
av>=10.0
qwen-vl-utils>=0.0.8
langchain-dartmouth>=0.3.0
google-genai>=0.3.0
```

G.3 Code paths

```
# label construction
oracle/scripts/mss_extract.py
oracle/scripts/mss/*.py # segments, masking, oracle backends,
                        # extraction, labeling, evaluation, helpers
oracle/scripts/mss/prompts/__init__.py # the prompt loader
oracle/scripts/{vjepa2,videomae,vjepa2_official,cross_oracle,
```

Figure 22: Something-Something v2 qualitative losses.

```
  classify}/__init__.py
oracle/scripts/mss/prompts/{direct_scoring,direct_scoring_ssv2,
  direct_scoring_k400,direct_scoring_diving48,mss_verification,
  mss_verification_ssv2,mss_verification_k400}.md

# evaluation
oracle/scripts/{run_classifier_eval,eval_paired_stats,eval_id_curves,
  shuffle_damage_analysis,temporal_coverage_analysis,
  oracle_agreement_eval,train_head,train_attentive_probe,
  classify_ssv2,classify_ssv2_eval}.py
oracle/scripts/vjepa2/{selectors,inference,adapter,head,cache_features,
  compute_motion}.py
oracle/scripts/videomae/{inference,head,cache_features,train_head}.py
oracle/scripts/vjepa2_official/{run_eval,wrapper,cache_features,
  cache_token_features,fix_ssv2_gt_ids}.py
oracle/scripts/cross_oracle/*.py
```

Table 26: Per-(oracle × dataset × condition) recognition top-1 on the shared pilot subset ($n \approx 200$ per cell). `full` is identical across oracles by construction. Bold marks the best `vlm-selected` top-1 per substrate. Absolute `full` values differ from the paper-checkpoint tables because this uses the HF-port recognizers at fixed protocol; the comparison that matters is across oracles at fixed recognizer.

Substrate	Oracle	full	vlm-sel	vlm-wgt	uniform	lowest-ev
SSv2	Qwen	68.50	63.50	54.00	62.50	49.75
	Gemini	68.50	**65.00**	63.00	67.50	51.24
	Azure GPT-5.5	68.50	64.00	62.00	58.71	52.50
	InternVL3-38B	68.50	56.50	50.00	63.00	50.75
K400	Qwen	80.50	82.50	75.50	77.00	79.10
	Gemini	80.50	**84.00**	82.50	79.50	79.60
	Azure GPT-5.5	80.50	76.50	71.00	70.50	67.66
	InternVL3-38B	80.50	83.50	73.50	78.00	79.60
D48	Qwen	37.50	40.50	28.50	24.50	12.94
	Gemini	37.50	**41.00**	37.50	25.87	12.00
	Azure GPT-5.5	37.50	37.50	15.00	13.50	10.45
	InternVL3-38B	37.50	37.50	35.00	33.50	35.32

```
oracle/scripts/classify/*.py
scripts/{recompute_exp5_cis,sample_eval_subsets,sample_id_subset,
  mode_prompt_agreement,qualitative_gallery,qualitative_gallery_ranking}.py

# statistics + plotting
scripts/distributions/extract_distributions.py
paper_tables_20260514_140732/plot_{alpha,threshold,id,class}*.py
paper_tables_20260514_140732/plot_per_class_kept.py
thesis/figures/make_thesis_figures.py
```

The paper-checkpoint evaluation path (`vjepa2_official/`) additionally requires Meta's V-JEPA 2 source tree, which is not redistributed: clone `facebookresearch/vjepa2` at the commit pinned in the code repository's `REPRODUCING.md` and point the `VJEPA2_REPO` environment variable at the clone.

G.4 Evaluation CSVs

- `data/csvs/ssv2/eval_2k_stratified.csv` (+ `.meta.json`).
- `data/csvs/k400/eval_2k_stratified.csv` (+ `.meta.json`).
- `data/csvs/diving48/eval_full.csv` (no `.meta.json`).
- `data/csvs/{ssv2,k400,diving48}/id_500_stratified.csv` (+ `.meta.json`; the D48 meta records `dataset=ssv2`, the label bug noted in Section C.7).
- `data/csvs/oracle_agreement/sample_600.csv` (+ `sample_600_manifest.json`): the cross-oracle pilot pool behind the $n = 369$ joint-precheck-pass analysis (an earlier `sample_400.csv` from the N-way pilot is superseded).
- `data/csvs/ssv2/val_sample.csv`: the 100-video SSv2 sample behind the mode/prompt ablations; the agreement tables are recomputable via `scripts/mode_prompt_agreement.py`.

G.5 Recognizer checkpoints

The headline recognizers load the following checkpoints; the two HF-port heads appear only in diagnostic runs.

```
external/vjepa2_checkpoints/ssv2-vitl-16x2x3.pt # V-JEPA 2, SSv2
external/vjepa2_checkpoints/diving48-vitl-256.pt # V-JEPA 2, Diving-48
external/vjepa2_checkpoints/vitl.pt # ViT-L encoder
MCG-NJU/videomae-large-finetuned-kinetics # K400 (Hugging Face)
facebook/vjepa2-vitl-fpc16-256-ssv2 # HF-port head (diagnostics)
facebook/vjepa2-vitl-fpc32-256-diving48 # HF-port head (diagnostics)
```

G.6 Oracle model identifiers

Production oracle (corpus): Qwen3-VL-32B-Instruct (`Qwen/Qwen3-VL-32B-Instruct`; technical report `arXiv:2511.21631`). Cross-oracle comparators: InternVL3-38B (`OpenGVLab/InternVL3-38B`), Gemini 3.1 Pro (Dartmouth ChatAPI), and GPT-5.5 (Azure ChatAPI).

G.7 Prompt versions

The corpus uses the direct-scoring prompts (`prompts/direct_scoring.md` and its `_ssv2`, `_k400`, and `_diving48` variants); the greedy prompts (`prompts/mss_verification.md` and its `_ssv2` and `_k400` variants) appear only in the mode ablations. Every sidecar records the `prompt_template_id` used, and the release additionally ships a content hash of each prompt for exact-version verification.

G.8 Script-to-table/figure map

Table 27 maps each plotting or analysis script to the artifact it produces.

Table 27: Which script produces which figure or table artifact.

Script	Artifact
`plot_alpha_curves.py`	`{d48,ssv2,k400}_curve.pdf`
`plot_threshold_curves.py`	`{d48,ssv2,k400}_threshold_budget.pdf`
`plot_id_curves.py`	`{ssv2,k400,d48}_id_curves.pdf`
`plot_class_coverage.py`	`fig_{class_coverage_hist, weight_distribution, source_mm_parity}.pdf` (input: the committed `scripts/distributions/dataset_figures_v2.json`)
`plot_per_class_kept.py`	`fig_per_class_kept.pdf` (from the release manifests)
(hand-authored diagram)	`fig_pipeline.pdf`
`recompute_exp5_cis.py`	headline paired-bootstrap CIs ($B = 10{,}000$)
`mode_prompt_agreement.py`	App. A.2 agreement tables (A.2/A.3) and the mode-vs-prompt decomposition
`distributions/extract_distributions.py`	`distributions/out/distributions.json`
`distributions/plot_temporal_profile.py`	`fig_temporal_profile.pdf` (binned data committed at `distributions/temporal_profile.json`)
`thesis/figures/make_thesis_figures.py`	`fig_{teaser, dataset_examples, alpha_sampling, greedy_walkthrough, precheck_examples}.pdf` (stills sourced from the defense deck; not in the code release)

G.9 Release version

Meaningful Moments v1.0 is the production direct-scoring corpus at the release commit. Future releases will be tagged with a semantic version together with the corresponding commit hash and the `prompt_version` content hash.