SPARSE VIDEO UNDERSTANDING WITHOUT THE HEURISTICS

Anonymous authors
Paper under double-blind review

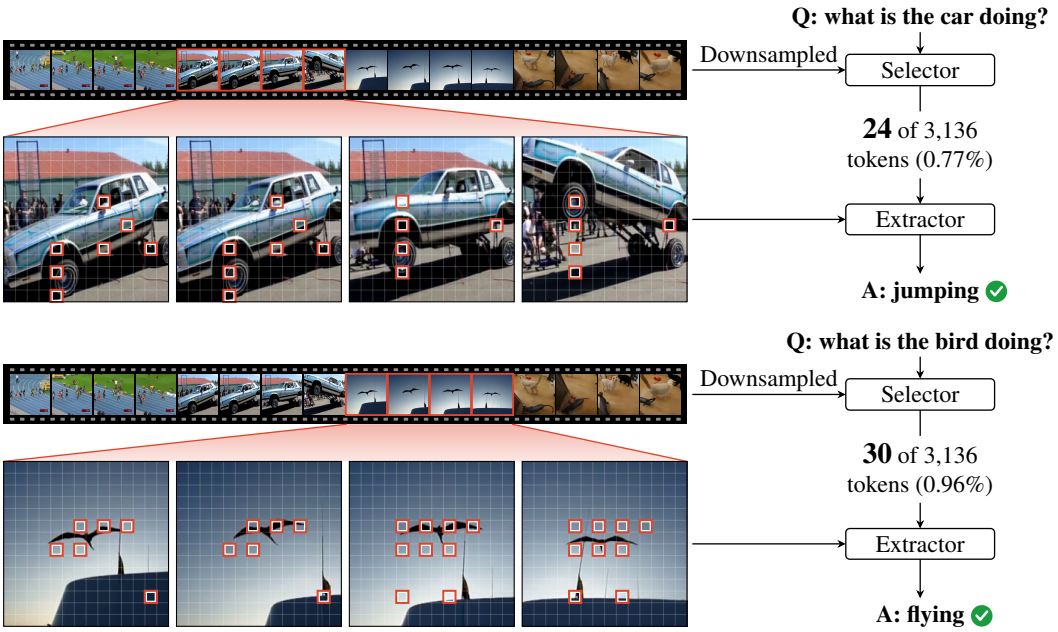


Figure 1: **Efficient video understanding through conditional token selection.** Each question only asks about part of the video. Our end-to-end method learns to sparsely select under 1% of the tokens (boxed in red) to correctly answer the questions. A deeper extractor answers from those tokens alone. Examples shown from the Actor-Action Dataset (Xu et al., 2015).

ABSTRACT

Video recognition often requires only a handful of tokens, but which tokens matter depends on the task. Existing approaches for sparse video computation rely on heuristics such as motion, diversity, or reconstruction to guide selection. These signals capture redundancy rather than task relevance, often retaining more tokens than necessary. We instead optimize selection directly for the task, jointly learning a lightweight selector that glimpses across space and time and a deeper extractor that processes only the selected tokens. Both are trained end-to-end through reinforcement learning, directly coupling the learning of selection with the performance of recognition. Across six action-recognition benchmarks (Kinetics-400, SSv2, Epic-Kitchens, Diving48, Jester, and Charades) and two text-conditioned tasks (A2D and CLEVRER), our approach pushes the Pareto frontier for accuracy-efficiency, outperforming state-of-the-art efficient video methods while generalizing across architectures (DINOv3, VideoMAEv2, and InternVideo2).

1 INTRODUCTION

Most video frames are redundant, with the same background, textures, and static regions persisting across frames. Yet information relevant to recognition is often localized in space and time, appearing only briefly within an otherwise continuous visual stream. Humans navigate this stream effortlessly, interpreting the scene from sparse, task-dependent glimpses. Transformers (Vaswani et al., 2017;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Dosovitskiy et al., 2021) need not process everything either. Their tokenized inputs naturally support sparse computation, offering a path to efficiency, but only if we know where and when to compute.

Such selection requires learning when and where to compute, while keeping the selection cheap for efficiency. The central challenge is that the "right" tokens are task-dependent and must be learned. Different objectives may require different evidence from the same visual stream, particularly for sparse computation, and discrete selection is not easily optimized end-to-end.

Prior work reduces tokens by pruning or merging progressively across layers (Choi et al., 2024; Li et al., 2024; Bergner et al., 2025), but pays the computational cost of estimating importance from intermediate representations that are themselves costly to compute. Alternative approaches select cheaply before processing deeply. To bypass the non-differentiability of hard selection, they often rely on heuristics (Choudhury et al., 2024; Hao et al., 2025; Salamatian et al., 2026; Shi et al., 2026) or coarser relaxations (Wang et al., 2022) that optimize only indirectly, and may fail to generalize across tasks or representations, retaining more tokens than necessary (Rammohan et al., 2026).

We address these challenges by moving beyond heuristics and directly optimizing which tokens to process for a given task. Our approach factorizes recognition into token selection followed by feature extraction, following prior work (Fuller et al., 2025; Salamatian et al., 2026). A lightweight selector glimpses across space and time, predicting *when* and *where* the essential tokens lie, after which a higher-capacity feature extractor operates only on the selected tokens. By selecting before any deep processing, we recover task-relevant features without ever computing the full input.

We tailor selection to a task with reinforcement learning (RL) (Sutton & Barto, 1998), jointly optimizing the selector and extractor for sparse approximation of dense, task-specific computation. Conditioned on the task, the selector learns which times and regions are worth processing, while the extractor learns to recover strong representations of the *full* input from only those selected glimpses. The two co-adapt through joint, end-to-end training with the extractor shaping the selector's notion of importance, and the selector, in turn, steering the extractor toward what is worth processing. The result is a model that is not only efficient, but specialized to the tokens the task requires.

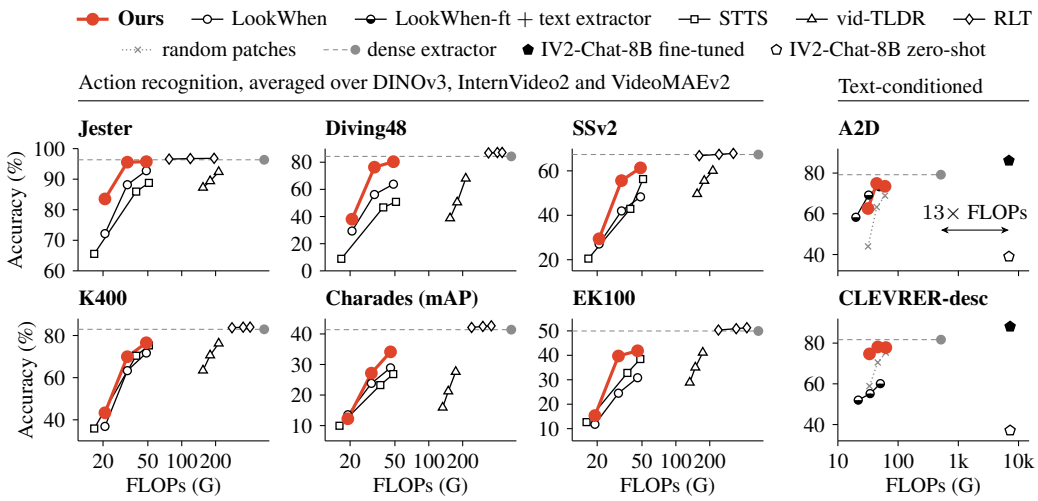
Our approach pushes the Pareto frontier for accuracy and efficiency (Fig. 2) while generalizing across tasks and models. We validate on standard benchmarks, including Kinetics-400 (K400) (Kay et al., 2017), Something-Something-v2 (SSv2) (Goyal et al., 2017), Epic-Kitchens (EK100) (Damen et al., 2020), Diving48 (Li et al., 2018), Jester (Materzynska et al., 2019), and Charades (Sigurdsson et al., 2016), showing that directly optimizing selection improves efficiency over dense video models, and outperforms state-of-the-art (SoTA) baselines for efficiency while remaining robust to representation. To further assess specialization, we fine-tune for text-conditioned selection on the Actor-Action Dataset (A2D) (Xu et al., 2015), and visual question answering (VQA) with CLEVRER (Yi et al., 2019), where selection and recognition are guided by text.

2 Related Work

We adopt token selection for video, but differ in how selection is learned. These methods choose their inputs to reduce computation, but must do so efficiently. Existing approaches for token selection have primarily relied on heuristics such as diversity (Park et al., 2022; Salamatian et al., 2026), motion (Hwang et al., 2024), and reconstruction (Shi et al., 2026). While effective at identifying redundancy, these signals are not task-specific and may fail to generalize across tasks or models.

Patch Selection. Earlier methods select patches by comparing in pixel space. EVEREST (Hwang et al., 2024) selects patches containing high-motion features by comparing consecutive patch embeddings and discarding those remaining similar over time. RLT (Choudhury et al., 2024) similarly compares temporally consecutive patches in pixel space to remove static regions, while K-centered (Park et al., 2022) maximizes diversity through pairwise patch comparisons. However, the low-level features provided by pixels do not necessarily predict task-relevant importance (Hao et al., 2025).

Learned Token Selection. Later approaches process tokens cheaply before selecting which patches to process more deeply, but still rely on heuristic supervision. AutoGaze (Shi et al., 2026) reduces redundancy through autoregressive selection to optimize reconstruction. LITE (Hao et al., 2025) learns a selector to predict token importance and shows that importance does not necessarily coincide with visual cues. It supervises selection using gradient-based attributions from a dense model, but



		Dense*	STTS	RLT†	vid-TLDR	LookWhen	Ours
FLOPs (G)	95% sparsity	540.1	37.1	294.1	148.9	30.9	**30.9**
Speed (clip/s)	95% sparsity	58.7	413.5	102.5	75.5	485.3	**485.3**
Memory (GB)	95% sparsity	0.57	0.46	0.42	2.05	0.53	**0.53**

*Our dense extractor, mean of the three backbones. †RLT at τ=0.1, mean over the six datasets.

Figure 2: **Pushing the Pareto frontier.** We plot accuracy against inference FLOPs for adaptive computation methods at varying sparsity, and report per-clip cost. Our method defines the Pareto frontier in the low-compute regime, outperforming vid-TLDR on every benchmark at roughly a $\frac{1}{4}$ of its FLOPs, and matches or exceeds STTS and LookWhen at equal cost. RLT extends the frontier to higher accuracy, but costs more. Generally, we approach dense accuracy at a fraction of its cost.

this model is not adapted to sparse computation and therefore may select more tokens. STTS (Wang et al., 2022) trains a selector end-to-end using a differentiable selection relaxation, but does not scale to fine-grained token selection and instead operates on coarser regions.

Selection-Extraction. LookWhen (Salamatian et al., 2026) compares in feature space and uses uniqueness as a heuristic for diverse selection. It ranks uniqueness, and distills these scores into a shallow selector while simultaneously training a sparse extractor by supervising representation with a dense teacher. Diversity, however, is not necessarily aligned with what a task needs. Rare tokens may be distinctive but irrelevant, while common ones may provide useful context. The heuristic also relies on positionally grounded representations that are not generally available in video models, limiting generalization across models and requiring careful pretraining of an image model for video.

For images, LookThere (Rammohan et al., 2026) shows that heuristic selection can fail to generalize across models and tasks, and that learning selection-extraction by RL can do better. Videos exhibit more redundancy, offering more opportunity for efficiency, but scaling RL to video tokens is not entirely straightforward due to challenges with sample efficiency and credit assignment.

Token Reduction. Other work reduces tokens by merging or pruning, but does so gradually. STA (Ding et al., 2023) aggregates patch similarity across frames, reweighting these scores by activations to prune redundant tokens. Merging methods combine similar tokens using soft bipartite matching. ToMe (Bolya et al., 2022) uses similarity between token keys, while vid-TLDR (Choi et al., 2024) defines its score using attention sharpness to enable even faster reduction that outperforms ToMe.

Despite its effectiveness, token reduction is fundamentally reactive, and pays more to select. Tokens are first processed, most for several layers, and only then identified as redundant. Selection avoids this extra computation by lightweight selection of tokens before deep processing. This distinction is also important for hardware efficiency. Merging relies on clustering operations that may reduce FLOPs but remain slow on hardware (Fuller et al., 2025). Our selection relies instead on hardware-accelerated transformer layers to identify relevant tokens before expensive computation.

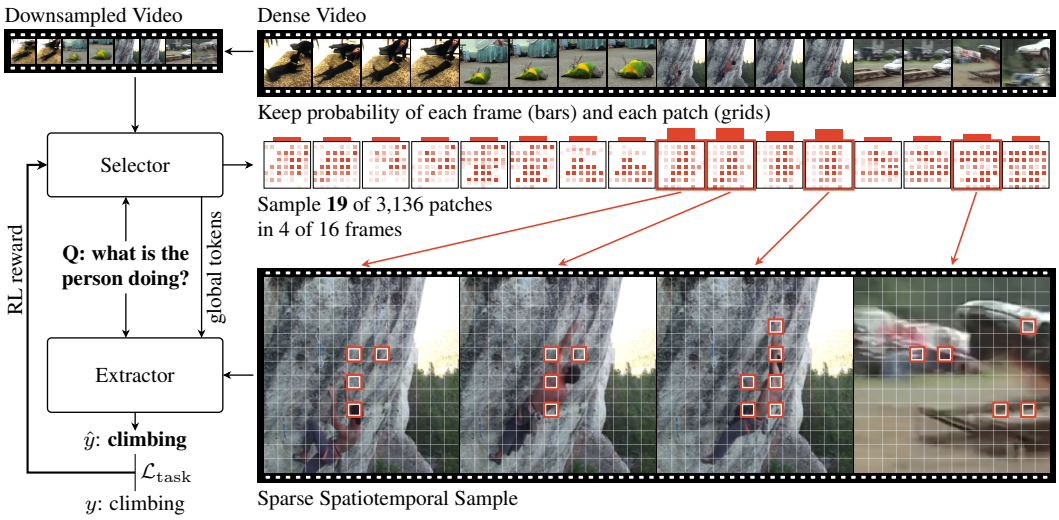


Figure 3: **Our method learns task-conditioned selection for video end-to-end.** The selector decides where and when to look from a question and low-resolution copy of the video. The extractor predicts the answer from the sampled high-resolution patches and the selector's global tokens. Here, the pair recognizes the person *climbing* from only 4 of 16 frames and 0.6% of the patches. The two are trained jointly by task supervision $\mathcal{L}_{\text{task}}$ and selection feedback $\mathcal{L}_{\text{selection}}$.

3 Method

We build on LookThere (Rammohan et al., 2026), which demonstrates that learned, task-specific selection outperforms heuristics for efficient processing of images, and extend the approach to video.

3.1 Selection-Extraction Factorization

Given a dataset $\mathcal{D}$ of videos x and labels y, we seek to efficiently predict y with minimal tokens. To this end, we factorize the prediction as $\hat{y} = f_\theta(\pi_\phi(x))$, where π_ϕ is a lightweight selector with parameters ϕ and f_θ is a higher-capacity feature extractor with parameters θ. Efficiency is gained by sparse selection, with selection's overhead being negligible relative to the cost of the tokens it removes. We train both components jointly by minimizing a task loss subject to a budget K

$$\min_{\theta,\phi} \mathcal{L}^{\theta}_{\text{task}}(y, f_\theta(\pi_\phi(x))) \qquad \text{s.t.} \qquad |\mathcal{P}| \le K \tag{1}$$

where $\mathcal{P}$ is the set of selected patches, and $\mathcal{L}_{\text{task}}$ is cross-entropy for classification (Sec. 4) or mean-squared error (MSE) for distillation pretraining by feature reconstruction (Sec. 3.3). Optimizing (1) is challenging because hard selection is non-differentiable in ϕ. Prior work on video sidesteps this challenge by learning from heuristics, whereas we learn ϕ directly by reinforcement (Sec. 3.5).

3.2 Inference with a Trained Selector and Extractor

Given a video x with T_E frames at resolution $R_E \times R_E$, the selector operates only on a downsampled copy of size $T_S \times R_S \times R_S$ using a reduced number L_S of layers, both making it cheaper than the extractor. It patchifies this copy into a token sequence, prepends G register tokens and T_S frame tokens, and encodes the result into latents that parameterize a selection policy (Fig. 3).

We model the policy as a product of independent Bernoulli distributions following Rammohan et al. (2026), one for each of the $T_E N_E^2$ extractor patches, with $N_E = R_E/P$ for patch size P. The selector projects its latents to logits ℓ_i for each extractor patch $x_E^{(i)}$, which is kept with probability $\sigma(\ell_i)$ where σ is the sigmoid function. Sampling $a_i \in \{0, 1\}$ for every patch yields the selected set of patches, $\mathcal{P}(a) := \{x_E^{(i)} \mid a_i = 1\}$, processed by the extractor, reducing token count for efficiency.

Having seen the whole video, the selector passes its global context forward by prepending its register and frame tokens to the selected patch sequence, linearly interpolating from T_S to T_E (Fuller et al.,

2025; Salamatian et al., 2026). This costs nothing extra for a small gain in performance (App. A.2.1). Standard transformer layers then map this sparse input to features $\hat{z}_E \in \mathbb{R}^{(G+T_E+|\mathcal{P}(a)|)\times D}$, and a task-specific head reads the first register token, serving as the class token, to predict $\hat{y}$.

3.3 Policy Initialization by Distillation Pretraining

Learning to sparsify without a prior is difficult. From random masks, the selector may struggle to assign credit to the patches that matter for recognition, and training the extractor with poor selection may harm its performance. Hence, we prepare both components for downstream fine-tuning by distilling. The extractor learns to reconstruct dense representation from sparse input, while the selector learns by RL to select the patches that minimize the resulting loss at higher sparsity. Though not necessary, pretraining improves both downstream performance and sample efficiency (Sec. 4.3).

Teachers. We produce the targets for pretraining by processing the full input with a frozen teacher. Many backbones are suitable as these targets do not directly supervise selection. We use DINOv3 (Siméoni et al., 2026), for comparison with LookWhen (Salamatian et al., 2026), and the video models InternVideo2 (Wang et al., 2024) and VideoMAEv2 (Wang et al., 2023). The extractor is initialized from the teacher, and the selector from the teacher's first L_S layers.

Extractor Pretraining. We train the extractor by reconstructing two summaries provided by the teacher: a video token $z^{\text{teacher}}_{\text{video}} \in \mathbb{R}^D$ and per-frame tokens $z^{\text{teacher}}_{\text{frames}} \in \mathbb{R}^{T_E\times D}$, pooled from the teacher's dense features by frame (details in App. A.3.1). Unlike LookWhen (Salamatian et al., 2026), we also align the selector's frame tokens $\hat{z}^{\text{sel}}_{\text{frames}} \in \mathbb{R}^{T_S\times D}$ with the frame targets to help the selector reason in time. The distillation loss then sums the three resulting reconstruction terms

$$\mathcal{L}^{\theta}_{\text{distill}} = \text{MSE}(\hat{z}_{\text{video}}, z^{\text{teacher}}_{\text{video}}) + \text{MSE}(\hat{z}_{\text{frames}}, z^{\text{teacher}}_{\text{frames}}) + \text{MSE}(\hat{z}^{\text{sel}}_{\text{frames}}, z^{\text{teacher}}_{\text{frames}}). \tag{2}$$

Selector Pretraining. The selector trains by reinforcement learning, rewarding selections that distill well from fewer patches. To prevent indiscriminate dropping for a sparsity bonus, we adapt Look-There's (Rammohan et al., 2026) gated reward and grant the bonus only when the loss is lower than its running median. Hence, the median falls and the bar for sparsity rises as the extractor improves.

3.4 Hierarchical Spatio-temporal Selection

Discriminative information in video tends to concentrate in specific regions and moments, but independent patch sampling ignores these correlations. Rewarding large sets of selected patches also increases the difficulty of credit assignment. We respond by factorizing the policy hierarchically, refining selection from coarse to fine with the selector first deciding *when*, then *where*, to process.

Temporal Selection. The selector first decides *when* to look, by mapping its frame tokens to logits ℓ^{frame}_j over the T_E extractor frames. As video is highly redundant, dropping a single frame barely affects the extractor's prediction, which obscures the frame's relevance. We therefore average-pool frame logits over contiguous groups that are kept or dropped jointly, shrinking the group size during training (e.g., $8 \to 4 \to 2 \to 1$ for 16 frames). Dropping a large group removes information unavailable elsewhere, and the coarser action spaces are smaller, both aspects helping improve credit assignment. The selector then learns which segments matter before refining to individual frames.

Spatial Cascade. Within each frame, the selector chooses patches via nested hierarchy over increasing grid resolutions $g_1 < \cdots < g_L = N_E$ following Rammohan et al. (2026). Patch logits ℓ_i for frame i are average-pooled to each resolution and shifted down wherever the parent was dropped, encouraging large redundant areas to be dropped by consensus, rather than one patch at a time

$$\ell^{(l)}_i = \text{AvgPool}\left(\ell_i, g_l\right) - \beta_l \cdot \text{UpSample}\left(1 - a^{(l-1)}_i, g_l\right), \tag{3}$$

where $a^{(l)}_i$ is frame i's mask at level l, upsampling is nearest-neighbor, and each $\beta_l \in \mathbb{R}$ is learned. At the coarsest level, the parent is the frame decision itself $a^{(0)}_i := a^{\text{frame}}_i$, which couples *where* with *when*. With $a^{\text{frame}} \in \{0,1\}^{T_E}$ as the set of selected frames, the full policy then factors as

$$\pi_\phi(a \mid x) = \pi^{\text{frame}}_\phi(a^{\text{frame}} \mid x) \prod_{i=1}^{T_E} \prod_{l=1}^{L} \pi_\phi\left(a^{(l)}_i \mid a^{(l-1)}_i, x\right). \tag{4}$$

3.5 Learning the Selector and Extractor

Selection is discrete and thus non-differentiable, complicating optimization (Cordonnier et al., 2021; Wang et al., 2022). Prior work sidesteps this difficulty with heuristics based on diversity (Hwang et al., 2024; Choi et al., 2024; Choudhury et al., 2024; Salamatian et al., 2026) or reconstruction (Park et al., 2022; Shi et al., 2026), but may miss task-relevant tokens and fail to generalize across models. Following Rammohan et al. (2026), we instead optimize selection *directly* for the downstream objective in (1) with reinforcement learning, targeting high performance at high sparsity.

Formulation. We cast selection as a one-step Markov decision process with videos $x \in \mathcal{X}$ as states and whose actions are selection masks $a \in \mathcal{A}$. The selector is a policy $\pi_\phi : \mathcal{X} \to \Delta_{\mathcal{A}}$ over masks.

Rewards. We encourage the selector to balance task performance against sparsity with a reward $r(x, a) = -\mathcal{L}^\theta_{\text{task}} + \lambda s(a)$ where $s(a) = 1 - |\mathcal{P}(a)|/(T_E N_E^2)$ is the fraction of patches dropped and λ balances the trade-off. To prevent the selector from discarding patches faster than the extractor can adapt to the resulting sparsity, we gate the sparsity term on task success, which yields a self-paced curriculum. Specifically, classification requires $\hat{y} = y$, while distillation and regression require the task loss to fall below a running threshold, which we set to the median loss. We then target a desired sparsity ρ by rewarding selection masks according to their proximity to the target

$$r(x, a) = -\mathcal{L}^\theta_{\text{task}} + \lambda \cdot \mathbf{1}[\text{success}] s_\rho(a), \qquad s_\rho(a) = 1 - \frac{|s(a) - \rho|}{\max(\rho, 1 - \rho)}. \tag{5}$$

Optimization. Policy learning amounts to maximizing the expected reward by the policy gradient

$$\phi' \leftarrow \phi + \alpha \nabla_\phi \mathbb{E}_{\pi_\phi}[r(x, a)], \quad \text{with} \quad \nabla_\phi \mathbb{E}_{\pi_\phi}[r(x, a)] = \mathbb{E}_{\pi_\phi}[r(x, a) \nabla_\phi \log \pi_\phi(a \mid x)]. \tag{6}$$

We estimate this gradient with Group Relative Policy Optimization (GRPO), contrasting M masks sampled for the same video. Given $a_{(m)} \sim \pi_\phi(\cdot \mid x)$, $m \in [M]$, the group baseline $\overline{r}$ is the mean reward, and normalizing by the group standard deviation $\hat{\sigma}_r$ gives the selection loss

$$\mathcal{L}^\phi_{\text{selection}} = -\frac{1}{M} \sum_{m=1}^{M} \log \pi_\phi(a_{(m)} \mid x) \, \text{clip}\left(\frac{r_{(m)} - \overline{r}}{\hat{\sigma}_r}, -\alpha, \alpha \right), \tag{7}$$

where the group-relative advantage provides a stronger relative signal for credit assignment by contrasting multiple masks for each video, and the α-clipping stabilizes optimization.

Conditioned selection. The same video may require different glimpses depending on what we seek to recognize. Because our selection is learned from reward rather than fixed by heuristics, it adapts naturally to such context by augmenting the state with a context c, so the reward becomes

$$r((x, c), a) = -\mathcal{L}^\theta_{\text{task}}(c) + \lambda \cdot \mathbf{1}[\text{success}] s_\rho(a). \tag{8}$$

Implementation Details. We first warm up the extractor on the full input so it provides useful feedback for selection (Rammohan et al., 2026), then fine-tune the selector and extractor jointly with RL, improving efficiency while maintaining performance. The selector uses $L_S = 3$ layers for general selection and $L_S = 6$ for text-conditioned selection, where the added capacity yields more accurate frame tokens. Following LookWhen (Salamatian et al., 2026), we use ViT-B models: the extractor receives $T_E = 16$ frames at $R_E = 224$px and the selector half of each ($T_S = 8$, $R_S = 112$px), yielding a $50\times$ FLOP reduction. We fix $\lambda = 0.5$.

4 Experiments

We evaluate across diverse tasks and pretrained backbones, measuring efficiency, accuracy, and robustness to initialization. For action recognition (Sec. 4.1), we pretrain by distillation on K400 (Kay et al., 2017) and SSv2 (Goyal et al., 2017) (Sec. 3.3) and fine-tune on these and other standard benchmarks. For task-conditioned selection (Sec. 4.2), we prompt with text for class-specific retrieval on A2D (Xu et al., 2015) and question answering on CLEVRER (Yi et al., 2019). Finally, we ablate the selector, the extractor, and key choices in the reward, policy, and sparsification (Sec. 4.3).

Baselines. We compare against SoTA token selection (STTS (Wang et al., 2022), RLT (Choudhury et al., 2024), LookWhen (Salamatian et al., 2026)) and token reduction (vid-TLDR (Choi et al., 2024)) across several backbones to assess robustness to initialization. LookWhen is our primary baseline, holding the prior SoTA in the efficiency-accuracy trade-off. We share its architecture, so the comparison isolates the effect of optimizing selection for the task rather than for diversity.

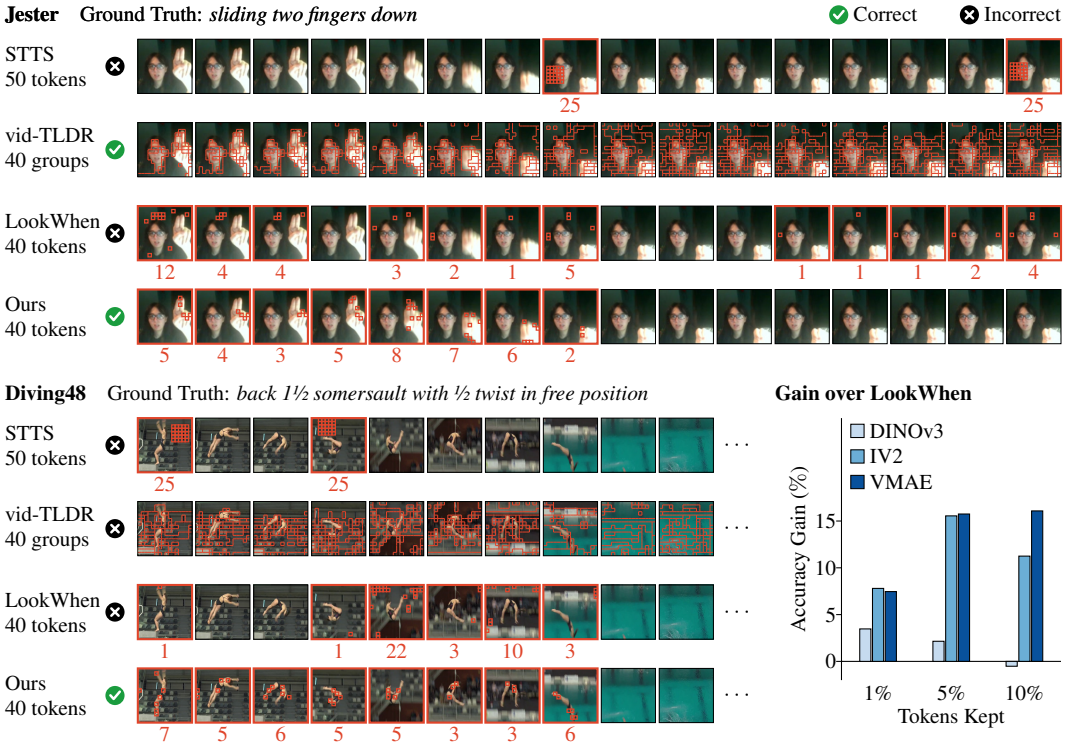


Figure 4: **Learned selection keeps the few tokens that distinguish the action.** We visualize selection on Jester and Diving48 with InternVideo2 at $\sim$1% of tokens (numbers give tokens kept per frame). Our selector follows the moving hand and diver, predicting both clips correctly. STTS keeps coarse regions in only two frames, vid-TLDR merges the clip into 40 tokens, and LookWhen-IV2 spends its budget away from the motion. (right) Average accuracy gain of our method over LookWhen at 1%/5%/10% of tokens. Gains are lower for DINOv3, whose patch features suit LookWhen's uniqueness heuristic, but reach 7-16% on InternVideo2 and VideoMAEv2. See App. A.4.

4.1 Action Recognition with Diverse Pretrained Backbones

We post-train selection-extraction by RL (Sec. 3.5) using DINOv3 (Siméoni et al., 2026), InternVideo2 (Wang et al., 2024), and VideoMAEv2 (Wang et al., 2023) as backbones to assess robustness to the choice of pretraining. These backbones represent SoTA in image and video recognition, and their general representations from large-scale self-supervised training make them strong teachers.

Setup. We pretrain all models for 30 epochs on a dataset combining Kinetics-400 (Kay et al., 2017) and SSv2 (Goyal et al., 2017) following Sec. 3.3, initializing the selector and extractor from each backbone. We then fine-tune for 40 epochs per downstream dataset with batch size 96; the first 10 epochs only train the extractor, serving as warmup, and use all patches (App. A.3).

We tune our hyperparameters once (Sec. 4.3) and hold them fixed for practicality, with selector and extractor learning rates of 2e$-$5 and 1e$-$4. Each method is fine-tuned separately at every target sparsity, and the baselines are tuned equally. We sweep LookWhen's learning rate over $\{3e{-}5, 1e{-}4, 3e{-}4, 5e{-}4\}$, RLT's threshold over $\tau \in \{0.05, 0.1, 0.2\}$, and use the four documented sparsity schedules for vid-TLDR (Choi et al., 2024). Remaining hyperparameters follow each baseline's tuned configuration, and we report the best-performing setting.

Datasets. To evaluate both motion and appearance understanding, we use six common video recognition benchmarks (Wang et al., 2024; Assran et al., 2025; Salamatian et al., 2026) with distinct sources of discriminative information. Charades, Diving48, and Jester emphasize human actions, movements, and gestures, while SSv2, K400, and EK100 also depend on objects and scene context.

Downstream Performance. Averaged across backbones, our approach establishes a new accuracy-efficiency Pareto frontier on all six tasks at high sparsity (90-99%; Fig. 2). Learned task-specific selection achieves the highest accuracy among adaptive methods with substantially less computation.

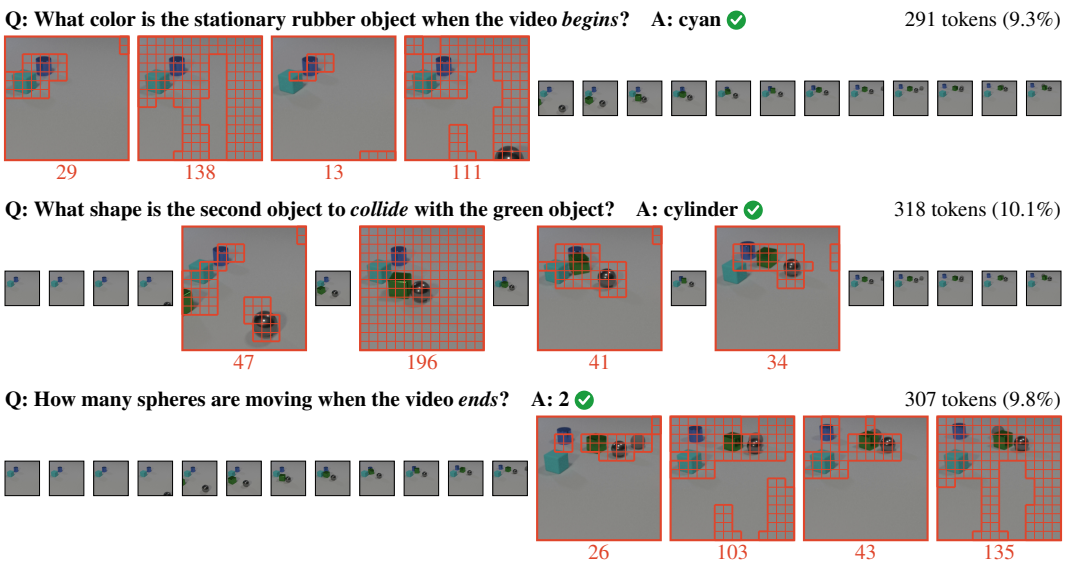


Figure 5: **Different questions need adaptive selections.** We visualize the selected frames/tokens on CLEVRER for three questions. Learned selection follows the question's timing, keeping early frames for *begins*, frames around the collisions for *collide*, and late frames for *ends*. Within those frames, it spends its budget where the answer lies, from a few tokens on the relevant objects to the full frame at the moment of collision.

STTS, which also learns selection but at coarser scales, achieves higher throughput at the highest sparsity but up to 10% lower accuracy. Sharing LookWhen's architecture lets us inherit its efficiency while improving accuracy by up to 10%. Against vid-TLDR, we achieve higher accuracy at a quarter of the FLOPs and $4\times$ less memory. We rival RLT in accuracy while increasing throughput by $5\times$ and reducing FLOPs by $10\times$, though RLT reaches higher accuracy with more computation.

Generalization to Initialization. Across DINOv3, VideoMAEv2, and InternVideo2, our learned selection consistently delivers the best accuracy-efficiency trade-off, with performance tracking each backbone's dense accuracy rather than its fit to a fixed heuristic (Figs. 7 to 9). Our approach generally retains over 95% of dense performance with $\frac{1}{10}$ of the tokens and $9\times$ fewer FLOPs.

On DINOv3, used by LookWhen for its strong patch representations, we closely match or exceed it on every benchmark (Fig. 10). Since there are no released implementations for other backbones, we reproduce LookWhen on VideoMAEv2 and InternVideo2. Our DINOv3 reproduction exceeds their reported results, yet falls behind using the video backbones, as LookWhen's uniqueness heuristic requires salient, positionally grounded representations that these video models lack (Fig. 4).

Our joint optimization instead learns selection end-to-end, relying less on pretrained features; it outperforms heuristics by 10-20$\%$ on video backbones, suggesting applicability to future models.

4.2 Conditioning Selection by Textual Instruction

Different tasks may require different information from the same video. Existing methods select the visually interesting, but interest is not relevance, so they must select broadly to avoid missing what matters. We instead optimize selection for the objective itself, and demonstrate by conditioning selection on text to enable task-relevant selection from the same video depending on a query.

CLEVRER (Yi et al., 2019) is a standard question-answering testbed on interactions among colored shapes. We use its descriptive subset of over 100K prompts and 13 classes ($\approx$ 100K/50K train/val), and fine-tune for 50 epochs (25 warmup) at 224^2px. Selection adapts frames and tokens to each question (Fig. 5). At 95% sparsity, it reaches 78.1% accuracy, nearly matching the dense backbone's 81.7% and outperforming random and heuristic selection by 8% and 20% (Fig. 2).

A2D Retrieval. We test text-conditioned selection further with a retrieval task from the Actor-Action Dataset, which contains 3782 clips of 7 actors performing 7 actions (Xu et al., 2015). Ex-

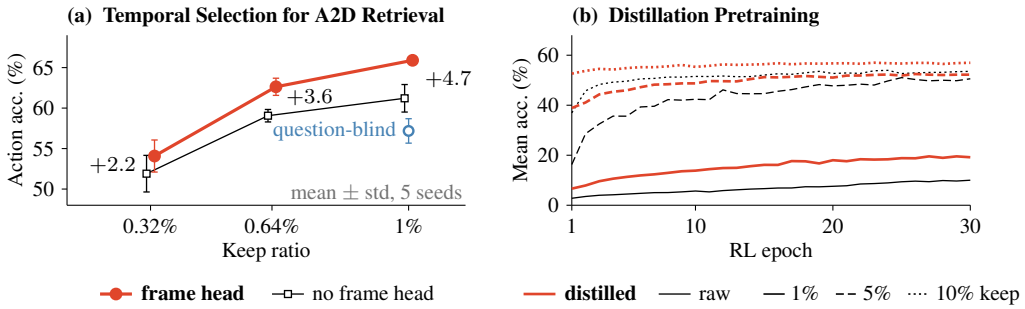


Figure 6: **Ablating temporal selection and distillation.** (a) Temporal selection improves at every budget on A2D retrieval, with gains growing as budget increases. A question-blind selector falls below the model without a frame head, showing the benefits of conditioned selection. (b) Distillation gives RL a stronger starting point, leading to improved final accuracy, with the largest gains at the sparsest budgets; averaged over Diving48, EK100, and Charades using InternVideo2 as a backbone.

amples combine 4 clips of different actors and ask about one. Learned selection again achieves the best trade-off, ahead of LookWhen and random, which in turn beat zero-shot InternVideo2-Chat-8B. At 95% sparsity, we reach 74.9% accuracy, close to the dense model's 79.2% (Fig. 2). Hiding the question from the selector costs 8.7% at 99% sparsity, highlighting the benefit of conditioning.

4.3 Ablations

Our approach is robust across a range of its hyperparameters. As the selector drives both efficiency and performance, we ablate the design choices targeting the challenges of learning by reinforcement.

Temporal Selection. We revisit temporal selection on A2D retrieval to measure the benefit of selecting frames. Frame selection raises the precision of selected patches that land in the relevant clip across sparsity levels, from 71/68/53% to 82/83/84% when keeping 0.32/0.64/1% of tokens. The increased precision leads to improved accuracy, improving performance by up to 4.7% (Fig. 6a).

Distillation Pretraining. We ablate distillation pretraining of the extractor and selector, using distillation loss as reward for selection learning. Pretrained models start with and maintain higher accuracy throughout RL finetuning on Diving48, EK100, and Charades (Fig. 6b). Without pretraining, models learn selection and sparse input representations from scratch downstream, limiting the sparsity that can be reached before degradation. Pretraining shifts this learning upstream to a stage that transfers downstream, improving sample-efficiency and amortizing the cost of later finetuning.

Learning Rates, Sparsity, and Entropy. We sweep the Cartesian product of selector LRs 2e−5/5e−5/1e−4, extractor LRs 5e−5/1e−4/5e−4, entropy coefficients 2/5/10, and sparsity weights λ of 0.5/1/1.5, for 81 runs on Diving48. Mean accuracy varies by at most 2 points across each hyperparameter. An extractor LR of 1e−4 is best in 26 of 27 settings, smaller selector LRs and entropy coefficients are preferred, and λ has little effect. We select 2e−5/1e−4/2/0.5 (App. A.2.2).

5 Conclusion

We introduce a reinforcement learning framework for sparse video understanding that replaces heuristics and optimizes token selection directly for accuracy and sparsity. A selector learns when and where to look, an extractor learns what to see, and together they converge on sparse glimpses that suffice for the task. Across diverse datasets and backbones, our approach delivers efficient recognition that remains accurate at high sparsity. Conditioning selection on text extends these gains to question answering and retrieval, where the question defines both the task and the tokens that matter.

Limitations and Future Work. Compute limits our evaluation to ViT-B models, and rewarding whole sets of patches makes credit assignment coarse; hierarchical selection and GRPO provide attribution, but denser rewards may do better. Beyond recognition and prediction, selection may benefit other tasks and even be learned by RL *for* RL, via sparse world models for efficient planning. Selection could also be recursive, choosing future tokens based on what has already been seen.

AI USE STATEMENT

In this work, we used generative AI tools for polishing our writing and creating figures, retrieval and discovery of related work, and research execution. We have not used generative AI tools for drafting written sections of the paper, generating synthetic datasets, or proving mathematical claims. We have reviewed all AI-assisted work. For example, reading all surfaced related works, testing research code thoroughly, and checking to ensure generated figures remain faithful to the underlying data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the necessary resources to ensure reproducibility of our work. All hyperparameters and training recipes are documented in the Appendix, and we will release our code and models upon acceptance.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Benjamin Bergner, Christoph Lippert, and Aravindh Mahendran. Token Cropr: Faster ViTs for quite a few tasks. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9740–9750. IEEE, 2025.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Joonmyung Choi, Sanghyeok Lee, Jaewon Chu, Minhyuk Choi, and Hyunwoo J Kim. vid-TLDR: Training free token merging for light-weight video transformer. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18771–18781. IEEE, 2024.

Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M Kitani, and László A Jeni. Don't look twice: Faster video transformers with run-length tokenization. *Advances in Neural Information Processing Systems*, 37:28127–28149, 2024.

Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. Differentiable patch selection for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2351–2360, 2021.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The EPIC-KITCHENS dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141, 2020.

Shuangrui Ding, Peisen Zhao, Xiaopeng Zhang, Rui Qian, Hongkai Xiong, and Qi Tian. Prune spatio-temporal tokens by semantic-aware temporal accumulation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16899–16910. IEEE, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3298–3308. IEEE, 2021.

Anthony Fuller, Yousef Yassin, Junfeng Wen, Tarek Ibrahim, Daniel Kyrollos, James Green, and Evan Shelhamer. LookWhere? Efficient visual recognition by learning where to look and what to see from self-supervision. *Advances in Neural Information Processing Systems*, 38:87001–87039, 2025.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "Something Something" video database for learning and evaluating visual common sense. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 5843–5851. IEEE, 2017.

Xinyue Hao, Gen Li, Shreyank N Gowda, Robert B Fisher, Jonathan Huang, Anurag Arnab, and Laura Sevilla-Lara. Principles of visual tokens for efficient video understanding. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21254–21264. IEEE, 2025.

Sunil Hwang, Jaehong Yoon, Youngwan Lee, and Sung Ju Hwang. EVEREST: Efficient masked video autoencoder by removing redundant spatiotemporal tokens. In *Forty-first International Conference on Machine Learning*, 2024.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. VidToMe: Video token merging for zero-shot video editing. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7486–7495. IEEE, 2024.

Yingwei Li, Yi Li, and Nuno Vasconcelos. RESOUND: Towards action recognition without representation bias. In *European conference on computer vision*, pp. 520–535. Springer, 2018.

Joanna Materzynska, Guillaume Berger, Ingo Bax, and Roland Memisevic. The Jester dataset: A large-scale video dataset of human gestures. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2874–2882. IEEE, 2019.

Seong Hyeon Park, Jihoon Tack, Byeongho Heo, Jung-Woo Ha, and Jinwoo Shin. K-centered patch sampling for efficient video recognition. In *European Conference on Computer Vision*, pp. 160–176. Springer, 2022.

Sreehari Rammohan, Yousef Yassin, Anthony Fuller, Junfeng Wen, Carl Vondrick, and Evan Shelhamer. LookThere! Sparse vision by reinforced selection. *arXiv*, 2026.

Jona Ruthardt, Manu Gaur, Deva Ramanan, Makarand Tapaswi, and Yuki M. Asano. Steerable visual representations. In *European Conference on Computer Vision (ECCV)*, 2026.

Ali Salamatian, Anthony Fuller, Pritam Sarkar, James R Green, Leonid Sigal, and Evan Shelhamer. LookWhen? Fast video recognition by learning when, where, and what to compute. *arXiv preprint arXiv:2605.06809*, 2026.

Baifeng Shi, Stephanie Fu, Long Lian, Hanrong Ye, David Eigen, Aaron Reite, Jan Kautz, Boyi Li, David M Chan, Trevor Darrell, et al. Attend before attention: Efficient and scalable video understanding via autoregressive gazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17022–17034, 2026.

Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European conference on computer vision*, pp. 510–526. Springer, 2016.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. Featured Certification.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Junke Wang, Xitong Yang, Hengduo Li, Li Liu, Zuxuan Wu, and Yu-Gang Jiang. Efficient video transformers with spatial-temporal token selection. In *European Conference on Computer Vision*, pp. 69–86. Springer, 2022.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018a.

Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14549–14560. IEEE, 2023.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 7794–7803. IEEE, 2018b.

Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, et al. InternVideo2: Scaling foundation models for multimodal video understanding. In *European conference on computer vision*, pp. 396–416. Springer, 2024.

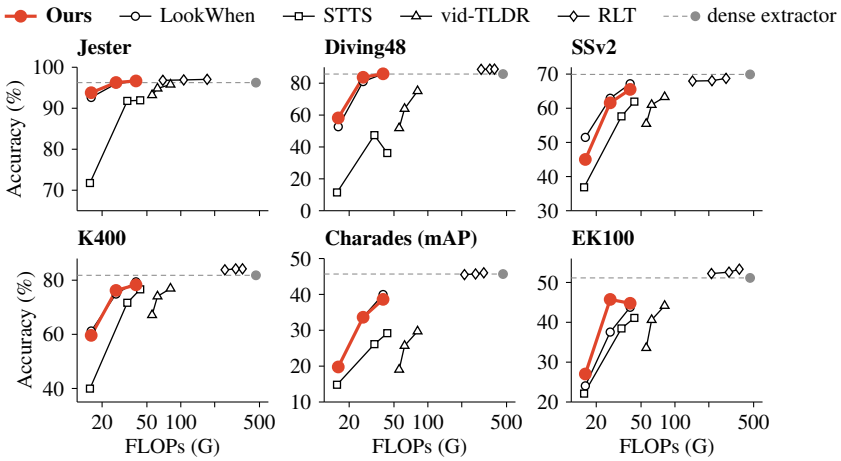
Chenliang Xu, Shao-Hang Hsieh, Caiming Xiong, and Jason J Corso. Can humans fly? Action understanding with multiple classes of actors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2264–2273, 2015.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. CLEVRER: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A Appendix

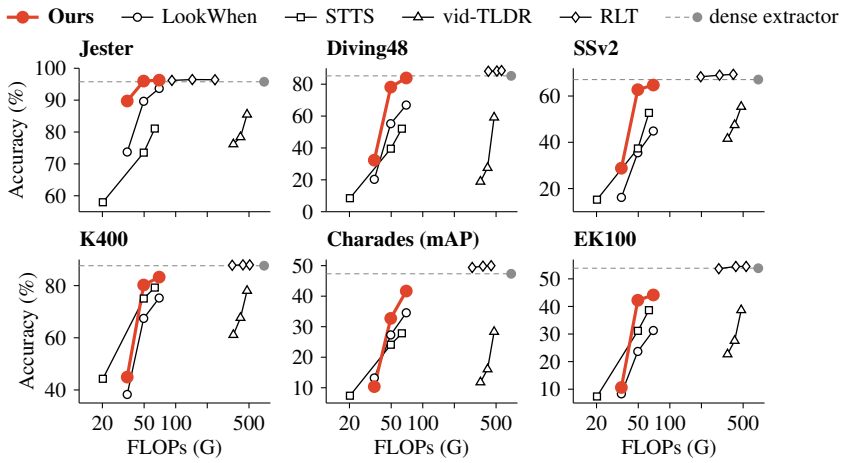
A.1 Accuracy-Efficiency by Backbone



		Dense*	STTS	RLT†	vid-TLDR	LookWhen	Ours
FLOPs (G)	90% sparsity	465.0	43.6	314.5	81.0	40.0	**40.0**
	95% sparsity	465.0	33.5	258.2	62.1	26.6	**26.6**
	99% sparsity	465.0	15.6	196.4	55.6	16.0	**16.0**
Speed (clip/s)	90% sparsity	58.5	352.9	84.0	62.9	406.8	**406.8**
	95% sparsity	58.5	431.4	100.8	74.9	554.0	**554.0**
	99% sparsity	58.5	766.9	125.5	77.0	775.8	**775.8**
Memory (GB)	all levels	0.50	0.40	0.40	1.48	0.46	**0.46**

*Our dense extractor on DINOv3. †RLT at τ=0.05/0.1/0.2 for 90/95/99%; FLOPs are the mean over the six datasets.

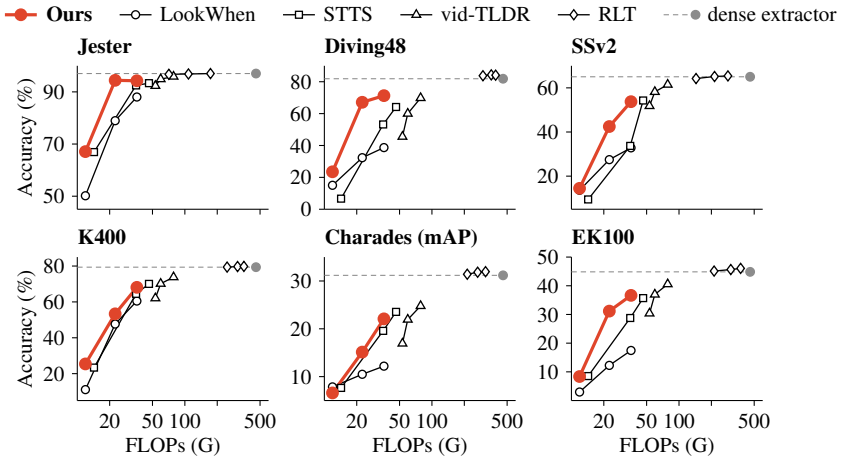
Figure 7: **DINOv3** Accuracy vs. inference FLOPs at 90/95/99 sparsity for action recognition.



		Dense*	STTS	RLT†	vid-TLDR	LookWhen	Ours
FLOPs (G)	90% sparsity	696.9	63.2	449.2	479.1	69.6	**69.6**
	95% sparsity	696.9	49.6	365.0	414.7	49.5	**49.5**
	99% sparsity	696.9	20.2	274.3	354.7	34.5	**34.5**
Speed (clip/s)	90% sparsity	36.8	253.2	55.9	44.3	237.1	**237.1**
	95% sparsity	36.8	308.9	66.5	51.5	312.0	**312.0**
	99% sparsity	36.8	583.1	85.0	52.6	412.9	**412.9**
Memory (GB)	all levels	0.70	0.51	0.48	2.35	0.63	**0.63**

*Our dense extractor on InternVideo2. †RLT at τ=0.05/0.1/0.2 for 90/95/99%; FLOPs are the mean over the six datasets.

Figure 8: **InternVideo2** Accuracy vs. inference FLOPs at 90/95/99% sparsity for action recognition.



		Dense*	STTS	RLT†	vid-TLDR	LookWhen	Ours
FLOPs (G)	90% sparsity	458.3	46.4	315.3	78.5	35.8	**35.8**
	95% sparsity	458.3	35.2	259.3	59.6	22.5	**22.5**
	99% sparsity	458.3	14.3	197.5	53.2	11.9	**11.9**
Speed (clip/s)	90% sparsity	80.7	425.9	117.0	80.3	474.9	**474.9**
	95% sparsity	80.7	500.1	140.3	100.0	590.0	**590.0**
	99% sparsity	80.7	736.5	175.6	102.8	731.8	**731.8**
Memory (GB)	all levels	0.50	0.41	0.41	1.48	0.49	**0.49**

*Our dense extractor on VideoMAEv2. †RLT at τ=0.05/0.1/0.2 for 90/95/99%; FLOPs are the mean over the six datasets.

Figure 9: **VideoMAEv2** Accuracy vs. infer. FLOPs at 90/95/99% sparsity for action recognition.

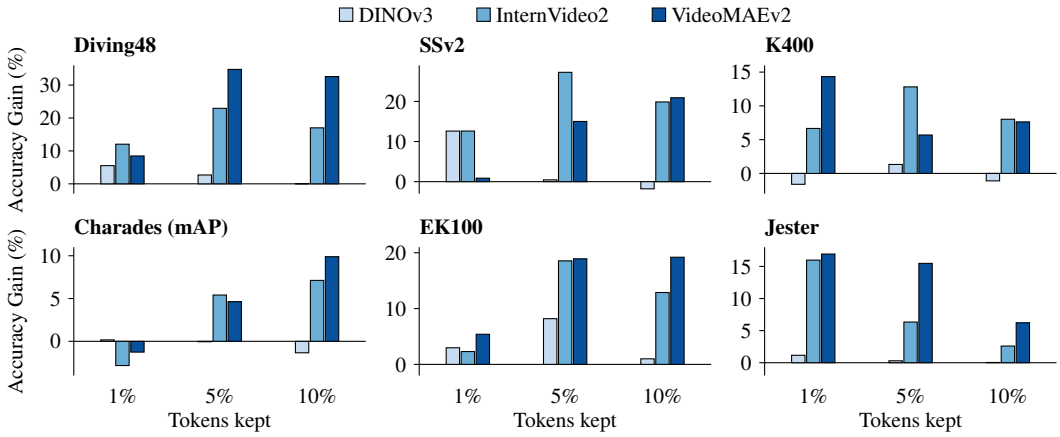


Figure 10: **Gain over LookWhen across backbones.** Accuracy of our method minus LookWhen's (mAP for Charades) at 1%/5%/10% of tokens.

A.2 Additional Ablations

A.2.1 Ablating Selector Tokens Fed Into Extractor

To ensure that it is the patch tokens contributing to performance gain and not the selector state (registers and global CLS representations) which is passed to the extractor, we run an ablation on Diving48 which fixes the selector CLS and 4 selector registers to learned constants in the extractor. At the top-k 1% token budget, accuracy decreases 2% from 65.6% to 63.6% when removing the selector global state from the extractor. Selector global state leads to small improvements in performance, but the patch tokens chosen are carrying most of the performance gain.

A.2.2 Selector LR, Extractor LR, Entropy, and λ sparsity

For RL post-training, we perform an 81-run sweep encompassing the cross product of the following: selector LRs {2e-5, 5e-5, 1e-4}, extractor LR {5e-5, 1e-4, 5e-4}, entropy {2, 5, 10}, and λ sparsity {0.5, 1, 1.5} on Diving48 starting from our RL distillation (Section 3.3) pretrained DINOv3 backbone and training for 40 RL epochs (excluding an initial 10 warmup epochs). Our best run serves as our base configuration and achieves 86.1% on diving48: Selector LR 2e-5, Extractor LR 1e-4, Entropy coefficient 2, Sparsity λ 0.5.

Table 1: Sweep over selector/extractor LR, entropy, and λ during RL post-training

Hyperparameter	Value	Δ vs. recipe	Mean over grid	Wins (of 27)
Selector LR	**2e−5**	–	83.1	16
	5e−5	−0.5	82.6	11
	1e−4	−1.4	81.7	0
Extractor LR	5e−5	−2	81.7	0
	1e−4	–	83.7	26
	5e−4	−1.5	82.2	1
Entropy coefficient	**2**	–	83.4	21
	5	−0.7	82.7	6
	10	−1.9	81.5	0
Sparsity weight λ	**0.5**	–	82.3	11
	1	−0.8	81.5	10
	1.5	−1.1	81.2	6

A.3 Implementation Details

We follow the training recipe used by LookWhen (Salamatian et al., 2026) unless noted otherwise.

A.3.1 Pretraining Settings

We pretrain for 30 epochs on K400 and SSv2, using the fine-tuning input size and augmentations (Tab. 3). We keep the teacher frozen. The extractor is initialized with its weights, and the selector from its first $L_S = 3$ layers.

We distill video backbones from themselves, with the primary purpose of sparse approximation of dense computation. For InternVideo2 (Wang et al., 2024) and VideoMAEv2 (Wang et al., 2023), the video target is the model's own pooled token (attention pooling and mean pooling, respectively), and each frame target pools that frame's tokens. DINOv3 (Siméoni et al., 2026), an image model, needs more help to interpret time, so we distill with mixed teachers following Salamatian et al. (2026). The image model's per-frame classification tokens supply the frame targets. They are concatenated to make a video target, to which we also add InternVideo2's pooled token.

The teacher is the bottleneck, so we reuse each batch $8\times$. Flipping videos and targets together expands the batch $2\times$, and mixing up (Zhang et al., 2017) the pairs expands it another $4\times$. The dense teacher runs once for every 8 sparse passes of the selector and extractor, and each video is seen $30 \cdot 8 = 240$ times over training.

We use our finetuning learning rate and batch size (Tab. 3).

A.3.2 Finetuning Settings

We initialize the classification head from a linear probe, trained for 100 epochs with settings in Tab. 2. All experiments use the full training sets, and we evaluate on the respective validation sets. Following LookWhen (Salamatian et al., 2026), we sample K400 densely (Wang et al., 2018b; Feichtenhofer et al., 2021) and all other datasets TSN-style (Wang et al., 2018a).

We tune our method once and hold the best parameters fixed (Tab. 3). The baselines are tuned equally. We sweep LookWhen's learning rate over $\{3\text{e}{-5}, 1\text{e}{-4}, 3\text{e}{-4}, 5\text{e}{-4}\}$ and RLT's threshold over $\tau \in \{0.05, 0.1, 0.2\}$. For vid-TLDR, we follow its authors and tune the merging schedule. At each sparsity level, we run the four templates from their Table A and report the best.

Ablations run using their respective datasets' settings.

A.3.3 Textual Conditioning

We demonstrate examples with text-conditioned selection by adding the gated cross-attention of SteerViT (Ruthardt et al., 2026) to both the selector and extractor in Sec. 4.2. We use a DINOv3 backbone for these experiments, with $L_S = 6$ layers to learn strong selector frame token representations that improve contextual frame selection. We use the settings from SSv2 (Tab. 3) for experiments on CLEVRER and A2D retrieval.

Table 2: **Linear probing configuration.**

Config	Value
Optimizer	AdamW
Base learning rate	1e-4
Batch size	256
Learning rate schedule	Cosine annealing

Table 3: **Fine-tuning configuration.**

Dataset	K400	SSv2	Diving48	EK100	Jester	Charades
Optimizer	AdamW					
Weight decay	0.01					
Min. LR	5×10^{-6}					
Batch size	96					
LR schedule	Cosine annealing w/ linear warmup					
Selector LR	2×10^{-5}					
Extractor LR	1×10^{-4}					
Entropy coefficient	2					
Sparsity weight λ	0.5					
Training epochs	40					
Extractor warmup epochs	10 (all patches)					
Input resolution	224×224					
# input frames	16					
Drop path rate	0.1					
Random crop scale	$[0.5,\ 1.0]$					
Horizontal flip	✓	✗	✓	✗	✗	✗
Color jitter	$(p{=}0.8,\ b/c/s{=}0.4,\ h{=}0.1)$					
Gaussian noise	$(p{=}0.5,\ \sigma{=}0.1)$					
Batch renorm. aug.	$(p{=}0.5)$					
Mixup	$(\alpha{=}0.8)$					
CutMix	$(\alpha{=}1.0)$					
Label smoothing	0.1	0.1	0.1	0.1	0.1	0.0

A.4 Visualizing Learned Selection



Figure 11: **DINOv3 on Jester at a 1% target**



Figure 12: **DINOv3 on Jester at a 5% target**

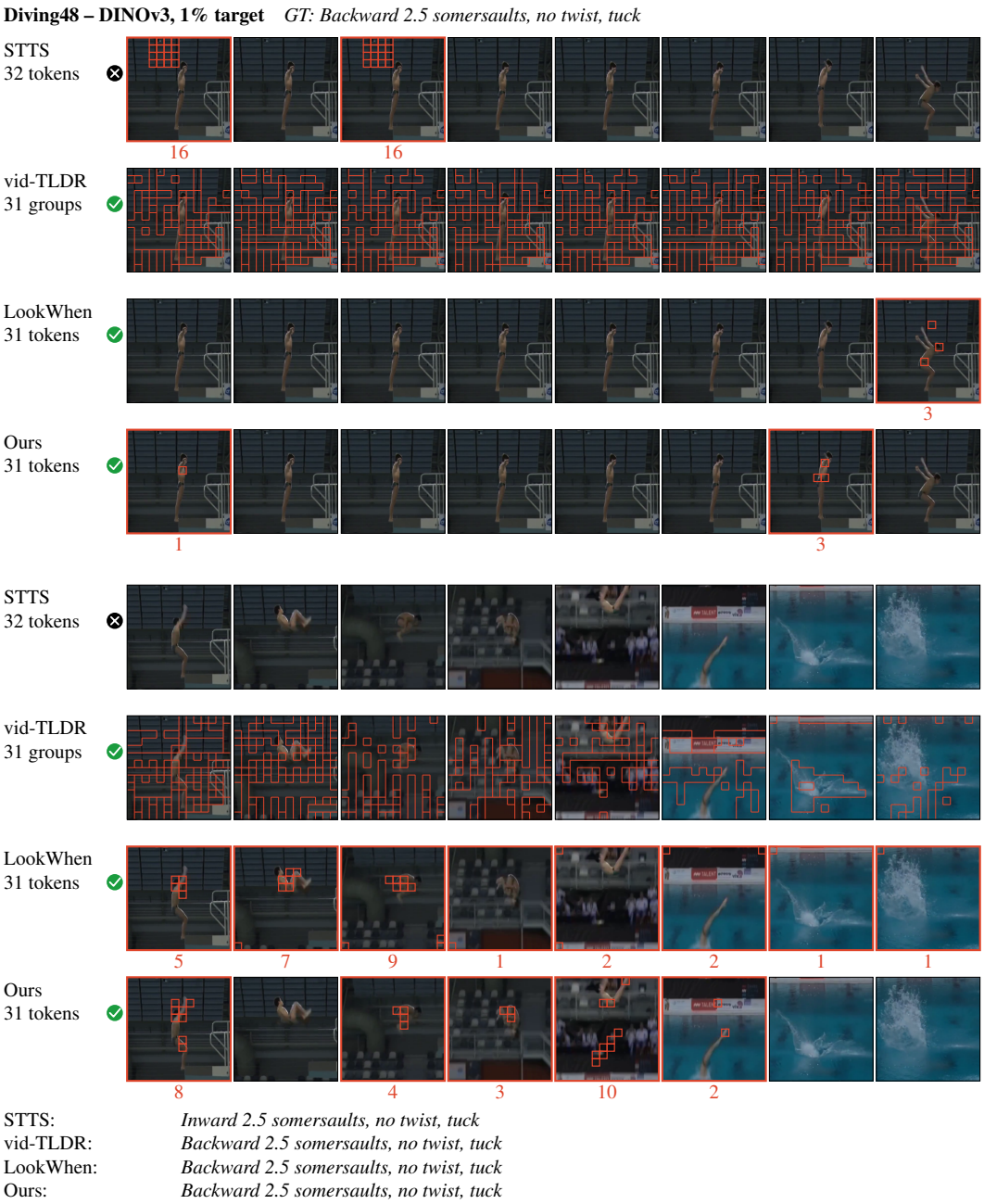


Figure 13: **DINOv3 on Diving48 at a 1% target**

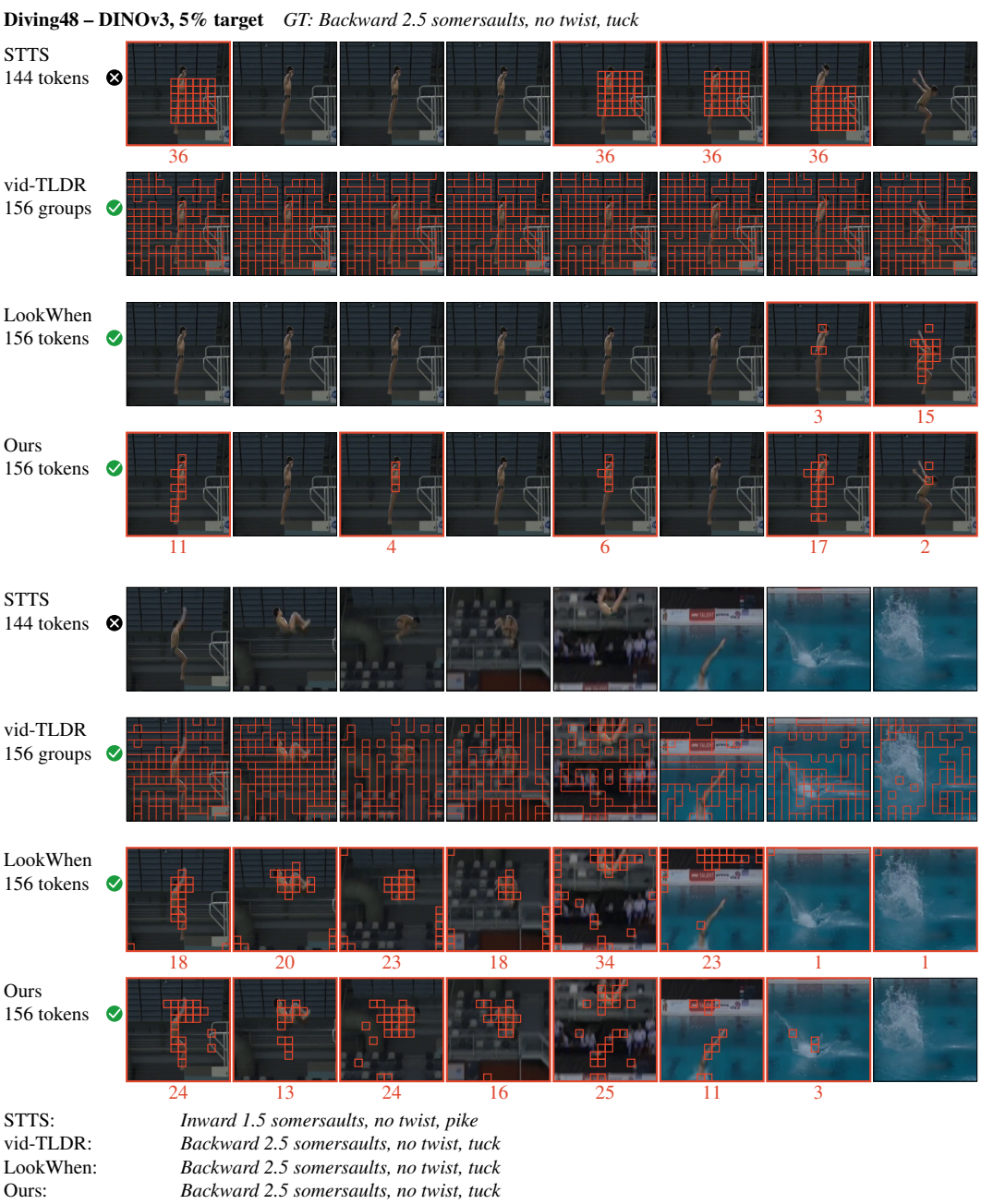


Figure 14: **DINOv3 on Diving48 at a 5% target**

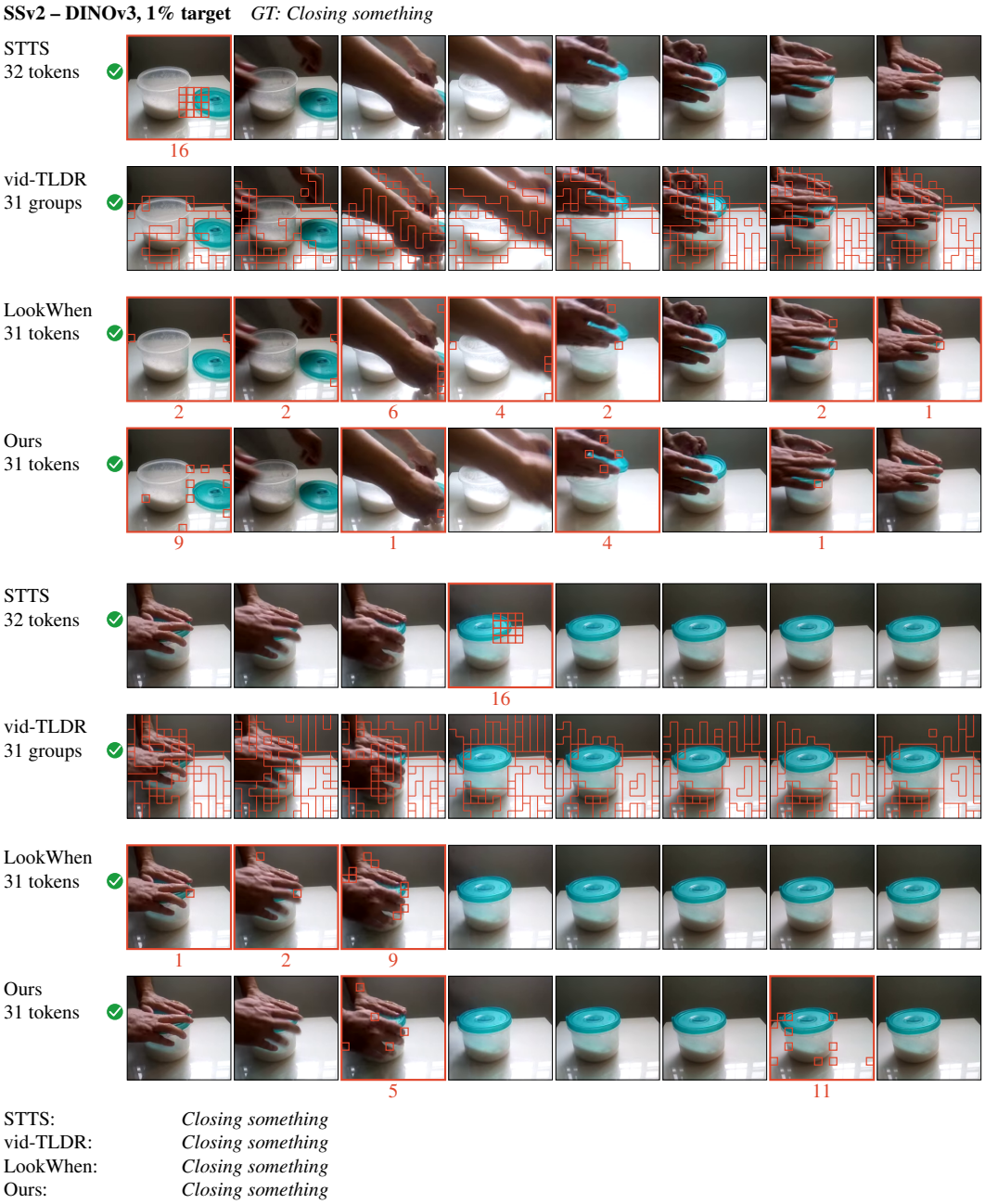


Figure 15: **DINOv3 on SSv2 at a 1% target**

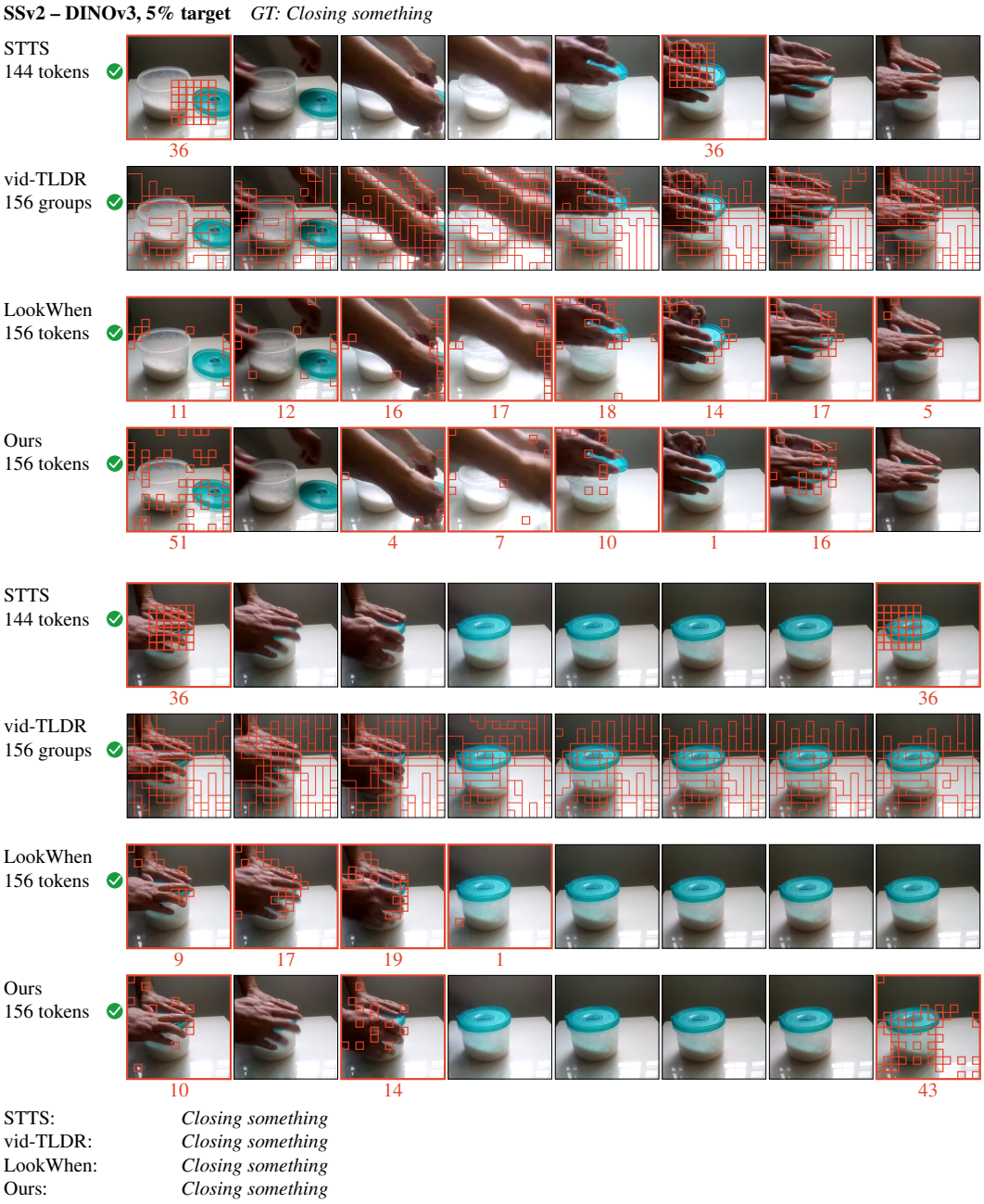


Figure 16: **DINOv3 on SSv2 at a 5% target**



Figure 17: **DINOv3 on Kinetics-400 at a 1% target**



Figure 18: **DINOv3 on Kinetics-400 at a 5% target**

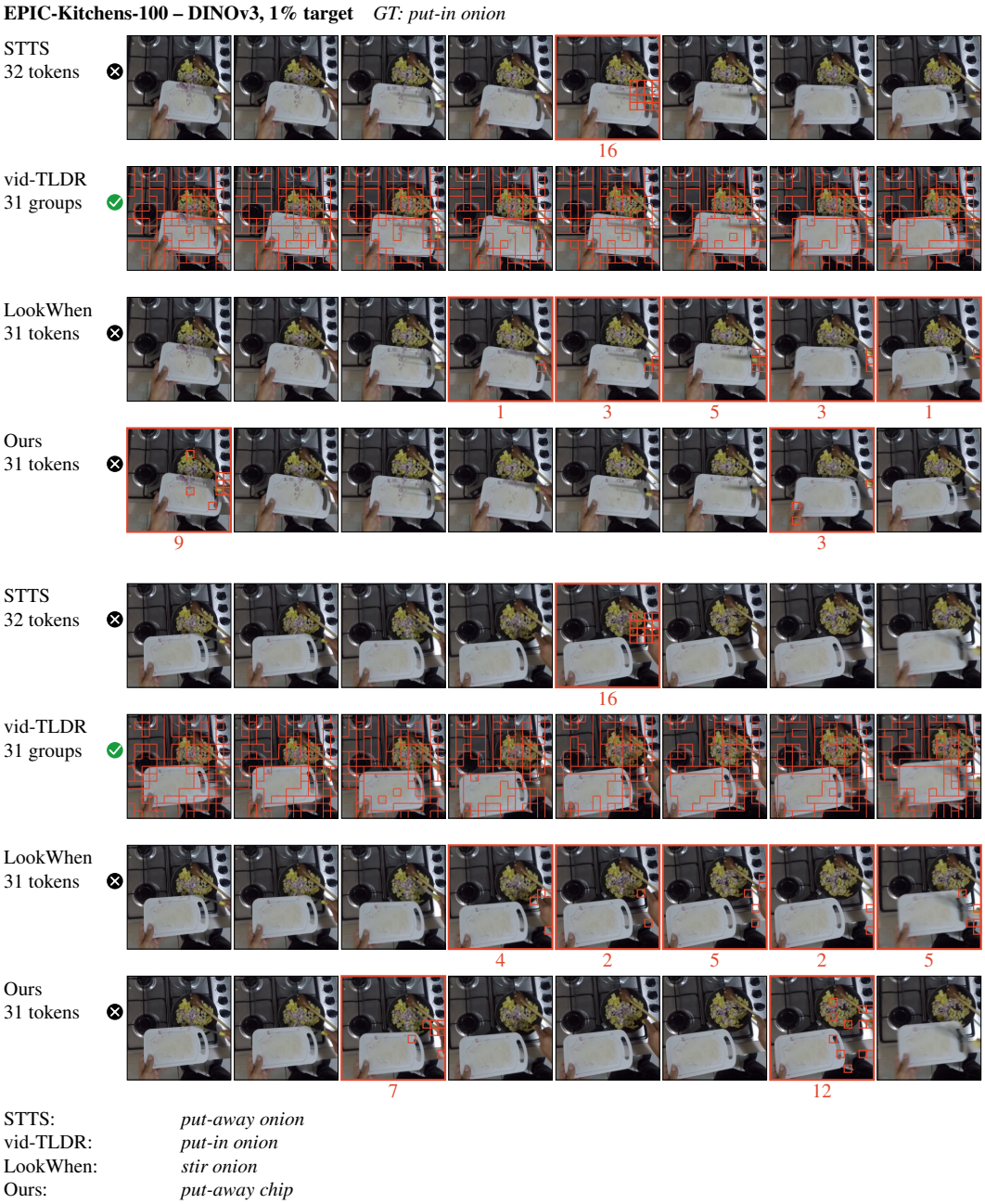
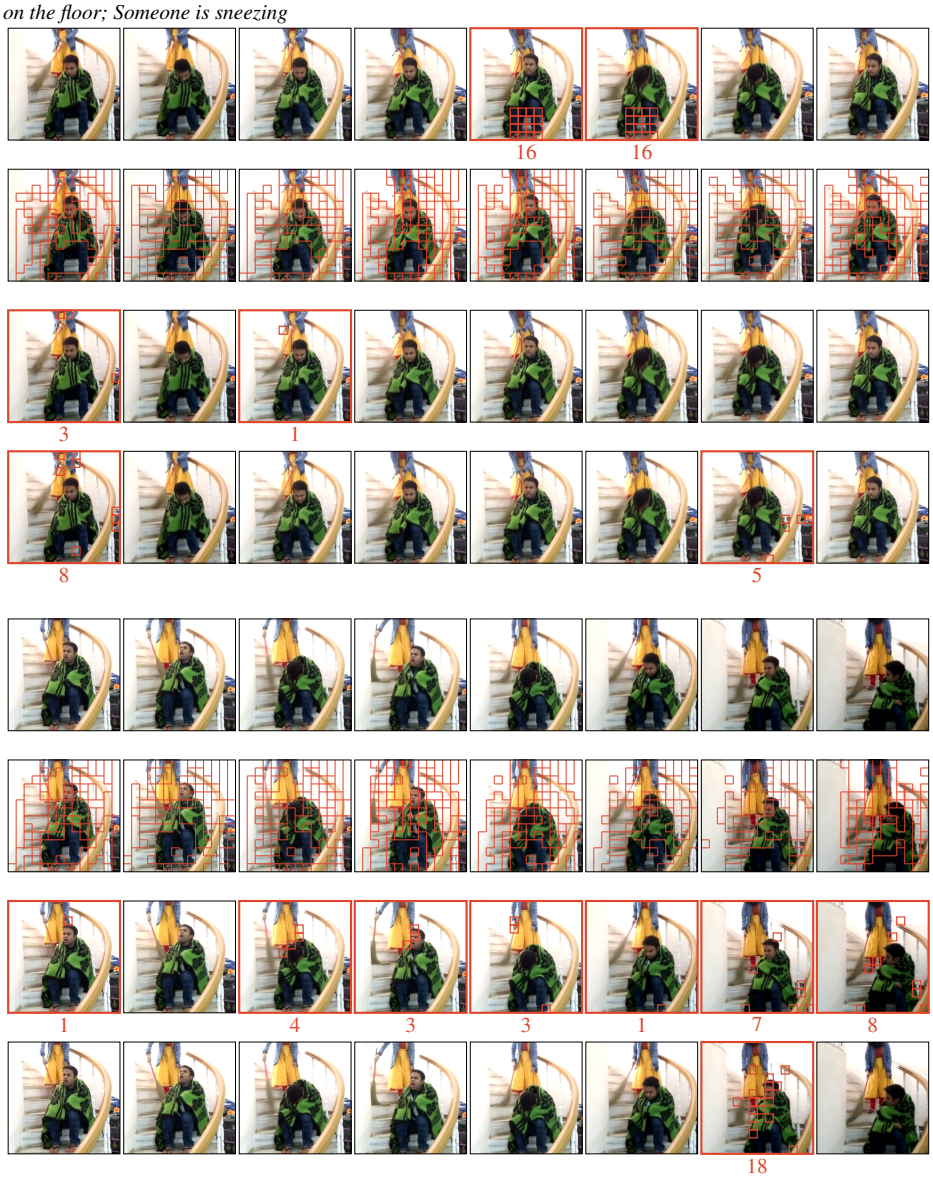


Figure 19: **DINOv3 on EPIC-Kitchens-100 at a 1% target**



Figure 20: **DINOv3 on EPIC-Kitchens-100 at a 5% target**

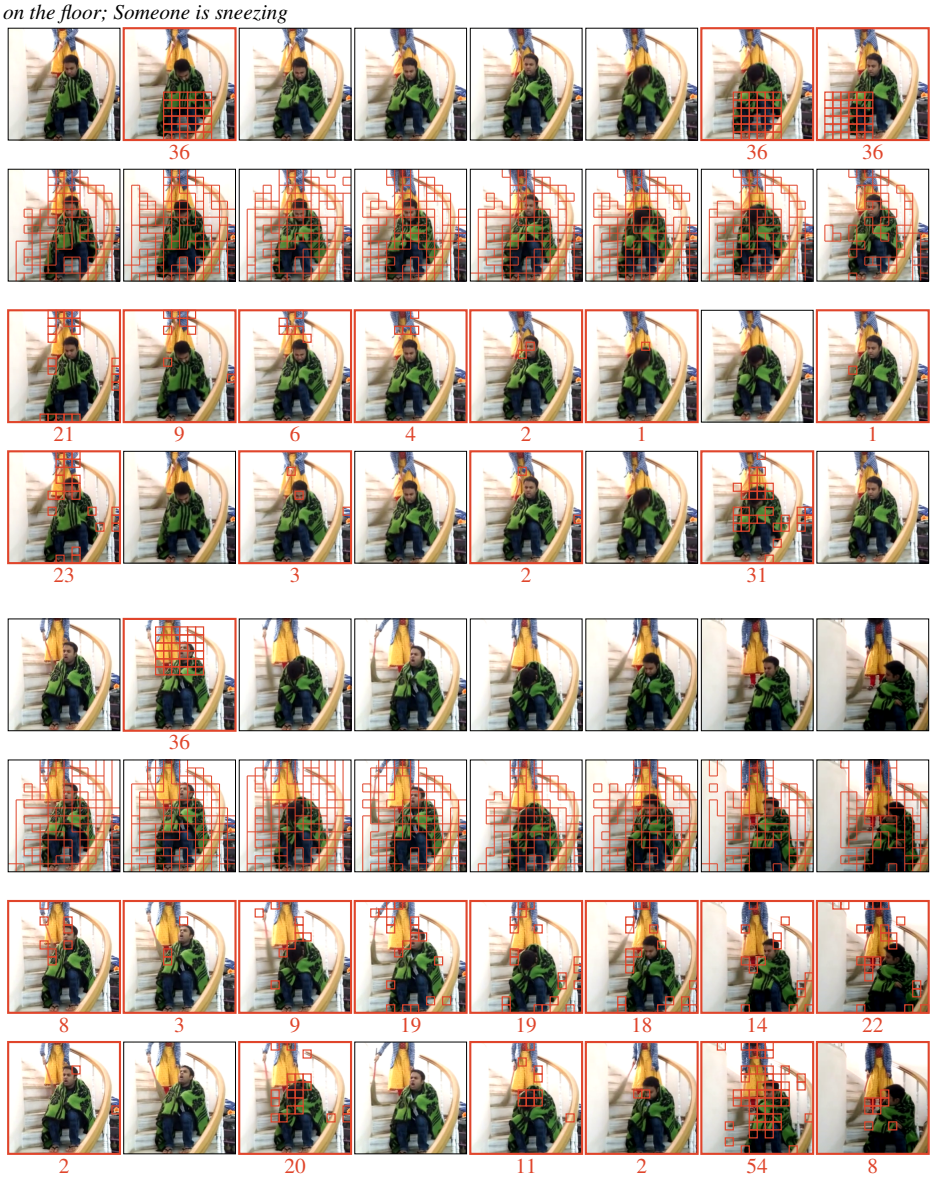
Charades – DINOv3, 1% target *GT: Holding a blanket; Snuggling with a blanket; Holding a broom; Tidying up with a broom; Sitting on the floor; Someone is sneezing*



STTS: *Ranked top 3: Someone is standing up from somewhere; Sitting in a chair; Holding a phone/camera*
vid-TLDR: *Ranked top 3: Snuggling with a blanket; Someone is eating something; Holding some food*
LookWhen: *Ranked top 3: Someone is standing up from somewhere; Sitting on the floor; Snuggling with a blanket*
Ours: *Ranked top 3: Snuggling with a blanket; Holding a blanket; Holding a dish*

Figure 21: **DINOv3 on Charades at a 1% target**

Charades – DINOv3, 5% target *GT: Holding a blanket; Snuggling with a blanket; Holding a broom; Tidying up with a broom; Sitting on the floor; Someone is sneezing*



STTS: *Ranked top 3: Snuggling with a blanket; Holding a blanket; Sitting on the floor*
vid-TLDR: *Ranked top 3: Snuggling with a blanket; Holding a blanket; Sitting on the floor*
LookWhen: *Ranked top 3: Snuggling with a blanket; Someone is standing up from somewhere; Sitting on the floor*
Ours: *Ranked top 3: Sitting on the floor; Snuggling with a blanket; Holding a blanket*

Figure 22: **DINOv3 on Charades at a 5% target**

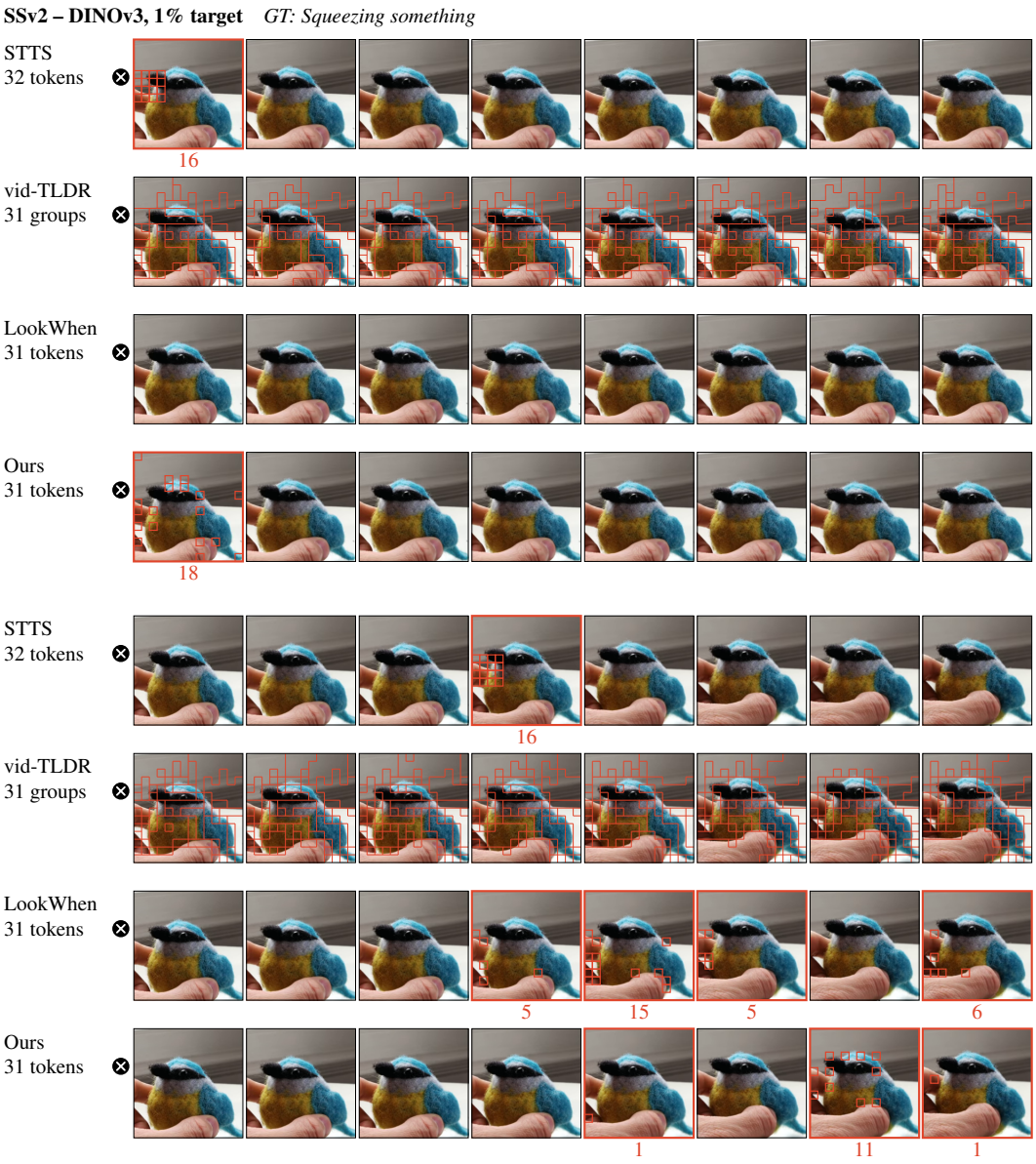


Figure 23: **DINOv3 on SSv2 at a 1% target**

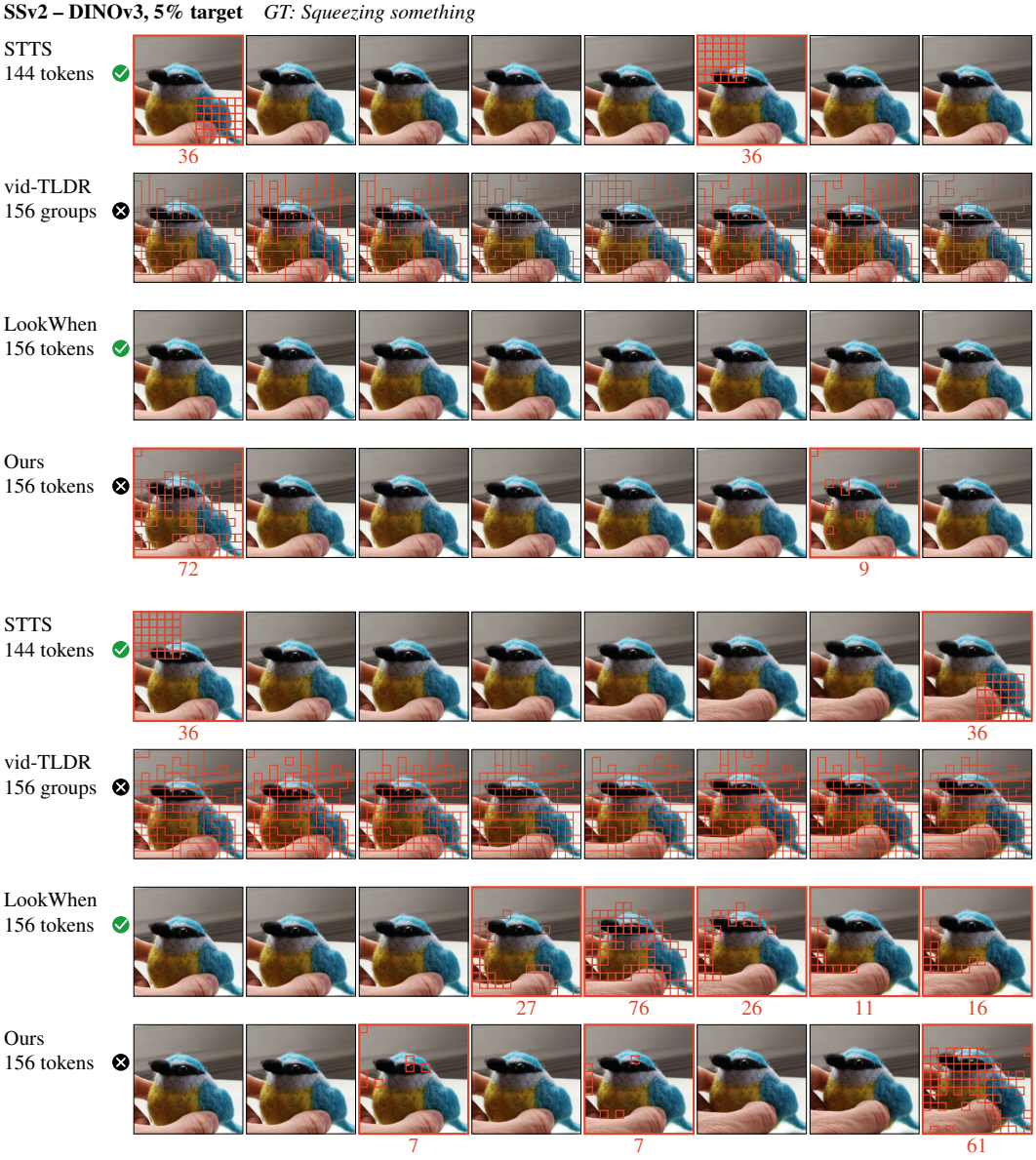


Figure 24: **DINOv3 on SSv2 at a 5% target**

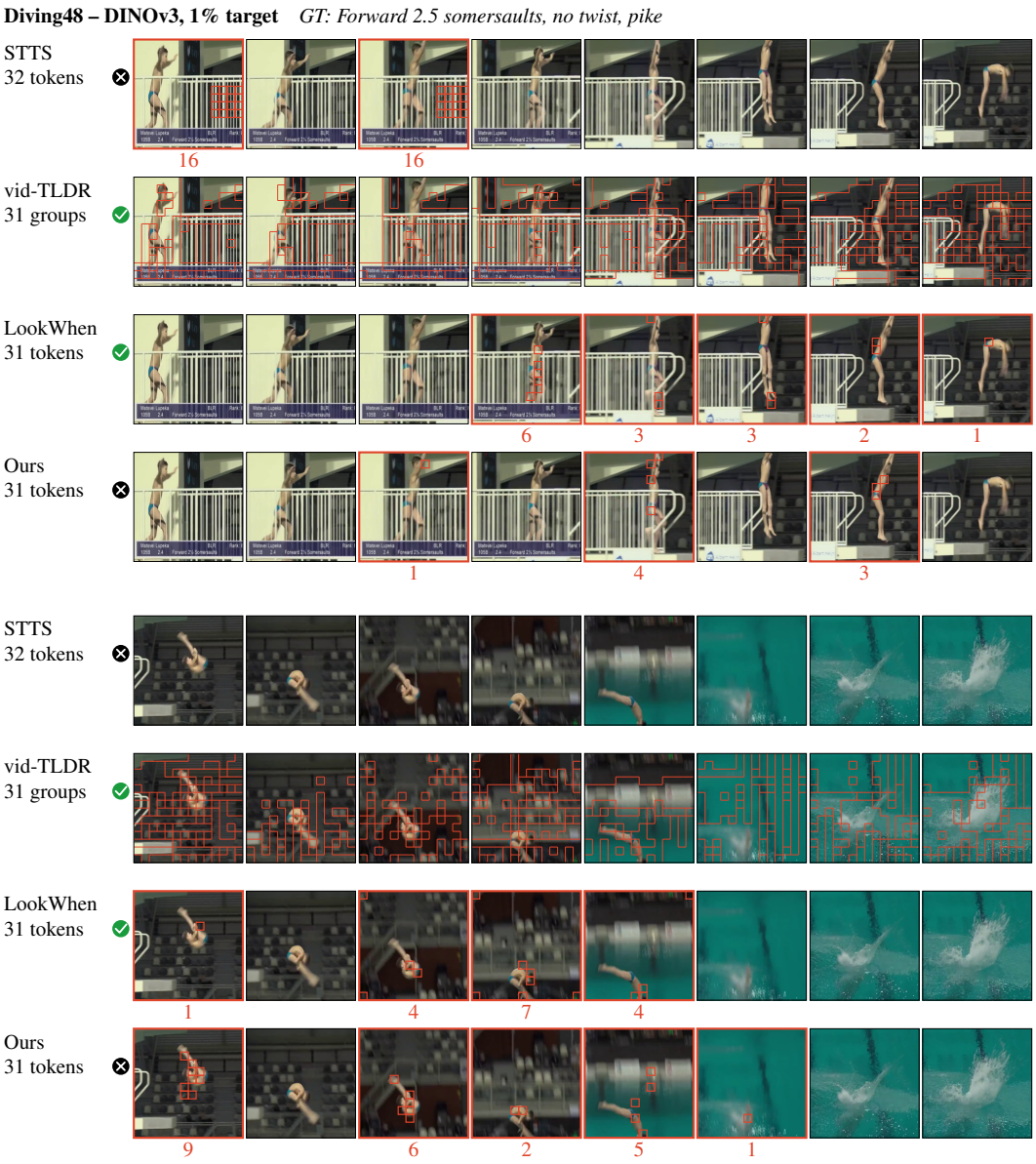


Figure 25: **DINOv3 on Diving48 at a 1% target**

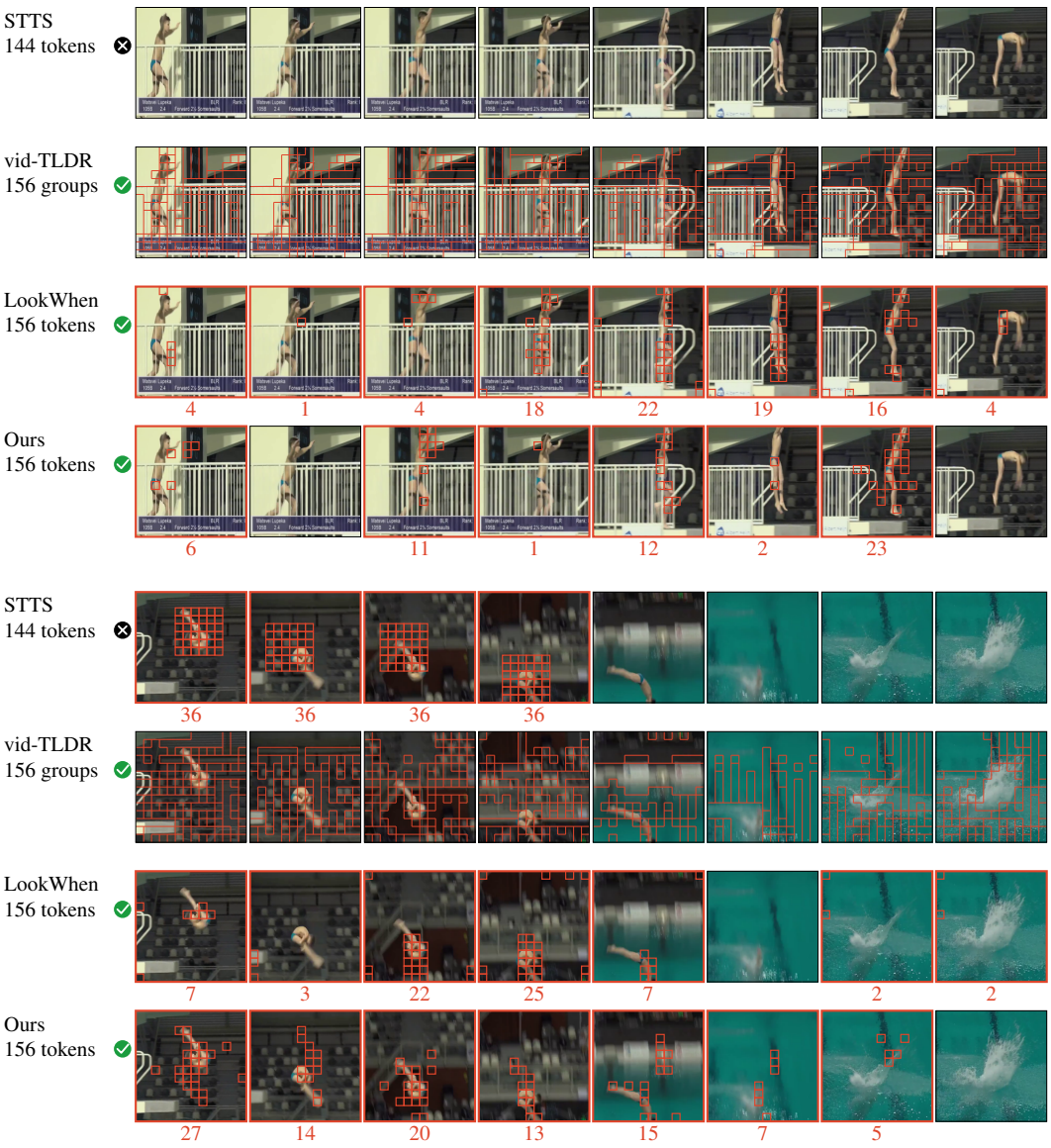


Figure 26: **DINOv3 on Diving48 at a 5% target**

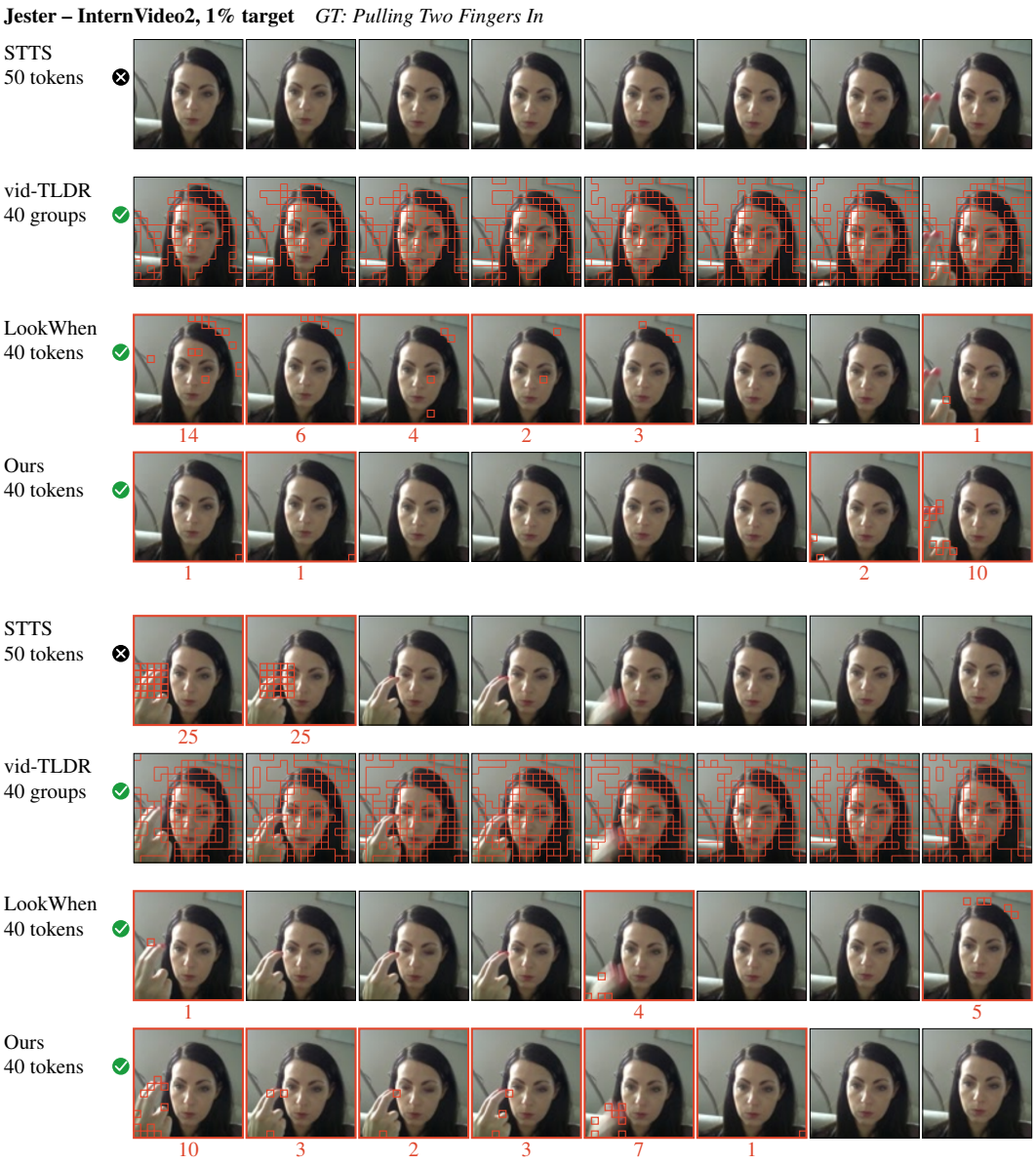


Figure 27: **InternVideo2 on Jester at a 1% target**

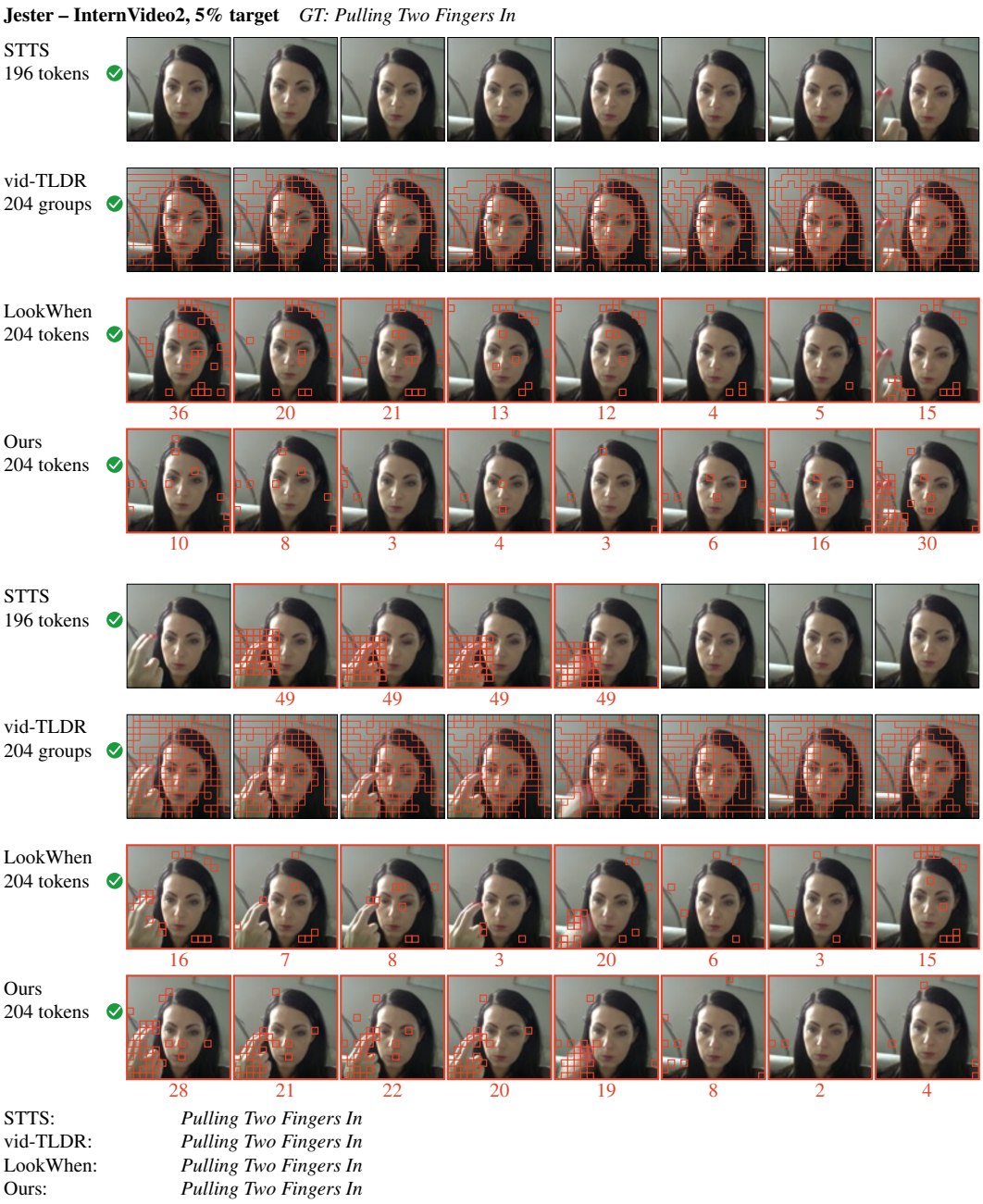


Figure 28: **InternVideo2 on Jester at a 5% target**

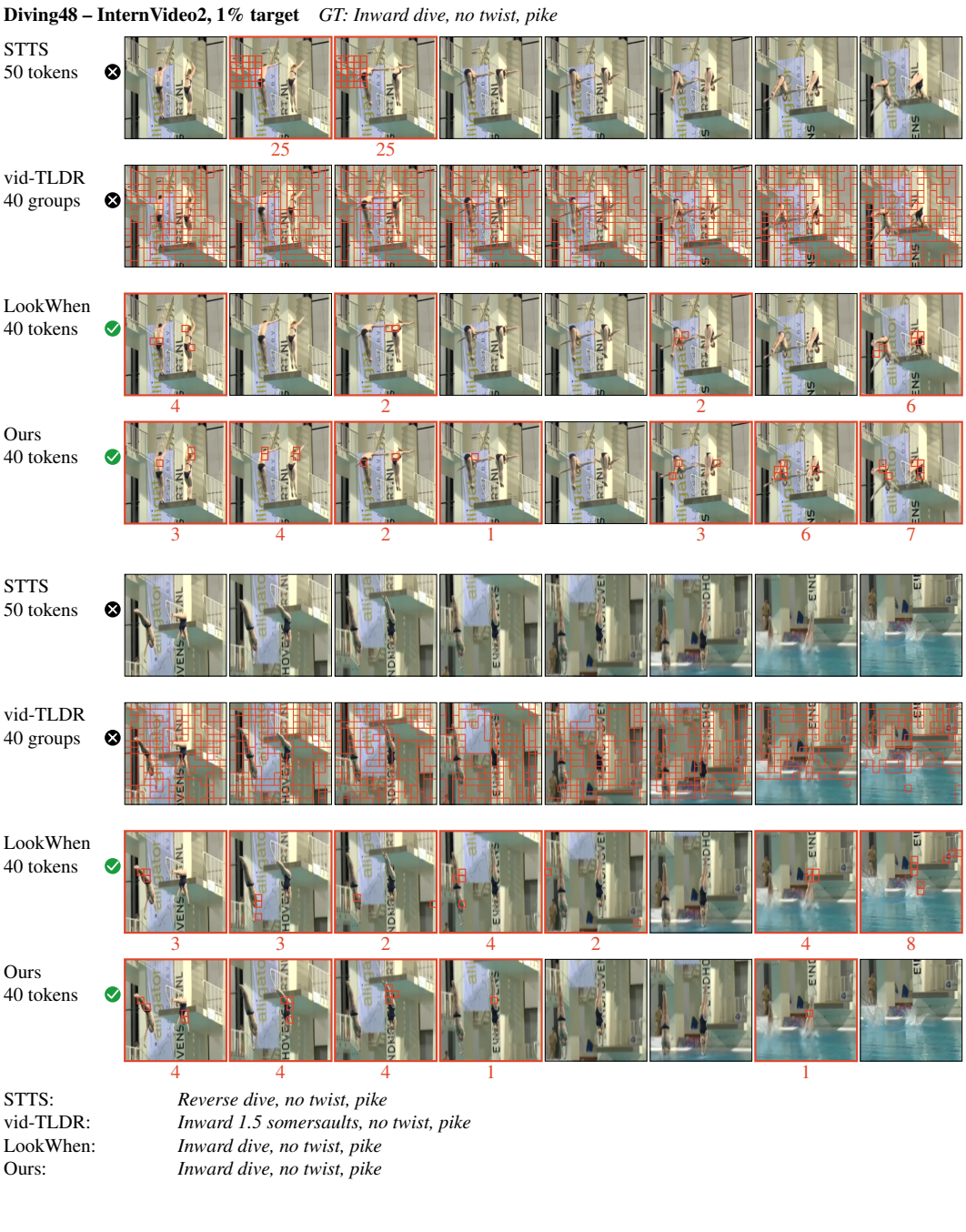


Figure 29: **InternVideo2 on Diving48 at a 1% target**



Figure 30: **InternVideo2 on Diving48 at a 5% target**



Figure 31: **VideoMAEv2 on Jester at a 1% target**



Figure 32: **VideoMAEv2 on Jester at a 5% target**

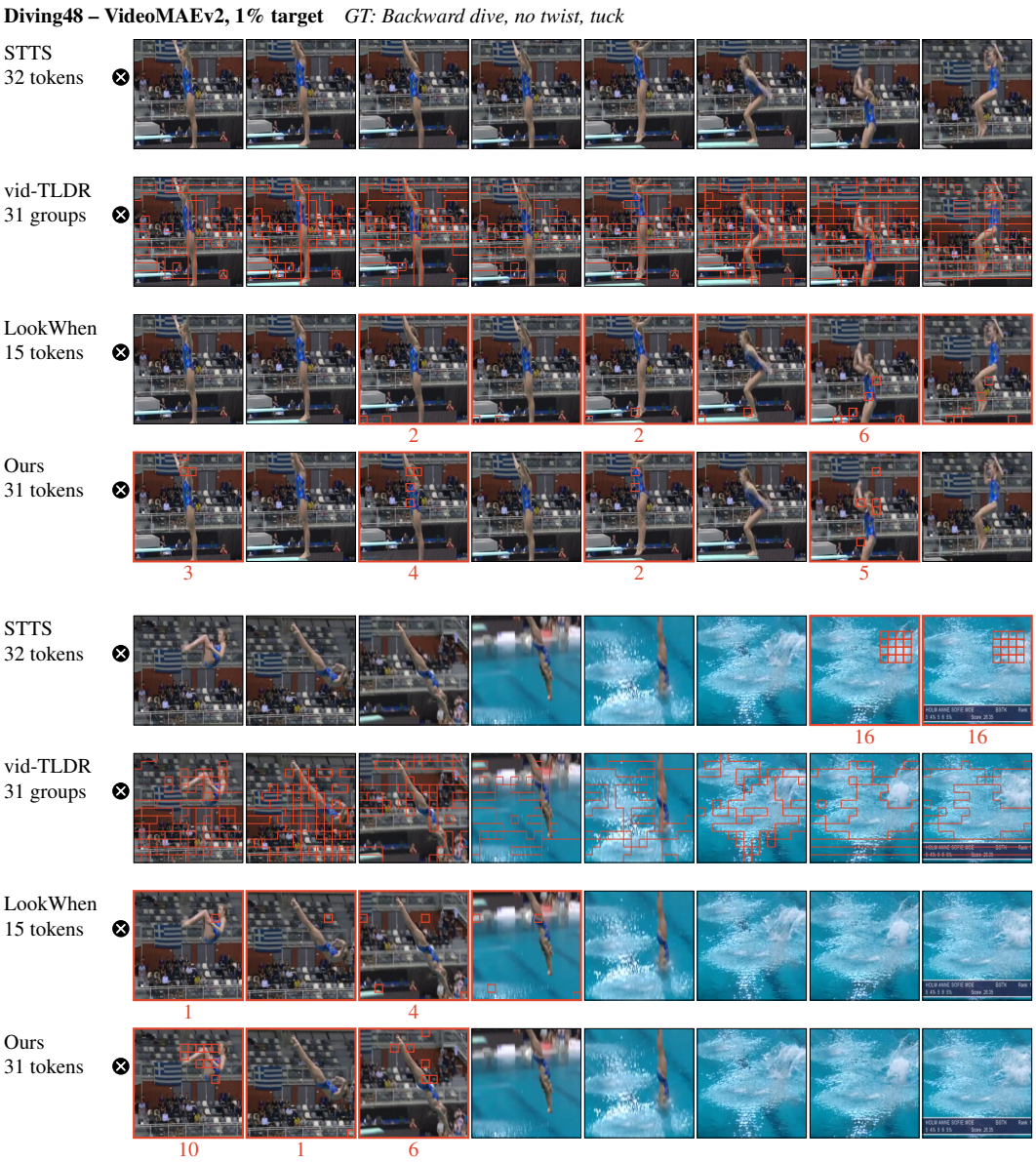


Figure 33: **VideoMAEv2 on Diving48 at a 1% target**

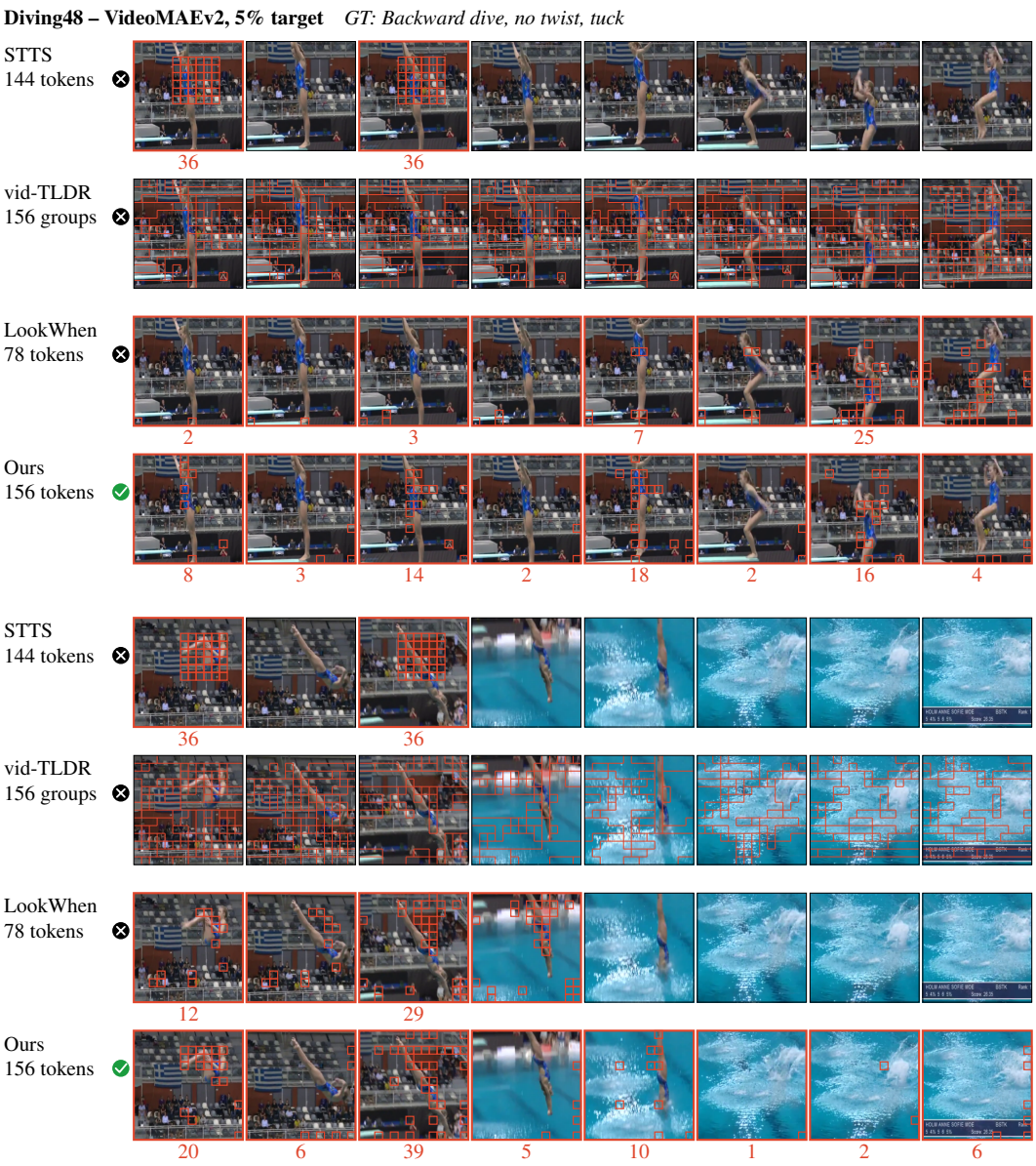


Figure 34: **VideoMAEv2 on Diving48 at a 5% target**