# SAVOR: SAMPLE-AWARE VIDEO TOKEN REDUCTION WITH NATIVE VARIABLE-LENGTH EXECUTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Video Transformers spend much of their computation on temporally redundant tokens: nearby frames are often very similar, and static backgrounds do not change across frames. Many token-reduction methods either keep the same number of tokens for every sample, or keep masked tokens in the computational graph during training. The former ignores per-sample redundancy, while the latter does not shorten the actual sequences that are fed to attention layers. To address these issues, we propose **S**ample-**A**ware **V**ideo t**O**ken **R**eduction (SAVOR), which operates on a variable-length transformer backbone. It flattens all clips in a batch into one sequence with cumulative lengths and uses a per-sample attention mask to prevent cross-sample attention, so that the removed tokens can be physically deleted during both training and inference. SAVOR combines three components: a deterministic PreMerge that merges redundant temporally adjacent tokens, a CLS-conditioned scorer that keeps the highest-scoring tokens, merges low-scoring neighbors, and drops the rest, and a Hierarchical FLOPs Allocator that turns a global FLOPs target into per-sample, per-layer retention ratios. We evaluate SAVOR on two video benchmarks, i.e., EPIC-KITCHENS-100 (EK-100) and Kinetics-400 (K400). On EK-100 multi-instance retrieval, SAVOR reduces the analytical encoder FLOPs proxy by 30.3%, achieves 60.74 mAP versus 60.71 for the dense backbone, and improves nDCG by 0.37 percentage points. Across the EK-100 target sweep, the realized proxy ratio differs from the target by less than 2% in relative terms. Code is released at `https://anonymous.4open.science/r/SAVOR`.

## 1 INTRODUCTION

Video Transformers adapt the ViT (Dosovitskiy et al., 2021) architecture to model both spatial and temporal dimensions (Bertasius et al., 2021; Jin et al., 2024; Assran et al., 2025). The temporal dimension multiplies the token count, and self-attention is quadratic in that count, so the cost grows quickly with the length of a clip. However, this growth is particularly wasteful for video data, where much of the content is redundant and only a small fraction of tokens are truly informative.

Frame-level reduction treats each frame as a unit (Buch et al., 2025). Either the whole frame survives or none of it does, so small informative regions cannot be preserved selectively. In video data, discriminative evidence like hand–object interactions often concentrates in small regions of an otherwise static frame (Zhang et al., 2023). Dropping the frame discards the key evidence, while keeping it retains redundant background tokens. Token-level reduction avoids this dilemma: informative feature tokens are preserved, static background ones are removed, and redundant ones across nearby frames are merged rather than dropped.

Most existing token-reduction methods decide *which* tokens to keep via learned scoring or similarity (Rao et al., 2021; Yin et al., 2022; Liang et al., 2022; Bolya et al., 2023), but leave the budget, *how many* to keep, fixed at training time. A few methods do vary the budget, but only along one axis at a time, either across inputs or across layers. To our knowledge, none of them varies both while holding to a single global FLOPs target, and none shortens the sequence physically within the encoder during training (Fayyaz et al., 2022; Chen et al., 2023; Zhang et al., 2026). Meanwhile, recent post-ViT reducers (Chen et al., 2024; Shang et al., 2025; Hyun et al., 2025; Fu et al., 2025) compress the visual-token stream after the encoder, leaving the token counts inside the encoder untouched.



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.

**Design goals.** Closing this gap requires rethinking token reduction across three dimensions: *Signal separation* ($\mathcal{C}1$): redundancy, measured by feature similarity, and importance, measured by attention, are distinct quantities and should not be handled by the same action. *Physical execution* ($\mathcal{C}2$): deleting tokens must physically reduce FLOPs during both training and inference, rather than merely masking them in the computation graph. *Conditional budgeting* ($\mathcal{C}3$): the budget should depend on the video content and the layer depth, under a single global FLOPs target instead of a schedule fixed before training.

To meet $\mathcal{C}2$, we rebuild the EVA backbone (Fang et al., 2024; Wang et al., 2026) on a packed representation with per-sample attention masks, i.e. samples are concatenated into one sequence with cumulative lengths, and attention operates within each sample but not across sample boundaries. This design addresses two limitations: First, sample-aware token reduction produces variable-length outputs that cannot form equal-length batches. Second, sample boundaries can be passed through FlashAttention's variable-length interface using cumulative sequence lengths, allowing attention to operate independently within each sample (Dao et al., 2022; Shah et al., 2024). Prior packing methods (Dehghani et al., 2023; Wang et al., 2024) determine token count from the input and keep it fixed throughout all layers. SAVOR instead removes tokens after each block and repacks the sequence with updated cumulative lengths, physically deleting removed tokens from subsequent computation and saving FLOPs. The same backbone makes $\mathcal{C}3$ executable: per-sample budgets become per-sample lengths.

The SAVOR reduction policy implements $\mathcal{C}1$ and $\mathcal{C}3$ as three components: A deterministic **PreMerge** merges temporally adjacent tokens whose cosine similarity exceeds a threshold, capturing visible redundancy before the scorer evaluates the remaining tokens. An **Adaptive Token Reduction (ATR)** module scores each token with a lightweight CLS-conditioned head, then keeps high-scoring tokens, merges low-scoring neighbors, and drops the rest. A **Hierarchical FLOPs Allocator (HFA)** converts a global target $\tau$ into per-sample and per-layer retention ratios, so that the reduction intensity adapts to both the video content and the layer depth. Training follows the same hard discrete path as inference via straight-through estimators (STEs) (Bengio et al., 2013).

We evaluate SAVOR on two video benchmarks: EK-100 (Damen et al., 2022; 2018) and K400 (Carreira & Zisserman, 2017). SAVOR achieves substantial FLOPs reduction with minimal accuracy loss compared with the dense backbone, and the realized proxy ratio closely tracks the target across a range of $\tau$ values.

## 2 Related Work

**Token selection and budget learning.** Token-reduction methods make two decisions, i.e., which tokens to preserve and how many to preserve. For the first decision, DynamicViT (Rao et al., 2021) and EViT (Liang et al., 2022) sort tokens by their attention scores and retain the most-attended ones: DynamicViT drops low-attention tokens, whereas EViT merges them into a single aggregated token so their information is preserved. While these methods make a one-time selection, A-ViT (Yin et al., 2022) takes a progressive approach: each token accumulates a score layer by layer, and once the score crosses a threshold, the token stops contributing to all subsequent layers. A separate family of methods, including ToMe (Bolya et al., 2023) and TokenLearner (Ryoo et al., 2021), avoids selection entirely: they keep all tokens but compress them via merging (ToMe) or learnable spatial pooling (TokenLearner), reducing the sequence length without discarding any single token outright.

The second, how many tokens to preserve, is usually fixed in advance. ATS (Fayyaz et al., 2022) is the main exception: with no learned parameters, it scores tokens by relevance and samples a subset, so different inputs naturally retain different numbers of tokens. DiffRate (Chen et al., 2023) instead learns per-layer compression rates via gradient descent rather than setting them manually, but the resulting rates are shared across all inputs and frozen after training. By contrast, SAVOR addresses both decisions jointly: it selects which tokens to keep via PreMerge and ATR, and determines how many via the per-sample, per-layer retention ratios of HFA, while physically removing tokens during both training and inference.

**Variable-length execution.** Most token reduction methods above assume that each sample keeps a fixed number of tokens, so they can simply be batched in the standard way without any packing. NaViT (Dehghani et al., 2023) and Qwen2-VL (Wang et al., 2024) go a step further by allowing

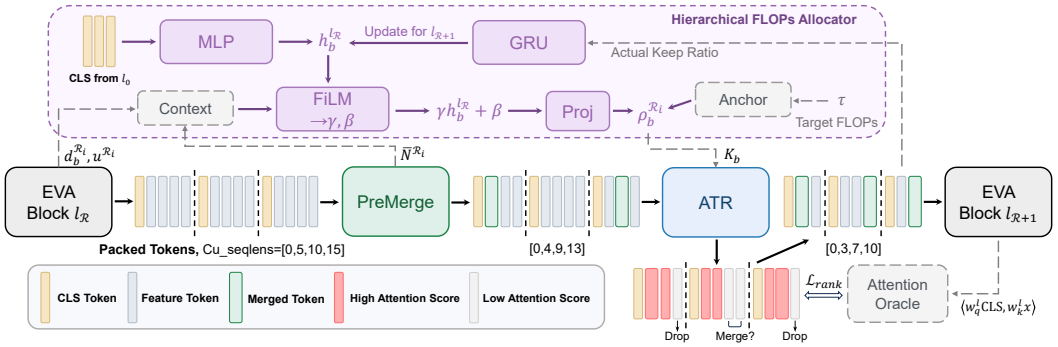


Figure 1: Overview of the SAVOR architecture at a reduction stage. A packed multi-sample sequence (per-sample token counts annotated above each segment) is shortened first by PreMerge (green), which merges temporally adjacent feature tokens based on a similarity threshold, and then by the ATR module (blue), which keeps, merges, or drops tokens by importance. The HFA (purple) turns the global FLOPs target $\tau$ into a budget for each sample, and adjusts it at later stages using the retention it has already observed. The sequence is repacked and then passed to the next block.

samples of different sizes within the same batch through packing. Specifically, NaViT combines patches from multiple images at native resolution into one sequence, masking attention so that each image only attends to itself, and Qwen2-VL varies how many tokens an image produces based on its resolution. Neither, however, changes the token count of a sample after it is fixed at the input: the number is set before the forward pass and stays constant through all layers. SAVOR instead determines the token count of a sample during the forward pass. At each reduction stage, HFA allocates a retention ratio, and the reduction modules decide which tokens to keep, merge, or drop before repacking the sequence with updated cumulative lengths.

## 3 Method

Figure 1 shows the SAVOR pipeline, whose design follows the goals above: redundancy and importance are distinct criteria for removal, and neither on its own determines how many tokens should be removed. At each reduction stage $R_i$, PreMerge (Section 3.2.1) handles redundancy by merging temporally adjacent tokens with a cosine similarity-based threshold. Sample-aware ATR (Section 3.2.3) scores the remaining tokens by importance, keeping the highest-scoring ones, merging low-scoring neighbors, and dropping the rest. The Hierarchical FLOPs Allocator (HFA, Section 3.2.2) determines how many tokens to retain by converting the global FLOPs target $\tau$ into per-sample retention ratios $\rho_b^{R_i}$, which in turn determine the target token count $K_b$ at each stage. The entire process is implemented on a variable-length backbone that packs samples into one sequence and repacks after each reduction stage. Appendix D.1 gives the whole pass in execution order.

### 3.1 Problem Setup and Variable-Length Execution

**Problem setup.** Consider a batch of video samples indexed by $b \in \mathcal{B}$, $X \in \mathbb{R}^{|\mathcal{B}|\times C\times T\times H\times W}$, where $C$ is the channel count, $T$ the number of frames, and $H, W$ the frame height and width. Spatial patch embedding with patch size $p$ produces $D$-dimensional features and $N_0 = 1 + T(H/p)(W/p)$ tokens per sample. Here $\boldsymbol{c}_b \in \mathbb{R}^D$ denotes the CLS token for sample $b$. The other $N_0 - 1$ tokens are feature tokens with spatio-temporal coordinates. Token lengths written below include the CLS token, whereas counts of feature tokens exclude it.

The backbone has $L$ blocks, of which the first $c+1$ apply no reduction. The first reduction stage is $R_1$ before block $\ell_{c+1}$, and stage $R_i$ is before block $\ell_{c+i}$, with $R = L-c-1$. For sample $b$, we track the token count at three points in each stage, i.e., $N_b^{R_i}$ before the stage, $N_b^{R_i,\text{pm}}$ after PreMerge, and $N_b^{R_i,\text{atr}}$ after ATR, which is also the length entering the consuming block $\ell_{c+i}$. Since EVA blocks do not change length, $N_b^{R_i,\text{atr}} = N_b^{R_{i+1}}$ for $i < R$. Full notation is listed in Appendix B.

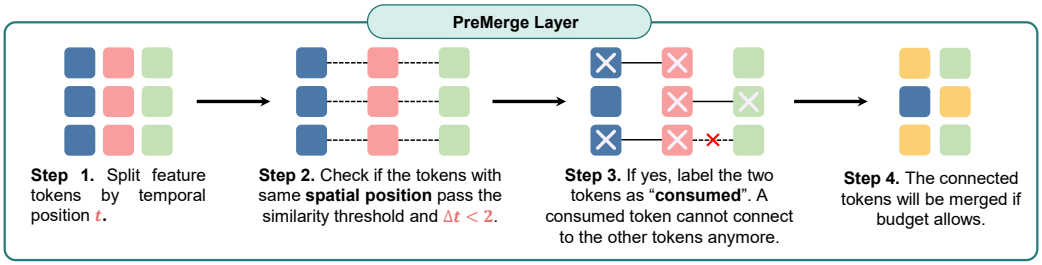


Figure 2: Illustration of the PreMerge temporal-redundancy merging pipeline. Within each spatial position $(h, w)$, temporally adjacent tokens with high cosine similarity are merged, reducing token count from $N_b^{R_i}$ to $N_b^{R_i,\mathrm{pm}}$ before sample-aware token selection.

The global FLOPs target $\tau$ is specified as a fraction of the dense EVA-block cost. We report the realized proxy ratio as the analytical FLOPs proxy $\Phi(X)$ evaluated on the hard token counts (Appendix D.4).

**Variable-length execution.** Samples are packed into one tensor at their current lengths $N_b$, with boundaries marked by cumulative lengths. This allows FlashAttention's variable-length (varlen) interface (Dao et al., 2022; Shah et al., 2024) to operate on the packed sequence, preventing attention across different samples, so the total attention cost scales as $\sum_b N_b^2$ rather than $|\mathcal{B}|(\max_b N_b)^2$. At each stage $R_i$, the sequence lengths are rewritten: any token that is merged or dropped is physically removed from the tensor and does not enter the subsequent block.

To support merging, each feature token carries a continuous temporal position $t$, a spatial position $(h, w)$, and a merge mass $\mu$ initialized to one. The mass tracks how many original feature tokens it represents. When merging two feature tokens $j$ and $k$ with feature vectors $\boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^D$, they share the same $(h, w)$ and are updated as

$$\mu' = \mu_j + \mu_k, \qquad \boldsymbol{x}' = \frac{\mu_j \boldsymbol{x}_j + \mu_k \boldsymbol{x}_k}{\mu'}, \qquad t' = \frac{\mu_j t_j + \mu_k t_k}{\mu'}. \tag{1}$$

The CLS token carries no merge metadata and is never merged or dropped. The EVA02-AT backbone uses three-dimensional spatio-temporal rotary positional embeddings (3D ST-RoPE) (Su et al., 2024; Wang et al., 2026). After a merge, the embedding is regenerated from the token metadata $(t', h, w)$ rather than interpolated from the source positions, so a merged token carries the positional embedding of its own mass-weighted temporal position.

## 3.2 SAVOR Token Reduction

### 3.2.1 Temporal PreMerge

PreMerge is a deterministic pre-processing step that precedes ATR at each reduction stage. It scans the features produced by the previous block and merges temporally adjacent tokens that share the same spatial position $(h, w)$. A pair of tokens is admissible for merging when their cosine similarity exceeds a threshold $\delta$ and their temporal positions are less than two frames apart ($|\Delta t| < 2$). The merge itself follows the centroid update in Eq. 1.

PreMerge works in two steps, as Figure 2 shows. The first step groups the tokens by spatial position and orders each group in time, then identifies the admissible pairs. The second step merges them. Once a pair is merged, its two tokens are consumed and the resulting token can take part in later merges. To bound over-aggressive early compression, PreMerge is not allowed to reduce a stage below a retention floor $\pi_{\mathrm{floor}}$, i.e. $(N_b^{R_i,\mathrm{pm}} - 1)/(N_b^{R_i} - 1) \geq \pi_{\mathrm{floor}}$.

### 3.2.2 Hierarchical FLOPs Allocator

After PreMerge, HFA determines how many tokens ATR should retain at the current stage, in the form of a per-sample retention ratio $\rho_b^{R_i}$.

The budgets are not chosen independently at each stage. All stages share one anchor $\rho_a$, the fraction of post-PreMerge feature tokens that would reproduce the global FLOPs target $\tau$ on average, and every $\rho_b^{R_i}$ is a bounded deviation from it. Without that anchor, each stage is free to set its own level, and the reduction then falls entirely on two early stages rather than spreading across depth. The anchor is calibrated at initialization and re-calibrated at epoch boundaries (Appendix D.5), which also records the unbalanced length schedule we observed without it.

**Allocator state.** A per-sample state $\mathbf{h}_b^i$ carries the compression history of each sample across stages, and the allocator turns it into $\rho_b^{R_i}$ at every stage. The state starts from the CLS feature of block $\ell_c$ through a learnable projection, $\mathbf{h}_b^1 = W_c \boldsymbol{c}_b^{(\ell_c)}$. Appendix E.2 gives the routing of each loss to the parameters it reaches.

At stage $R_i$, the allocator takes three detached scalars as context:

$$
\begin{aligned}
u^i &= \frac{i-1}{R-1}, \qquad r_{b,\mathrm{pm}}^i = \frac{N_b^{R_i,\mathrm{pm}} - 1}{\max\{1, N_b^{R_i} - 1\}}, \\
d_b^i &= \mathrm{clip}_{[-1,1]}\left(\frac{N_b^{R_i} - \bar{N}^{R_i}}{N_0}\right), \qquad \bar{N}^{R_i} = 1 + (N_0 - 1)\rho_a^{i-1}.
\end{aligned}
\tag{2}
$$

Here $u^i$ is the stage progress from 0 to 1, $r_{b,\mathrm{pm}}^i$ is the fraction of feature tokens retained by PreMerge, and $d_b^i$ is the compression debt, measuring how far the current length $N_b^{R_i}$ deviates from the anchor-projected length $\bar{N}^{R_i}$.

These signals modulate the state via FiLM (Perez et al., 2018):

$$
[\boldsymbol{\gamma}, \boldsymbol{\beta}] = W_f\, \mathbf{z}_b^i + \boldsymbol{b}_f, \qquad \mathbf{z}_b^i = [u^i, d_b^i, r_{b,\mathrm{pm}}^i]^\mathsf{T}, \tag{3}
$$

and a projection head converts the modulated state to the retention ratio:

$$
\rho_b^{R_i} = \mathrm{clip}_{[\rho_{\min}, \rho_{\max}]}\left(\rho_a \left[1 + \Delta_{\pm} \tanh(\mathrm{Proj}(\boldsymbol{\gamma} \odot \mathbf{h}_b^i + \boldsymbol{\beta}))\right]\right), \tag{4}
$$

where $\Delta_{\pm}$ selects $\Delta_+$ when the learned deviation is positive and $\Delta_-$ otherwise, with

$$
\Delta_+ = \min\left\{\kappa,\ \frac{\rho_{\max} - \rho_a}{\rho_a}\right\}, \qquad \Delta_- = \min\left\{\kappa,\ \frac{\rho_a - \rho_{\min}}{\rho_a}\right\}. \tag{5}
$$

The bounds $\rho_{\min}, \rho_{\max}$ and $\kappa$ set the overall feasible retention range and the maximum relative deviation from $\rho_a$, respectively.

After ATR performs the selection, the realized feature-token retention $r_{b,\mathrm{atr}}^i = (N_b^{R_i,\mathrm{atr}} - 1)/(N_b^{R_i,\mathrm{pm}} - 1)$ updates the state via a GRU (Cho et al., 2014):

$$
\mathbf{h}_b^{i+1} = \mathrm{GRU}(\mathrm{sg}(r_{b,\mathrm{atr}}^i), \mathrm{sg}(\mathbf{h}_b^i)). \tag{6}
$$

Both inputs to the GRU are detached to prevent gradients from flowing through the full recurrence. The update is therefore one-step truncated. Further details of the allocator initialization are given in Appendix D.3.

### 3.2.3 Sample-Aware Adaptive Token Reduction (ATR)

HFA gives ATR a retention ratio $\rho_b^{R_i}$, and ATR turns it into a concrete choice of which feature tokens are kept, which are merged together, and which are discarded. Importance is judged by a lightweight scorer: an attention logit between the CLS token and each feature token. The learnable matrices $W_{sq}$ and $W_{sk}$ project the CLS and the feature tokens into query and key vectors, and the score is averaged over $H_s$ heads of dimension $d_s$. For sample $b$, feature tokens are indexed by $j = 1, \ldots, N_b^{R_i,\mathrm{pm}} - 1$ and scorer heads by $m = 1, \ldots, H_s$. The logit $s_{b,j}$ for token $j$ is

$$
s_{b,j} = \frac{1}{H_s} \sum_{m=1}^{H_s} \frac{\langle \boldsymbol{q}_{b,m}, \boldsymbol{k}_{b,j,m} \rangle}{\sqrt{d_s}} + b_s, \tag{7}
$$

where $\boldsymbol{q}_{b,m}$ is the query from the CLS, $\boldsymbol{k}_{b,j,m}$ is the key from feature token $j$, and $b_s$ is a learnable bias. All scorer inputs are detached from the backbone, so no gradient reaches the encoder through this path.

The HFA ratio becomes an integer target, the number of feature tokens to retain after ATR:

$$K_b = \min\{N_b^{R_i,\mathrm{pm}} - 1, \max\{1, \lfloor (N_b^{R_i,\mathrm{pm}} - 1)\rho_b^{R_i} \rfloor\}\}. \tag{8}$$

The merge operation in the ATR module follows the same protocol as PreMerge, i.e. a pair is admissible when the two tokens share the same spatial position $(h, w)$ and their temporal positions are strictly less than two frames apart. ATR imposes no similarity threshold. In this way, a low-scoring token can still contribute its evidence as part of a merged token rather than being dropped.

Two limits bound the merges, and the smaller of them decides how the $K_b$ retained tokens split between tokens kept directly and tokens produced by a merge. A direct-keep fraction $f$ sets the preferred count kept directly to $D_b = \lfloor fK_b \rfloor$, which leaves at most $K_b - D_b$ merges. The stage also has to remove $N_b^{R_i,\mathrm{pm}} - 1 - K_b$ tokens in total, one per merge, so it cannot merge more than that either. When the pool holds fewer admissible pairs than the binding limit allows, the stage merges fewer and ends below $K_b$. The exact accounting is in Appendix D.2.

The forward pass sorts, matches, and repacks, none of which is differentiable. The scorer and the allocator are therefore trained through two straight-through estimators that leave the forward pass untouched (Appendix E.1). The scorer is additionally trained via the ranking loss described in Section 3.3.

### 3.3 Optimization and Training

SAVOR is trained with three objectives, combined in a weighted sum: the task loss (e.g., retrieval or classification), a FLOPs penalty that encourages the realized proxy ratio to match the target $\tau$, and a ranking loss that supervises the scorer using an attention oracle.

**FLOPs penalty.** We define an analytical FLOPs proxy $\Phi(X)$ for the realized proxy ratio of a batch $X$:

$$\Phi(X) = \frac{\sum_{\ell=0}^{L-1} \mathrm{mean}_b\, \mathcal{F}_{\mathrm{blk}}(N_b^{\ell}) + \sum_{i=1}^{R} \mathrm{mean}_b\, \mathcal{F}_{\mathcal{R}}(N_b^{R_i,\mathrm{pm}})}{L\, \mathcal{F}_{\mathrm{blk}}(N_0)}. \tag{9}$$

Here $\mathcal{F}_{\mathrm{blk}}(n)$ is the cost of one EVA block with $n$ tokens, and $\mathcal{F}_{\mathcal{R}}(n)$ is the cost of the scorer (both detailed in Appendix D.4). The lengths $N_b^{\ell}$ are the actual token counts entering each block, with $N_b^{\ell_{c+i}} = N_b^{R_i,\mathrm{atr}}$.

Because the hard $\Phi(X)$ is non-differentiable, a straight-through estimator $\widehat{\Phi}_F$ replaces it for gradient purposes (Appendix E.1). The retained-token count is an integer, so the FLOPs penalty cannot differentiate it and the allocator would receive no gradient. The penalty therefore acts on a differentiable, real-valued count computed from the retention ratios instead of from the discrete decisions, which we call the soft token count. A straight-through estimator forwards the hard count and backpropagates through the soft token count (Appendix E.1). The FLOPs penalty is

$$\mathcal{L}_F = \mathrm{Huber}_{\delta_H}(\widehat{\Phi}_F - \tau). \tag{10}$$

The Huber loss (Huber, 1992) is smooth near the target and grows linearly away from it, which keeps the gradient from becoming too large when the realized proxy ratio is far from $\tau$.

**Ranking loss.** The scorer needs a training signal, and the natural signal is the attention the encoder already computes. FlashAttention does not expose its attention scores, and recomputing the full matrix to read them off would cost what the reduction saves. We therefore introduce an attention oracle. It recomputes a detached attention logit for each feature token on the post-PreMerge sequence, using the query and key projections of the consuming block, i.e. the EVA block immediately following the current reduction stage. The oracle ranks the feature tokens by how strongly that block would attend to them, and it is used only during training. The ranking loss trains the scorer to reproduce that ordering. At inference, the scorer alone determines which tokens survive.

Specifically, let $a_{b,j}$ denote the oracle logit for feature token $j$ in sample $b$. With $H_{\text{blk}}$ attention heads and head width $d = D/H_{\text{blk}}$, it is computed as

$$a_{b,j} = \frac{1}{H_{\text{blk}}} \sum_{m=1}^{H_{\text{blk}}} \frac{\langle \boldsymbol{q}_{b,m}, \boldsymbol{k}_{b,j,m} \rangle}{\sqrt{d}}, \tag{11}$$

where $\boldsymbol{q}_{b,m}$ and $\boldsymbol{k}_{b,j,m}$ are the query and key projections of the consuming block on the post-PreMerge sequence, with RoPE applied to both queries and keys. The oracle uses no softmax, as softmax does not affect ranking. Appendix E.3 gives the implementation details of the oracle and of the correlation. The ranking loss is then

$$\mathcal{L}_{\text{rank}} = \frac{1}{R} \sum_{i=1}^{R} \text{mean}_{b \in \mathcal{V}_i} \left[ 1 - \text{Corr}(\boldsymbol{s}_b, \boldsymbol{a}_b) \right], \tag{12}$$

where Corr is the Pearson correlation computed over feature tokens within each sample. Here $\boldsymbol{s}_b$ and $\boldsymbol{a}_b$ collect the scorer and oracle logits for sample $b$, and $\mathcal{V}_i \subseteq \mathcal{B}$ restricts the mean to those samples that yield a finite correlation at stage $R_i$. Correlation suits this objective because the oracle provides an attention-based importance target for that stage, and aligning the two rankings does not require the scores to match in scale. The joint objective is then

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_F \mathcal{L}_F + \lambda_r \mathcal{L}_{\text{rank}}, \tag{13}$$

where $\mathcal{L}_{\text{task}}$ is the task loss of the dataset, and $\lambda_F$, $\lambda_r$ are fixed hyperparameters (Appendix C.2).

# 4 Experiments

The tables separate EK-100 multi-instance retrieval (MIR) measurements (Damen et al., 2022; 2018) from single-view K400 classification (Carreira & Zisserman, 2017). The EK-100 columns report mAP and nDCG averaged over video-to-text and text-to-video retrieval, and the K400 columns report top-1 and top-5 accuracy. All metrics are expressed as percentages, and every comparison uses the realized proxy ratio. All settings for the reference configurations are collected in Table 10, while the evaluation and FLOPs-accounting conventions are in Appendices F and D.4.

## 4.1 Main Results

Table 1 compares the dense backbone with four training-based methods and three training-free methods. The training-based methods are TokenLearner (Ryoo et al., 2021), STTS (Wang et al., 2022), Unified STTS (Zhang et al., 2026) and TempMe (Shen et al., 2025); the training-free ones are ToMe (Bolya et al., 2023), STA (Ding et al., 2023) and the video-aware merge of Pollard & Wray (Pollard & Wray, 2025). A uniform random drop is a reference rather than a method, since it removes tokens with no selection signal at all. The Dense (varlen) row uses the packed variable-length EVA path, retains all feature tokens, disables reduction, and serves as the baseline for all reported deltas. The Dense (original) row follows the standard dense execution path. On EK-100, the two dense rows agree within 0.01 mAP and 0.03 nDCG, indicating that the varlen dense backbone achieves performance comparable to the original dense backbone. Among the token-reduction methods in Table 1, SAVOR achieves the highest EK-100 mAP and nDCG at both targets, exceeding the strongest competing method by 0.77 mAP at a target of 0.5 and 0.14 mAP at a target of 0.7. On K400, SAVOR leads in three of the four columns; the exception is Top-1 at $\tau$=0.5, where TempMe is 0.25 percentage points higher while realizing a proxy ratio of 0.549 against SAVOR's 0.519, so the two are not budget-matched. K400 separates the methods much less sharply. At $\tau = 0.7$, the reduced models lie within about a point of the dense backbone there, against 2.2 points on EK-100, and the uniform random drop gives up only 0.22 against 0.75. The high late-layer temporal similarity observed on K400 Appendix A.8is consistent with the smaller performance gaps among reduction methods, although it does not establish their cause. Table 1 reports single-view K400 results. Under the 5×3 protocol, SAVOR exceeds random drop in top-1 accuracy by 0.68 and 0.47 percentage points at targets of 0.5 and 0.7, respectively. Appendix A.5 reports both $2 \times 3$ and $5 \times 3$ view protocols.

Table 1: EK-100 MIR and K400 results at the global FLOPs targets 0.5 and 0.7. The $\Phi(X)$ column lists the realized proxy ratio on K400 and on EK-100 in that order. Each method uses its closest official setting, listed in Table 11.

| Method | Global FLOPs target $\tau$=**0.5** | | | | | Global FLOPs target $\tau$=**0.7** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K400/EK-100 $\Phi(X)$ | K400 (1×1) @1 | K400 (1×1) @5 | EK-100 mAP | EK-100 nDCG | K400/EK-100 $\Phi(X)$ | K400 (1×1) @1 | K400 (1×1) @5 | EK-100 mAP | EK-100 nDCG |
| Dense (varlen) | 1.000/1.000 | 79.26 | 93.33 | 60.71 | 71.65 | 1.000/1.000 | 79.26 | 93.33 | 60.71 | 71.65 |
| Dense (original) | 1.000/1.000 | — | — | 60.72 | 71.62 | 1.000/1.000 | — | — | 60.72 | 71.62 |
| *Training-free* | | | | | | | | | | |
| Random drop | 0.500/0.500 | 78.49(-0.77) | 92.84(-0.49) | 58.69(-2.02) | 70.05(-1.60) | 0.700/0.700 | 79.04(-0.22) | 93.12(-0.21) | 59.96(-0.75) | 71.04(-0.61) |
| ToMe | 0.506/0.506 | 78.77(-0.49) | 93.07(-0.26) | 59.25(-1.46) | 70.66(-0.99) | 0.696/0.696 | 79.13(-0.13) | 93.24(-0.09) | 60.29(-0.42) | 71.44(-0.21) |
| STA | 0.496/0.496 | 78.00(-1.26) | 92.83(-0.50) | 57.81(-2.90) | 69.06(-2.59) | 0.728/0.728 | 79.06(-0.20) | 93.25(-0.08) | 59.82(-0.89) | 70.90(-0.75) |
| Pollard & Wray | 0.515/0.515 | 78.98(-0.28) | 93.20(-0.13) | 58.23(-2.48) | 69.58(-2.07) | 0.718/0.718 | 79.32(+0.06) | 93.34(+0.01) | 60.60(-0.11) | 71.46(-0.19) |
| *Training-based* | | | | | | | | | | |
| TokenLearner | 0.526/0.526 | 75.04(-4.22) | 91.26(-2.07) | 56.50(-4.21) | 69.00(-2.65) | 0.763/0.763 | 78.46(-0.80) | 93.18(-0.15) | 58.50(-2.21) | 70.40(-1.25) |
| STTS | 0.529/0.529 | 78.97(-0.29) | 93.21(-0.12) | 58.95(-1.76) | 70.12(-1.53) | 0.729/0.729 | 79.39(+0.13) | 92.95(-0.38) | 59.31(-1.40) | 70.70(-0.95) |
| TempMe | 0.549/0.549 | **79.33(+0.07)** | 92.90(-0.43) | 58.80(-1.91) | 70.60(-1.05) | 0.733/0.739 | 79.31(+0.05) | 93.70(+0.37) | 59.32(-1.39) | 71.23(-0.42) |
| Unified STTS | — | — | — | — | — | 0.708/0.713 | 79.38(+0.12) | 93.35(+0.02) | 60.14(-0.57) | 71.60(-0.05) |
| Full SAVOR | 0.519/0.495 | 79.08(-0.18) | **93.40(+0.07)** | **60.02(-0.69)** | **71.54(-0.11)** | 0.717/0.697 | **79.42(+0.16)** | **93.73(+0.40)** | **60.74(+0.03)** | **72.02(+0.37)** |

Table 2: EK-100 MIR at targets $\tau = 0.3$–$0.7$ and on the dense backbone. $\Phi(X)$ is the analytical FLOPs proxy, Reserved GPU memory is the peak GPU memory reserved during training, and Throughput is the actual training throughput.

| Target $\tau$ | $\Phi(X)$ | mAP | nDCG | Reserved GPU memory (MiB) | Throughput (videos/s) |
|---|---|---|---|---|---|
| dense | 1.0 | 60.71 | 71.65 | 14,750 | 18.6 |
| 0.7 | 0.6968 | 60.74 (+0.03) | 72.02 (+0.37) | 14,234 | 19.7 |
| 0.6 | 0.5976 | 60.30 (-0.41) | 71.61 (-0.04) | 13,937 | 22.1 |
| 0.5 | 0.4946 | 60.02 (-0.69) | 71.54 (-0.11) | 13,793 | 23.2 |
| 0.4 | 0.3977 | 58.87 (-1.84) | 70.49 (-1.16) | 12,636 | 29.4 |
| 0.3 | 0.2986 | 56.14 (-4.57) | 68.35 (-3.30) | 12,749 | 33.0 |

### 4.2 Compute Control and Efficiency

Table 2 gives accuracy, reserved GPU memory, and actual training throughput for the dense backbone and for different compression levels. Memory and throughput are measured on a single RTX 4090 at a batch size of 32. In fact, a single trained model can serve the nearby targets by re-solving the budget while the weights stay fixed (Appendix A.2). However, to conduct a fair comparison, we still trained all the models at $\tau \in [0.3, 0.7]$ to verify the reserved GPU memory and actual training throughput during the training stage.

We report peak GPU memory reserved by the caching allocator, which includes both live tensor allocations and cached memory. SAVOR's reserved memory stays below that of the dense backbone at every global FLOPs target. The reserved GPU memory for $\tau = 0.7$ is 14,234 MiB, slightly below the 14,750 MiB for the dense model, and it falls to 13,937 MiB at 0.6 and 13,793 MiB at 0.5. The saving is small because the ATR scorer and the HFA buffers fragment the caching allocator, as Appendix G details. Meanwhile, the actual throughput is close to the dense backbone at 0.7, 19.7 against 18.6, and peaks at 33.0 videos/s at 0.3.

Table 3: EK-100 MIR component ablations at the global FLOPs target $\tau$=0.5. Parentheses show percentage-point changes from Full SAVOR.

| Configuration | mAP | nDCG | $\Phi(X)$ |
|---|---|---|---|
| Dense (varlen) | 60.71 | 71.65 | 1.0000 |
| **Full SAVOR** | **60.02** | **71.54** | **0.4946** |
| w/o PreMerge | 59.78 (−0.24) | 71.21 (−0.33) | 0.4998 |
| Fixed-anchor control ($\rho$=$\rho_a$) | 59.68 (−0.34) | 71.30 (−0.24) | 0.5147 |
| w/o GRU feedback | 59.90 (−0.12) | 71.44 (−0.10) | 0.5015 |
| w/o attention oracle | 58.14 (−1.88) | 69.61 (−1.93) | 0.5330 |
| ATR drop-only ($f$=1.0) | 59.80 (−0.22) | 71.23 (−0.31) | 0.4935 |

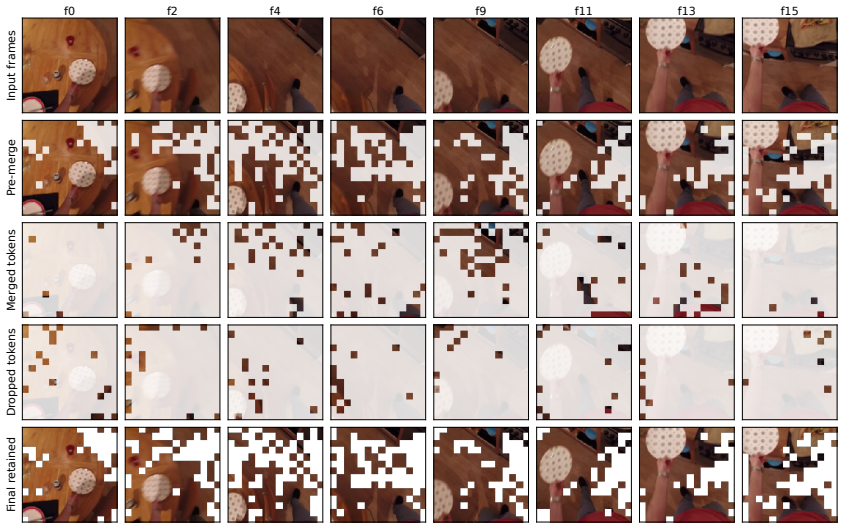


Figure 3: EK-100 clip P01_11 at $\tau$=0.7. Rows 1–5 show input frames, post-PreMerge tokens, merged tokens, dropped tokens (blue), and remaining tokens.

### 4.3 Ablation Study

Removing PreMerge costs 0.24 mAP and 0.33 nDCG, and disabling the merge branch costs 0.22 mAP. PreMerge handles visible redundancy, so the ATR scorer can focus on importance ranking, and the merge branch keeps the evidence of tokens that score too low to be kept directly. The oracle accounts for the largest change in Table 3, a drop of 1.88 mAP and 1.93 nDCG. Without the ranking loss, the ATR scorer selects no better than a uniform random drop (58.14 against 58.69).

The fixed-anchor control reaches 59.68 mAP and 71.30 nDCG. Pinning the retention ratio at the solver's anchor, $\rho_a = 0.9485$ (Appendix D.5), costs $0.34$ mAP and $0.24$ nDCG at $\tau = 0.5$, and the proxy ratio of the control moves outside the $2\%$ band that full SAVOR holds at this target. Removing only the GRU feedback costs 0.12 mAP. Appendix A.6 shows the details of the fixed-anchor control.

### 4.4 Visualization Results

Figure 3 follows one EK-100 clip through the reduction. During the static opening (frames 1–6), near-redundant feature tokens disappear and retention falls. When the hand–object interaction begins (frames 7–10), the surviving tokens concentrate on hands and objects rather than on the background. Appendix A.9 shows four further clips under the same setting, and Appendix A.7 reports the accompanying similarity and token-count statistics.

## 5 Conclusion

Variable-length execution allows sample-aware token reduction to lower the computation performed by subsequent encoder blocks. The packed variable-length backbone enables this: each stage rewrites per-sample lengths and repacks the sequence, so higher-level modules choose how many tokens each sample keeps rather than masking tokens in a fixed-length tensor.

Built on this backbone, SAVOR separates three decisions that token reduction usually conflates. PreMerge identifies redundant tokens based on feature similarity; the ATR scorer identifies important tokens with a CLS-conditioned head trained using the encoder's own attention; and HFA determines how many tokens should survive by solving for a per-layer budget under a single global FLOPs target. The global FLOPs target is specified externally, while the retention ratios adapt across samples and layers around a calibrated anchor. On EK-100, SAVOR reduces the analytical encoder FLOPs proxy by 30.3%, maintains comparable mAP to the dense backbone, and improves nDCG by 0.37 percentage points. Appendix G discusses its limitations.

## AI USE STATEMENT

We used generative AI tools to assist with implementing parts of the Hierarchical FLOPs Allocator (HFA), rewriting the token-reduction path from per-document Python loops into batched tensor operations, adapting official baseline implementations to the EVA backbone, and implementing and checking analytical FLOPs calculations, including those for the fixed-anchor control. We also used these tools to revise and polish the manuscript. The authors are responsible for reviewing all AI-assisted code, calculations, and text and take responsibility for the final content of this work, including AI-assisted text, claims, and artifacts.

## REFERENCES

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. *arXiv:2506.09985*, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021.

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

Shyamal Buch, Arsha Nagrani, Anurag Arnab, and Cordelia Schmid. Flexible frame selection for efficient video reasoning. In *CVPR*, pp. 29071–29082, 2025.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pp. 6299–6308, 2017.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *ECCV*, pp. 19–35, 2024.

Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable compression rate for efficient vision transformers. In *ICCV*, pp. 17118–17128, 2023.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pp. 1724–1734, 2014.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *IJCV*, 130:33–55, 2022.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *NeurIPS*, 35:16344–16359, 2022.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *NeurIPS*, 36:2252–2274, 2023.

Shuangrui Ding, Peisen Zhao, Xiaopeng Zhang, Rui Qian, Hongkai Xiong, and Qi Tian. Prune spatio-temporal tokens by semantic-aware temporal accumulation. In *ICCV*, pp. 16899–16910, 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 *ICLR*, 2021.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *ECCV*, pp. 396–414, 2022.

Tianyu Fu, Tengxuan Liu, Qinghao Han, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Framefusion: Combining similarity and importance for video token reduction on large vision language models. In *ICCV*, pp. 22654–22663, 2025.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, pp. 18995–19012, 2022.

Peter J Huber. Robust estimation of a location parameter. *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.

Jeongseok Hyun, Sukjun Hwang, Su Ho Han, Taeoh Kim, Inwoong Lee, Dongyoon Wee, Joon-Young Lee, Seon Joo Kim, and Minho Shim. Multi-granular spatio-temporal token merging for training-free acceleration of video llms. In *ICCV*, pp. 23990–24000, 2025.

Yang Jin, Zhicheng Sun, Kun Xu, Kun Xu, Liwei Chen, Hao Jiang, Quzhe Huang, Chengru Song, Yuliang Liu, Di Zhang, Yang Song, Kun Gai, and Yadong Mu. Video-LaVIT: Unified video-language pre-training with decoupled visual-motional tokenization. In *ICML*, volume 235, pp. 22185–22209, July 2024.

Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *ECCV*, pp. 639–655, 2018.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EViT: Expediting vision transformers via token reorganizations. In *ICLR*, 2022.

Kevin Qinghong Lin, Jinpeng Wang, Mattia Soldan, Michael Wray, Rui Yan, Eric Z Xu, Difei Gao, Rong-Cheng Tu, Wenzhe Zhao, Weijie Kong, et al. Egocentric video-language pretraining. *NeurIPS*, 35:7575–7586, 2022.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, volume 32, 2018.

Sam Pollard and Michael Wray. Video, how do your tokens merge? In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3338–3347. IEEE, 2025.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. *NeurIPS*, 34:13937–13949, 2021.

Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *NeurIPS*, 34:12786–12797, 2021.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *NeurIPS*, 37: 68658–68685, 2024.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *ICCV*, pp. 22857–22867, 2025.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Leqi Shen, Tianxiang Hao, Tao He, Sicheng Zhao, Yifeng Zhang, Yongjun Bao, Guiguang Ding, et al. Tempme: Video temporal token merging for efficient text-video retrieval. In *International Conference on Learning Representations*, volume 2025, pp. 60839–60860, 2025.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Xiaoqi Wang, Yi Wang, and Lap-Pui Chau. EVA02-AT: Egocentric video-language understanding with spatial-temporal rotary positional embeddings and symmetric optimization. *IEEE Transactions on Image Processing*, 2026.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *CVPR*, pp. 10799–10808, 2022.

Chuhan Zhang, Ankush Gupta, and Andrew Zisserman. Helping hands: An object-aware ego-centric video recognition model. In *ICCV*, pp. 13855–13866, 2023.

Jianrui Zhang, Yue Yang, Rohun Tripathi, Winson Han, Ranjay Krishna, Christopher Clark, Yong Jae Lee, and Sangho Lee. Unified spatio-temporal token scoring for efficient video vlms. *arXiv preprint arXiv:2603.18004*, 2026.

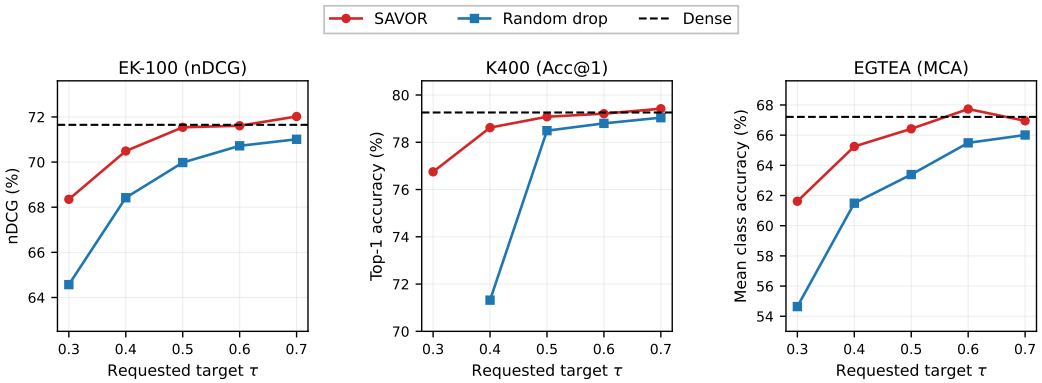


Figure 4: Global FLOPs target sweep. Panels report EK-100 nDCG, K400 top-1 accuracy under the single-view protocol, and EGTEA mean class accuracy on split 1. The dashed line marks the dense backbone.

# A Additional Results and Analysis

## A.1 Target Sweep across Datasets

Figure 4 sweeps the global FLOPs target from 0.3 to 0.7 on three datasets. SAVOR and uniform random drop are evaluated at every target, and the dense backbone is drawn as a horizontal dashed line. On K400, uniform random drop is shown from $\tau$=0.4 because at 0.3 its token budget is exhausted in the late layers.

## A.2 Budget Transfer

A trained SAVOR model can be moved to a different global FLOPs target without updating any weight, because $\tau$ enters the pipeline only through the shared anchor $\rho_a$. Budget transfer changes $\tau$ and re-solves the anchor from the hard forwards of the checkpoint, after which ATR applies the re-solved budget at every stage. During evaluation, the anchor is re-solved online from the hard token counts of the preceding forward passes, so the first 25 forward passes, about 2% of the 9,668-clip test split at the evaluation batch size, still use the anchor carried by the checkpoint. All transfer evaluations use the reference PreMerge setting.

Table 4: EK-100 MIR under budget transfer. Each configuration transfers the checkpoint trained at the source target to the evaluation target in the first column, with all weights frozen. "Trained" columns give the checkpoint trained at the evaluation target (Table 2), and parentheses show percentage-point changes from the dense backbone.

| Transfer | $\Phi(X)$ | mAP | nDCG | Trained mAP | Trained nDCG |
|---|---|---|---|---|---|
| dense | 1.0 | 60.71 | 71.65 | — | — |
| 0.5→0.3 | 0.3057 | 55.52 (-5.19) | 67.59 (-4.06) | 56.14 | 68.35 |
| 0.5→0.4 | 0.4016 | 59.00 (-1.71) | 70.67 (-0.98) | 58.87 | 70.49 |
| 0.5→0.6 | 0.5954 | 60.47 (-0.24) | 71.81 (+0.16) | 60.30 | 71.61 |
| 0.5→0.7 | 0.6934 | 60.61 (-0.10) | 71.89 (+0.24) | 60.74 | 72.02 |
| 0.7→0.5 | 0.5090 | 59.73 (-0.98) | 71.21 (-0.44) | 60.02 | 71.54 |
| 0.7→0.6 | 0.6009 | 60.51 (-0.20) | 71.91 (+0.26) | 60.30 | 71.61 |

Three observations follow.First, transfer to a higher target yields mAP values close to those of the natively trained checkpoints in these single-run evaluations. The largest gap to the natively trained checkpoint is 0.17 mAP, close to the 0.10 mAP variation between repeated trainings of one configuration, and the realized proxy ratio tracks the new target, 0.5954 against 0.6 and 0.6934 against 0.7. Second, a single step downward also costs nothing measurable, and both 0.5→0.4 and 0.7→0.6

land on par with or slightly above the natively trained checkpoint. Third, two steps downward pay a real cost, 0.62 mAP for 0.5→0.3 and 0.28 mAP for 0.7→0.5.

The cost grows with the distance between source and evaluation targets, while the native curve is nearly flat above 0.5. One training at 0.5 therefore covers the 0.4 to 0.7 range, and a wider range favors a training target near the middle. Each configuration is evaluated once.

### A.3 Backbone Generality

Table 5 repeats the comparison of Table 2 with a ViT-B/16 backbone in place of EVA02-AT (Wang et al., 2026), on both datasets. Dense, SAVOR at $\tau$=0.5 and SAVOR at $\tau$=0.7 are reported for each backbone.

Table 5: EK-100 MIR and K400 classification with two backbones. The $\Phi(X)$ column lists the realized proxy ratio on K400 and on EK-100 in that order.

| | | K400 (1×1) | | EK-100 | |
|---|---|---|---|---|---|
| Setting | K400/EK-100 | @1 | @5 | mAP | nDCG |
| *ViT-B/16* | | | | | |
| Dense | 1.000/1.000 | 77.86 | 92.79 | 58.30 | 70.30 |
| SAVOR ($\tau$=0.5) | 0.503/0.501 | 77.69(-0.17) | 92.76(-0.03) | 58.10(-0.20) | 70.31(+0.01) |
| SAVOR ($\tau$=0.7) | 0.708/0.701 | 78.17(+0.31) | 92.73(-0.06) | 58.44(+0.14) | 70.63(+0.33) |
| *EVA02-AT* | | | | | |
| Dense | 1.000/1.000 | 79.26 | 93.33 | 60.71 | 71.65 |
| SAVOR ($\tau$=0.5) | 0.519/0.495 | 79.08(-0.18) | 93.40(+0.07) | 60.02(-0.69) | 71.54(-0.11) |
| SAVOR ($\tau$=0.7) | 0.717/0.697 | 79.42(+0.16) | 93.73(+0.40) | 60.74(+0.03) | 72.02(+0.37) |

### A.4 An Additional Dataset

We additionally evaluate SAVOR on EGTEA (Li et al., 2018), an egocentric action recognition dataset outside the two benchmarks of the main text. EGTEA uses split 1 of the three official splits, with 106 classes and about 8,299 training and 2,022 test clips; its primary metric is mean class accuracy, with top-1 and top-5 accuracy reported alongside. Dense, uniform random drop and SAVOR are reported at $\tau$=0.5 and $\tau$=0.7.

Table 6: EGTEA action recognition on split 1. The $\Phi(X)$ column lists the realized proxy ratio.

| Setting | $\Phi(X)$ | MCA | Acc@1 | Acc@5 |
|---|---|---|---|---|
| Dense | 1.000 | 67.21 | 74.04 | 94.41 |
| Random drop ($\tau$=0.5) | 0.500 | 63.39(-3.82) | 71.27(-2.77) | 92.28(-2.13) |
| Random drop ($\tau$=0.7) | 0.700 | 66.01(-1.20) | 72.50(-1.54) | 93.67(-0.74) |
| SAVOR ($\tau$=0.5) | 0.509 | 66.42(-0.79) | 73.05(-0.99) | 94.31(-0.10) |
| SAVOR ($\tau$=0.7) | 0.706 | 66.95(-0.26) | 73.00(-1.04) | 94.36(-0.05) |

### A.5 Multi-View K400 Results

Table 7 repeats the comparison of Table 1 on Kinetics-400 under the $2 \times 3$ and $5 \times 3$ view protocols, at the same two targets.

### A.6 The Fixed-Anchor Control

The fixed-anchor control in Table 3 replaces the per-stage budget with a single fixed ratio $\rho$=$\rho_a$=0.9485, calibrated as described in Appendix D.5. Under HFA, each reduction stage receives its own ratio, produced by the allocator from the stage progress and the FiLM-modulated state, so the ratio moves around the anchor as the reduction progresses, and it also differs from sample to sample. The control pins that ratio at the anchor for every stage and every sample, so neither the depth nor the sample identity moves it any more. Per-sample token counts nonetheless keep varying. ATR scores

Table 7: Kinetics-400 under the $2 \times 3$ and $5 \times 3$ view protocols at the two targets. Dashes mark unavailable results.

| Method | Global FLOPs target $\tau$=**0.5** | | | | | Global FLOPs target $\tau$=**0.7** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\Phi(X)$ | K400 (2×3) | | K400 (5×3) | | $\Phi(X)$ | K400 (2×3) | | K400 (5×3) | |
| | | @1 | @5 | @1 | @5 | | @1 | @5 | @1 | @5 |
| Dense (varlen) | 1.000 | 81.44 | 94.43 | 83.25 | 95.53 | 1.000 | 81.44 | 94.43 | 83.25 | 95.53 |
| *Training-free* | | | | | | | | | | |
| Random drop | 0.500 | 80.85(-0.59) | 94.22(-0.21) | 82.66(-0.59) | 95.40(-0.13) | 0.700 | 81.15(-0.29) | 94.40(-0.03) | 82.96(-0.29) | 95.48(-0.05) |
| ToMe | 0.506 | 81.13(-0.31) | 94.27(-0.16) | 82.98(-0.27) | 95.47(-0.06) | 0.696 | 81.28(-0.16) | 94.34(-0.09) | 83.16(-0.09) | 95.52(-0.01) |
| STA | 0.496 | 80.48(-0.96) | 94.18(-0.25) | 82.47(-0.78) | 95.32(-0.21) | 0.728 | 81.31(-0.13) | 94.37(-0.06) | 83.09(-0.16) | 95.54(+0.01) |
| Pollard & Wray | 0.515 | 81.37(-0.07) | 94.40(-0.03) | 83.17(-0.08) | 95.58(+0.05) | 0.718 | 81.76(+0.32) | 94.53(+0.10) | 83.37(+0.12) | 95.71(+0.18) |
| *Training-based* | | | | | | | | | | |
| TokenLearner | 0.526 | 77.88(-3.56) | 92.83(-1.60) | 79.99(-3.26) | 94.24(-1.29) | 0.763 | 81.14(-0.30) | 94.34(-0.09) | 83.06(-0.19) | 95.59(+0.06) |
| STTS | 0.529 | 81.64(+0.20) | 94.53(+0.10) | **83.35(+0.10)** | 95.73(+0.20) | 0.729 | 81.60(+0.16) | 94.29(-0.14) | 83.29(+0.04) | 95.51(-0.02) |
| TempMe | 0.550 | **81.76(+0.32)** | 94.36(-0.07) | 83.23(-0.02) | 95.51(-0.02) | 0.733 | 81.61(+0.17) | **95.03(+0.60)** | 83.30(+0.05) | **96.06(+0.53)** |
| Unified STTS | — | — | — | — | — | 0.708 | 81.63(+0.19) | 94.47(+0.04) | 83.29(+0.04) | 95.61(+0.08) |
| Full SAVOR | 0.519 | 81.54(+0.10) | **94.81(+0.38)** | 83.34(+0.09) | **95.83(+0.30)** | 0.717 | **81.83(+0.39)** | 94.92(+0.49) | **83.43(+0.18)** | 96.04(+0.51) |

and merges on the packed sequence, and nothing in the reduction assigns a fixed length to a sample, so PreMerge and the admissible pairs still give each sample its own token count. The fixed-anchor control therefore keeps the variable-length backbone and removes only the budget's contribution to sample-adaptive behavior. SAVOR is sample-adaptive along two axes, the variable-length execution and the per-sample, per-layer budget, and the fixed-anchor control isolates the budget.

The value $\rho_a$=0.9485 was obtained by solving $\Phi(\rho_a) = \tau$ on a SAVOR checkpoint trained at $\tau$=0.5 (Appendix D.5), so the fixed schedule is the calibrated answer for this target. The comparison varies the learned deviation around a calibrated average: the control applies a single ratio to every stage and has no feedback path, so a stage that over-compresses is never offset by a later stage that compresses less, and the realized proxy ratio drifts above the target (Table 3).

### A.7 Layer-Wise Temporal Redundancy

Figure 5 compares the layer-wise distribution of cosine similarity for dense EVA and SAVOR. At the input to each EVA block, we collect every temporally adjacent token pair whose members share the same spatial position. Each column is normalized independently by the number of aligned pairs at that layer. The plotted distributions are therefore not cumulative across depth. The dense backbone processes 28.42 million such pairs at every layer, whereas SAVOR reduces this count to 6.20 million by the final block through physical token merging and removal. Meanwhile, the fraction above the PreMerge threshold $\delta$ falls from 38.7% at the first block to 0.5% at the final block under SAVOR, compared with 14.8% for dense EVA. The gap between the two panels widens with depth: the two models agree at the first block, where no reduction has been applied, and SAVOR's average similarity sits 0.010 below the dense backbone's at the fourth block and 0.138 below it at the last. This analysis uses the $\tau$=0.5 checkpoint under the reference PreMerge setting (realized proxy ratio 0.4946).

Table 8 gives the same reduction in counts rather than in similarity. It is close to even across depth: after the first stage, which removes 13%, every stage removes roughly a tenth of its input. That is the property the shared anchor is there to produce, since without it the reduction is not spread across depth (Appendix D.5). The min and max columns also show how far the samples spread from one another, 6% of the mean at the first reduced block and 22% at the last.

### A.8 Cross-Dataset Redundancy

Appendix A.7 measures the layer-wise temporal similarity of EK-100. Figure 6 repeats that measurement on K400 under the same protocol, that is, a single view, the same PreMerge threshold $\delta$, and the same binning.

The dense backbone behaves differently on the two datasets. On EK-100, the average similarity falls with depth, from 0.83 at the first block to 0.678 at the last, so the later blocks still carry frame-to-frame change. On K400, the average similarity rises with depth instead, from a dip in the middle blocks to 0.98 at the last block, so on K400 the later blocks are almost entirely redundant. The cost

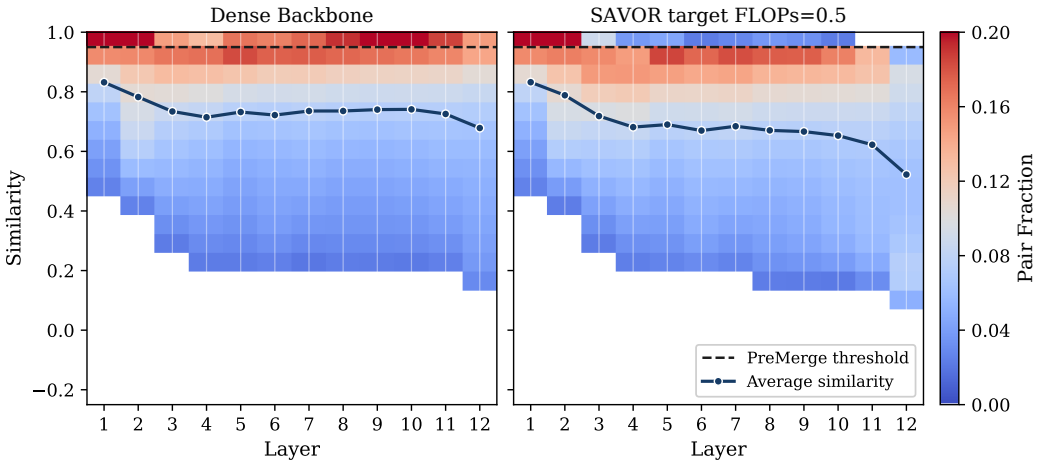


Figure 5: Layer-wise temporal redundancy on the full EK-100 test split. The two panels compare dense EVA with the $\tau$=0.5 SAVOR checkpoint. Color denotes the fraction of aligned temporal pairs in each cosine-similarity bin, the dashed line marks the PreMerge threshold $\delta$, and the marked curve is the per-layer average.

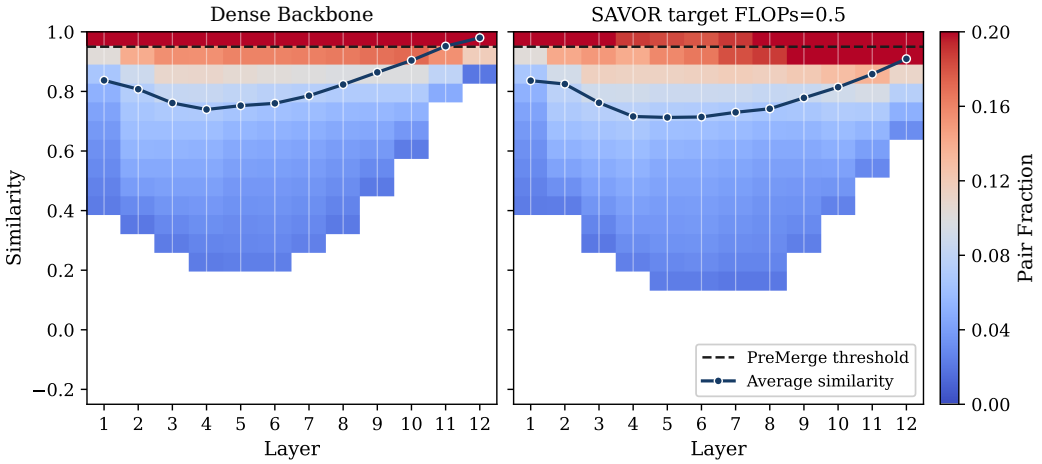


Figure 6: Layer-wise temporal redundancy on the full K400 test split, measured under the protocol of Figure 5. The two panels compare the dense backbone with the $\tau$=0.5 SAVOR checkpoint. Color denotes the fraction of aligned temporal pairs in each cosine-similarity bin, the dashed line marks the PreMerge threshold $\delta$, and the marked curve is the per-layer average.

Table 8: Token counts entering every EVA block on the full EK-100 test split, at $\tau$=0.5. Layer $\ell$ is the input to block $\ell$, so layer 0 is the unreduced clip and the last column is the change from the previous layer. Min and max are taken over the clips of the test split.

| Layer | Mean | Min | Max | vs. dense | vs. previous layer |
|---|---|---|---|---|---|
| 0 | 3,137 | 3,137 | 3,137 | 1.000 | – |
| 1 | 2,721 | 2,626 | 2,802 | 0.867 | −13.3% |
| 2 | 2,461 | 2,246 | 2,566 | 0.785 | −9.6% |
| 3 | 2,238 | 2,040 | 2,337 | 0.713 | −9.1% |
| 4 | 2,026 | 1,809 | 2,137 | 0.646 | −9.5% |
| 5 | 1,827 | 1,619 | 1,938 | 0.582 | −9.8% |
| 6 | 1,643 | 1,449 | 1,738 | 0.524 | −10.1% |
| 7 | 1,461 | 1,284 | 1,569 | 0.466 | −11.1% |
| 8 | 1,296 | 1,131 | 1,395 | 0.413 | −11.3% |
| 9 | 1,159 | 998 | 1,263 | 0.370 | −10.6% |
| 10 | 1,039 | 901 | 1,136 | 0.331 | −10.4% |
| 11 | 931 | 810 | 1,016 | 0.297 | −10.4% |

of the reduction follows the same contrast: at $\tau$=0.5, SAVOR gives up 1.1% of the EK-100 dense mAP and 0.2% of the K400 dense accuracy.

The reduction removes less of that similarity on K400 than on EK-100. SAVOR's final-block average similarity is 0.138 below the dense backbone's on EK-100 and 0.071 below on K400, so on K400 most of what SAVOR removes is redundancy the dense backbone was not using.

The accuracy cost and the gaps between reduction methods follow the same pattern on the two datasets (Section 4.1).

### A.9 Further Reduction Examples

Figure 3 follows a single EK-100 clip through the reduction pipeline. Figures 7, 8, 9 and 10 repeat that view for four further clips from the same test split, at the same target and with the same five rows and the same color convention.

## B Notation

EVA blocks use zero-based indices $\ell = 0, \ldots, L-1$. Reduction stages are indexed from one, so that $R_i$ is applied immediately before block $\ell_{c+i}$ and the stage number and the block number differ by exactly $c$.

Whole-sequence lengths $N$ and $n$ include the single retained CLS token, whereas every other count and ratio, such as $K_b$, $\rho_b^{R_i}$, $r_{b,\mathrm{pm}}^{i}$, and $r_{b,\mathrm{atr}}^{i}$, excludes it. Stage superscripts are dropped where the active stage is clear, as in $s_{b,j}$ for $s_{b,j}^{R_i}$. Table 9 lists the core symbols.

## C Configuration

### C.1 Backbone

The reference EK-100 configuration uses the EVA-B/16 encoder of EVA02-AT (Wang et al., 2026), pretrained on EgoClip+, an Ego4D-derived egocentric dataset (Grauman et al., 2022; Lin et al., 2022). Its input is a clip of $T$=16 frames, each resized and center-cropped to $224 \times 224$. The patch embedding turns that clip into $N_0 = 3137$ tokens. The SwiGLU MLP Shazeer (2020) has hidden width $D_{\mathrm{mlp}} = 2048$, which is the width that the FLOPs proxy of Appendix D.4 uses.

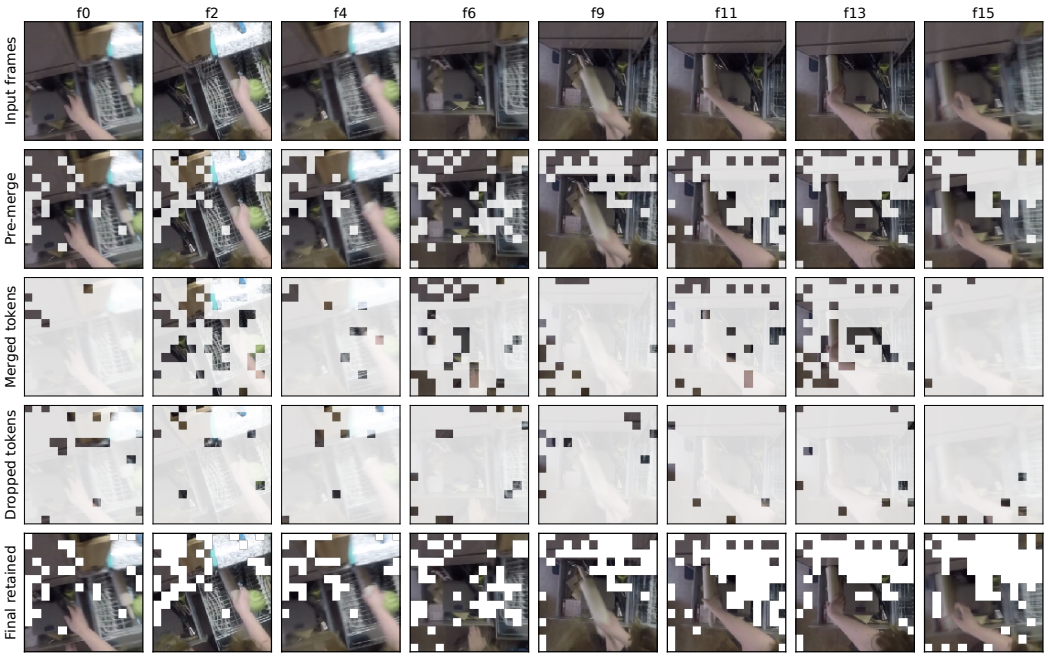


Figure 7: EK-100 clip P04_031 at $\tau$=0.7. Rows 1–5 show input frames, post-PreMerge tokens, merged tokens, dropped tokens (blue), and remaining tokens.

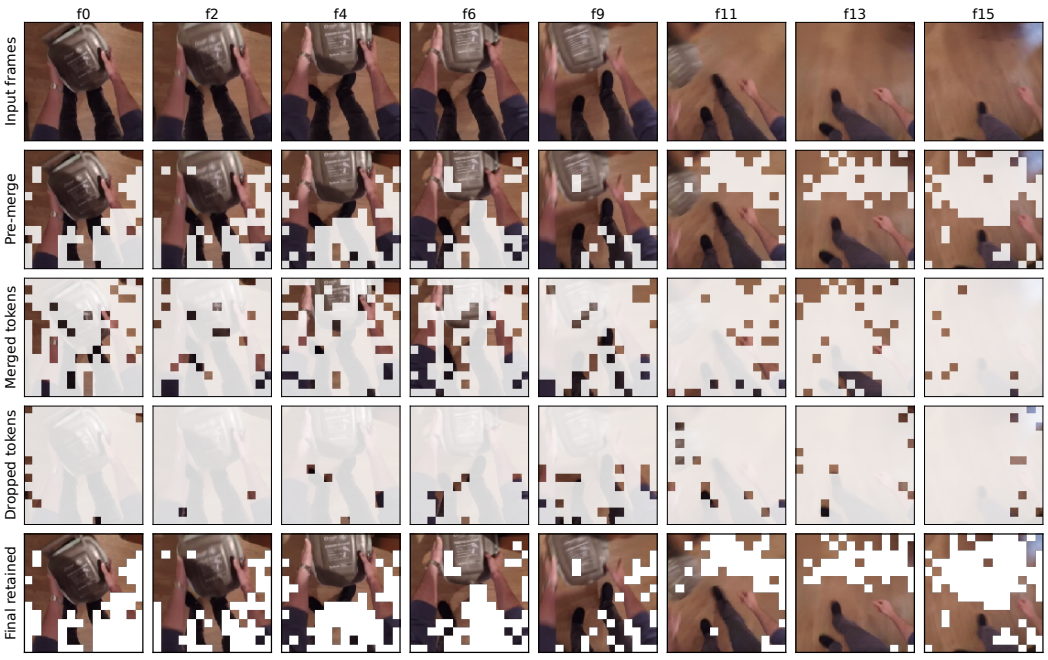


Figure 8: EK-100 clip P01_014 at $\tau$=0.7. Rows 1–5 show input frames, post-PreMerge tokens, merged tokens, dropped tokens (blue), and remaining tokens.

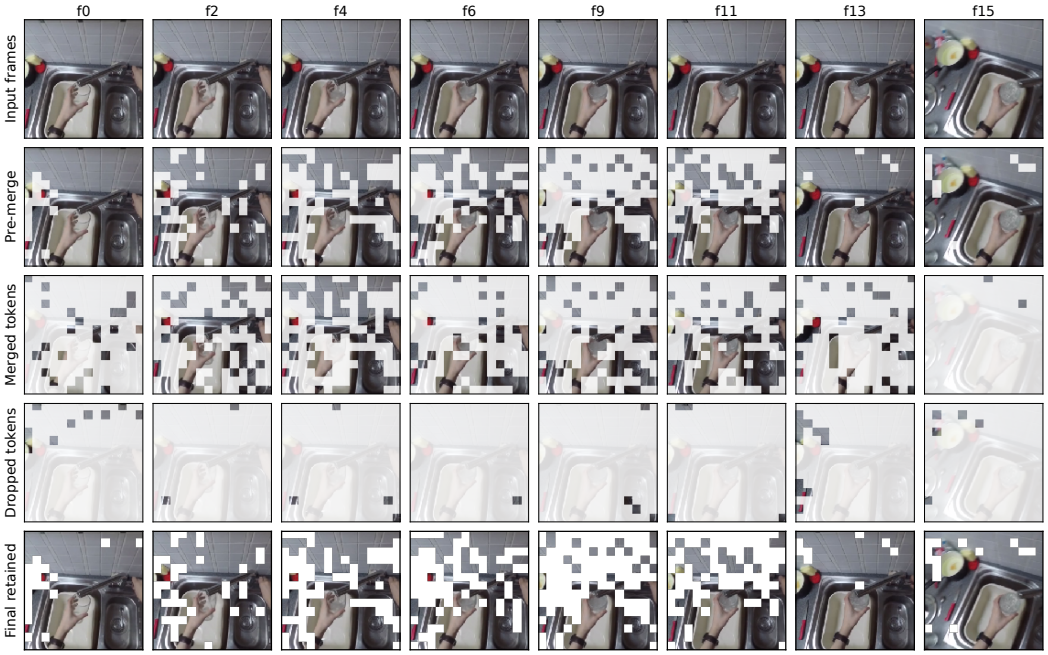


Figure 9: EK-100 clip P03_025 at $\tau$=0.7. Rows 1–5 show input frames, post-PreMerge tokens, merged tokens, dropped tokens (blue), and remaining tokens.




Figure 10: EK-100 clip P14_06 at $\tau$=0.7. Rows 1–5 show input frames, post-PreMerge tokens, merged tokens, dropped tokens (blue), and remaining tokens.

| Symbol | Meaning |
|---|---|
| *Lengths and model geometry* | |
| $N_0$ | Sequence length of one sample before reduction |
| $N_b^{\ell}$ | Sequence length of sample $b$ entering EVA block $\ell$ |
| $N_b^{R_i}$, $N_b^{R_i,\mathrm{pm}}$, $N_b^{R_i,\mathrm{atr}}$ | Sequence length of sample $b$ at stage $R_i$, before the stage and after PreMerge and ATR |
| $D, L$ | Token width and encoder depth |
| $D_{\mathrm{mlp}}$ | MLP hidden width |
| $c, R$ | Index of the last block without reduction and number of reduction stages, $R = L - c - 1$ |
| $H_{\mathrm{blk}}, d$ | EVA attention head count and head width, $d = D/H_{\mathrm{blk}}$ |
| $H_s, d_s, D_s$ | Scorer head count, head width, and projection width, $D_s = H_s d_s$ |
| *Feature tokens, scoring, and reduction controls* | |
| $(t, h, w), \mu$ | Temporal and spatial position of a feature token, and its merge mass |
| $\boldsymbol{c}_b, \boldsymbol{x}_j$ | CLS and feature-token vectors |
| $\boldsymbol{q}_{b,m}, \boldsymbol{k}_{b,j,m}$ | Scorer query from the CLS and key from feature token $j$, at head width $d_s$ |
| $s_{b,j}, b_s$ | Scorer logit for feature token $j$, i.e. the CLS-to-token attention logit, and its learnable bias |
| $a_{b,j}$ | Attention logit of the consuming block, used as the supervision target of $s_{b,j}$ |
| $\delta, \pi_{\mathrm{floor}}$ | PreMerge cosine-similarity threshold, and the retention floor that caps its merges |
| $K_b$ | Number of feature tokens ATR should retain at stage $R_i$ |
| $f$ | Direct-keep fraction, which sets the preferred direct-keep count $D_b$ and caps the merges at $K_b - D_b$ |
| *Allocation and feedback* | |
| $\rho_a$ | Shared anchor: the per-stage fraction of post-PreMerge feature tokens that reproduces $\tau$ on average |
| $\rho_b^{R_i}$ | Retention requested for sample $b$ at stage $R_i$, a bounded deviation from $\rho_a$ |
| $\rho_{\min}, \rho_{\max}, \kappa$ | Retention bounds, and the maximum relative deviation from the anchor |
| $\mathbf{h}_b^i$ | HFA state encoding sample $b$'s compression history |
| $H_{\mathrm{ctl}}$ | Hidden size of the HFA state |
| $r_{b,\mathrm{pm}}^i, r_{b,\mathrm{atr}}^i$ | Fraction of feature tokens retained at stage $R_i$ by PreMerge ($r_{b,\mathrm{pm}}^i$) and by ATR ($r_{b,\mathrm{atr}}^i$) |
| *Objective and accounting* | |
| $\Phi(X), \Phi(\rho), \widehat{\Phi}_F$ | Realized proxy ratio, the solver proxy at anchor $\rho$, and the straight-through estimator used for the FLOPs penalty |
| $\tau$ | Global FLOPs target, as a fraction of the dense cost |
| $\mathcal{F}_{\mathrm{blk}}, \mathcal{F}_{\mathcal{R}}$ | EVA-block and scorer costs |
| $\mathcal{L}_{\mathrm{task}}, \mathcal{L}_F, \mathcal{L}_{\mathrm{rank}}$ | Task loss, FLOPs penalty, and Pearson ranking loss |
| $\lambda_F, \lambda_r, \delta_H$ | Auxiliary loss weights and Huber transition point |
| $\sigma, \mathrm{sg}, \mathrm{clip}$ | Logistic sigmoid, stop-gradient, and interval clipping |

Table 9: Notation used throughout the paper. Symbols specific to the anchor solver, the straight-through estimators, and the pair matcher are defined in Appendices D.5, E.1, and D.2, respectively.

| Setting | Reference value |
|---|---|
| *Reduction layout* | |
| Stages | $c = 0$, $R = 11$ (before blocks $\ell_1, \ldots, \ell_{11}$) |
| Stage progress $u^i$ | $(i-1)/(R-1)$ |
| Global FLOPs target $\tau$ | 0.3–0.7 |
| *PreMerge* | |
| Cosine threshold $\delta$ | 0.95 (1.0 disables PreMerge) |
| Retention floor $\pi_{\text{floor}}$ | 0.90 |
| Temporal window | same $(h, w)$, $\lvert\Delta t\rvert < 2$ in units of sampled frames |
| Floor search | bisection over $[0.05, 1]$ with a $10^{-4}$ rounding margin |
| *Scorer and ATR* | |
| Scorer geometry | $H_s = 4$, $d_s = 48$, $D_s = 192$ for EVA-B/16 |
| Scorer bias $b_s$ | initialized to 2.0 |
| CLS logit override / initial merge mass | 10.0 / $\mu = 1.0$ |
| Direct-keep fraction $f$ | 0.8 |
| *Allocator (HFA)* | |
| HFA state | hidden size 64 |
| Ratio bounds $\rho_{\min}, \rho_{\max}$ | $[0.6, 1.0]$ |
| Deviation band $\kappa$ | 0.1 |
| Anchor updates | EMA momentum 0.9, frozen for 1 epoch, rate limit 0.01, dead zone 0.002 |
| Anchor calibration | 64 bisection iterations over $[\rho_{\min}, \rho_{\max}]$ |
| GRU initialization | all-zero weights, update-gate bias $+6$ (0.25% state decay per stage) |
| *Optimization* | |
| Auxiliary losses | $\lambda_F = 5$, $\lambda_r = 0.01$, Huber $\delta_H = 0.5$ |
| STE window / gate temperature | $w_{\text{STE}} = 32$; $\tau_g$: $1.0 \rightarrow \times 0.9$/epoch, floor 0.05 |
| Pearson norm floor | $10^{-6}$ |
| *Training recipe (EK-100 MIR)* | |
| Schedule | 100 epochs, AdaMuon (Si et al., 2025), batch 32 per device on 4 GPUs |
| Learning rate | $2 \times 10^{-5}$, warmup from $10^{-6}$ for 1 epoch, cosine to $2 \times 10^{-6}$ |
| Regularization | weight decay 0.01, FP16 autocast, gradient-checkpointing |
| Allocator learning rate | $10\times$ the backbone rate |
| Clip sampling / retrieval loss | 16 frames, stride 4; margin 0.6, threshold 0.1 |
| *Training recipe (K400)* | |
| Schedule | 30 epochs, AdaMuon, batch 32 per device on 4 GPUs |
| Learning rate | $10^{-3}$ |
| EMA evaluation | decay 0.999, single view |
| Regularization | weight decay 0.05, layer decay 0.75, drop path 0.1 |
| Augmentation | mixup 0.8, cutmix 1.0, label smoothing 0.1 |
| *Training recipe (EGTEA)* | |
| Schedule | 60 epochs, AdaMuon, batch 32 per device on 4 GPUs |
| Learning rate | $5 \times 10^{-4}$, warmup for 5 epochs, cosine to $10^{-6}$ |
| Regularization | weight decay 0.05, layer decay 0.75, drop path 0.1, FP16 autocast (BF16 for dense) |
| Augmentation | mixup 0.8, cutmix 1.0, label smoothing 0.1, random erasing 0.25 |
| Clip sampling | 8 frames, stride 8, $224 \times 224$ crop |
| EMA evaluation | decay 0.999, single view |

Table 10: Reference settings. Everything above the recipe blocks is shared between the configurations, unless a recipe block overrides it.

### C.2 Reference Hyperparameters

A checkpoint is trained at each target of the sweep, and the transfer study of Appendix A.2 re-uses the $\tau$=0.5 and $\tau$=0.7 checkpoints. The K400 dataset follows a separate recipe, with fewer epochs, a higher learning rate, and a different augmentation and loss weighting.

### C.3 Baseline Settings

Each baseline uses the official setting closest to the two targets of Table 1. The backbone, the task and the training recipe are ours, so these are official reduction policies on a common EVA backbone rather than reproductions of the published numbers. Every configuration uses a per-device batch size of 32 and AdaMuon. The EK-100 configurations use a learning rate of $2 \times 10^{-5}$ for 100 epochs. The K400 configurations use a learning rate of $10^{-3}$ for 30 epochs and are evaluated from EMA weights in a single view. The training-free reducers update no parameters.

| Method | Official setting used at the two targets |
|---|---|
| Random drop | A uniform surviving patch count per sample, drawn without replacement, with block 0 dense and the CLS always kept. The count is solved per layer by bisection. |
| ToMe | Per-frame bipartite soft matching after attention and before the MLP, removing 14 source tokens per frame at 0.5 and 8 at 0.7, with proportional attention. |
| STA | Temporal grouping and redundant-token pruning after blocks 0, 4 and 8, over a frame hierarchy of 16, 8 and 4, removing 64 tokens per group at 0.5 and 32 at 0.7. |
| Pollard & Wray | Video-aware merging on the packed sequence, with a base count of 441 under an increasing schedule at 0.5 and 196 at 0.7, and without proportional attention. |
| TokenLearner | Per-frame learned pooling by the V1.1 selector, at layer 6 with 16 tokens per frame at 0.5 and at layer 9 with 16 at 0.7. |
| STTS | A temporal selector at block 0 that keeps 11 of 16 frames at 0.5 and 14 at 0.7, then a spatial selector at block 4 over a $13 \times 13$ anchor. |
| Unified STTS | A deletion ratio of 0.4 at block 3, with pool width 3 and a scorer of width 2048. |
| TempMe | Cross-frame merging before attention and within-frame merging after it, at layers 8, 9 and 10, with a clip removal ratio of 0.40 and a frame removal ratio of 0.20. |

Table 11: The official reduction setting that the two target blocks of each baseline in Table 1 correspond to.

# D Method Details

## D.1 Full SAVOR Forward Pass

A reduction stage takes place between two EVA blocks. It removes the same tokens during training and inference, since the deletion is physical in both, and only the gradients that train the scorer and the allocator are specific to training.

**Reduction Stage.** SAVOR shortens the sequence in eleven stages, each placed immediately before one of the last eleven blocks, so only the first block sees the full dense sequence. Each stage removes a small number of tokens, and later stages can adjust their own budgets using the realized retention so far.

**Overview.** Table 12 gives the forward pass in execution order. A reduction stage comprises steps 1–6, i.e., PreMerge, budget allocation, scoring, the training-only oracle, selection, and feedback. Sections 3.2 and 3.3 describe these steps; the table lists them in that order and marks the ones that are disabled at inference.

## D.2 Temporal Admissibility Rule

At stage $R_i$, ATR targets $K_b$ feature tokens. It starts from $N_b^{R_i,\mathrm{pm}} - 1$ of them and lowers that count two ways, by an accepted merge and by a drop, each of which removes exactly one token. The stage reaches its target when enough pairs are admissible to merge and ends up shorter when they are not.

PreMerge and the ATR merge branch follow the same protocol for admitting a pair. A pair is admissible when the two tokens share a spatial position and their temporal positions are strictly less than two frames apart. Two tokens that pass this test are also consecutive at their position, since a surviving token lying between them would already put at least two frames between their centers. In one application, the pair matcher consumes each token at most once and returns a deterministic maximal matching over the admissibility graph of the current sequence. Each stage applies the pair matcher twice, once in PreMerge and once in the ATR merge branch. A merged token is a new token of the sequence, so a later application can match it again.

Table 12: Full SAVOR forward pass. "Pack" concatenates the samples and updates their cumulative lengths. Steps marked "training only" are disabled at inference.

**Input:** clip batch $X$, EVA blocks $\{\mathrm{EVA}_\ell\}_{\ell=0}^{L-1}$, target $\tau$, and stages $R_i$ with $R_i$ immediately before block $\ell_{c+i}$.
**Initialize:** patch-embed $X$ and add positional embeddings. Build token metadata $(t, h, w, \mu)$ with $\mu = 1$ for every feature token, and mark the CLS. Pack features, RoPE values, and cumulative lengths into $(X_{\mathrm{pack}}, \mathrm{rope}, \mathrm{cu})$. Reset per-forward statistics and the HFA state.

**For** $\ell = 0, \ldots, L-1$:
**If** $\ell \geq c+1$ (reduction stage $R_i$, $i = \ell - c$):

**1. PreMerge** — *merge near-identical temporal neighbors.* Let $N_b^{R_i}$ be the current per-sample lengths. Merge same-position, temporally adjacent tokens whose cosine similarity exceeds $\delta$ and whose continuous centers satisfy $|\Delta t| < 2$, then update centroids and merge masses by Eq. 1 and regenerate ST-RoPE. The merge count is capped by the retention floor $\pi_{\mathrm{floor}}$, i.e. at least a fraction $\pi_{\mathrm{floor}}$ of the feature tokens survives (Section 3.2.1).

**2. Allocate** — *decide the budget for this stage.* Record the post-PreMerge lengths $N_b^{R_i,\mathrm{pm}}$, form the detached HFA context $(u^i, d_b^i, r_{b,\mathrm{pm}}^i)$, and produce the per-sample retention ratio $\rho_b^{R_i}$ from the shared anchor and the current HFA state (Section 3.2.2).

**3. Token Scorer** — *rank the surviving feature tokens.* Detach the token features and compute the CLS-conditioned logit $s_{b,j}$ of Eq. 7. The CLS is assigned a logit of 10 so that it always outranks the feature tokens, and it is never dropped or merged.

**4. Oracle** (training only) — *supervise the ranking.* Use the Q/K projections of the consuming block $\mathrm{EVA}_\ell$ to compute the detached CLS-to-feature-token attention oracle $a_{b,j}$ on the post-PreMerge sequence, which is the target of $\mathcal{L}_{\mathrm{rank}}$ and is not computed at inference.

**5. Select** — *keep, merge, or drop.* Convert $\rho_b^{R_i}$ to the integer target $K_b$ and split it into the preferred direct count $D_b$ and the realizable merge budget $B_{b,\mathrm{mrg}}$ (Appendix D.2). Sort the feature tokens by score and retain the top $K_b - B_{b,\mathrm{mrg}}$ directly. From the remaining low-score pool, greedily match admissible pairs (same $(h, w)$ and $|\Delta t| < 2$), merge the accepted ones, drop the unmatched tokens, and repack $(X_{\mathrm{pack}}, \mathrm{rope}, \mathrm{cu})$. If fewer admissible pairs exist than the budget allows, the output is simply shorter than $K_b$. The decisions are hard in the forward pass; the gate and the soft token count are constructed to act only on the backward pass, leaving every hard value unchanged (Appendix E.1).

**6. Feedback** — *report the realized retention.* Measure the feature-token retention $r_{b,\mathrm{atr}}^i = (N_b^{R_i,\mathrm{atr}} - 1)/(N_b^{R_i,\mathrm{pm}} - 1)$, then update the HFA GRU with the detached value and record the hard token counts.

**Run** $\mathrm{EVA}_\ell$ with varlen attention and its MLP on the current packed sequence.

**After the loop:** apply the final normalization and visual head. During training, compute the task loss, the Pearson ranking loss, and the FLOPs penalty on $\widehat{\Phi}_F - \tau$. At inference, report the hard $\Phi(X)$ and disable these training-only quantities.

Two further details about this protocol matter. First, the distance is measured on the continuous temporal position of a token, not on an integer frame index. For example, a token merged from frames 0 and 1 sits at position 0.5, so its distance to frame 2 is 1.5 and the pair is admissible. Rounding the position to a frame index first would make that distance read as 2 and reject the pair. A later application of the matcher can pair a merged token with a surviving neighbor.

Second, the bound of two frames is not a free hyperparameter. A bound above two frames would allow a merged token to land on a surviving one. For example, frames 1 and 3 share a spatial position and are exactly two frames apart, and merging them would place the result at 2.0, the temporal position of the token at frame 2. Two tokens would then share one temporal position, and temporal position is what orders the tokens within a spatial position. Combined with the 1.5-frame pair above, which must be admissible, this constraint leaves the window $(1.5, 2]$, whose only integer endpoint is 2.

The number of tokens ATR may keep directly is set by a fraction $f$ of its target, bounded below by one and above by the number of feature tokens available:

$$D_b = \min\{N_b^{R_i,\mathrm{pm}} - 1, \max\{1, \lfloor f K_b \rfloor\}\}. \tag{14}$$

The preferred plan is to keep $D_b$ tokens directly and to make up the remaining gap of $K_b - D_b$ with merges, one accepted pair per token. The plan is revised downwards when it would remove more tokens than the stage has to remove in total. Reaching $K_b$ requires removing exactly $N_b^{R_i,\mathrm{pm}} - 1 - K_b$ tokens. Merging more pairs than that would take the count below the target, so the merges are capped

there. The realizable merge budget is the smaller of the two counts:

$$B_{b,\mathrm{mrg}} = \min\{K_b - D_b,\ N_b^{R_i,\mathrm{pm}} - 1 - K_b\}. \tag{15}$$

The excess is kept directly, so $K_{b,\mathrm{dir}}^{\mathrm{h}} = K_b - B_{b,\mathrm{mrg}}$ exceeds the preferred $D_b$ exactly when the total-removal count is the binding one. That leaves a low-score pool of $U_b$ tokens to match within, where $U_b = N_b^{R_i,\mathrm{pm}} - 1 - K_{b,\mathrm{dir}}^{\mathrm{h}}$. The stage closes with

$$N_b^{R_i,\mathrm{atr}} = 1 + K_{b,\mathrm{dir}}^{\mathrm{h}} + A_b^{\mathrm{h}} \tag{16}$$

tokens, and $A_b^{\mathrm{h}}$ counts the pairs accepted from that pool. A pool token that no accepted pair consumes is dropped.

The budget fixes how many pairs a sample may merge, and matching decides which ones they are. It proceeds in two passes.

The first pass enumerates the candidates. Within each sample, the scan walks the tokens at each spatial position in temporal order and pairs those that are admissible. Within a single application of the matcher, a token is consumed at most once, since a token merged into a centroid no longer exists to be merged again. Walking in temporal order makes one sweep enough to collect every admissible pair. The sweep also counts the admissible pairs it found in each sample, before the budget is applied. This count is the admissible-pair capacity $C_b^{\mathrm{h}}$ that the soft token count uses (Appendix E.1).

The second pass selects. Each sample takes the pairs its sweep found, in the order the sweep produced them, until its budget $B_{b,\mathrm{mrg}}$ is exhausted. That order therefore decides which pairs survive, at no additional cost.

If a pool holds fewer admissible pairs than its budget allows, fewer pairs are accepted, and the tokens that would otherwise have been merged are dropped. The stage cannot compensate for the missing merges by keeping extra tokens, because the direct-keep count was fixed from the planned budget before matching began. The sample therefore ends up shorter than the nominal $1 + K_b$. The anchor solver allows for this, estimating how many pairs a stage will actually merge using the admissible-pair ratio $\chi_i$ (Appendix D.5) rather than assuming that every pair permitted by the budget can be realized.

### D.3 Allocator Initialization

Every part of the allocator is initialized to do nothing at the start of training, which holds the first forward passes at exactly the calibrated anchor.

The projection head $\mathrm{Proj}(\cdot)$ is a linear layer whose weights and bias are zero-initialized, so its raw output is zero, $\tanh(0) = 0$, and $\rho_b^{R_i} = \rho_a$ on the first forward pass. The FiLM map $W_f \in \mathbb{R}^{2H_{\mathrm{ctl}} \times 3}$ turns the three context scalars into a scale and a shift for each hidden unit. It also starts as an identity, because its weights are zero-initialized, its scale bias is one, and its shift bias is zero. The context therefore leaves the state unchanged.

For the GRU, zeroing the weights alone would not make it an identity map. Both gates would sit at $\sigma(0) = 0.5$ and the candidate state at $\tanh(0) = 0$, so the update $\mathbf{h}' = (1 - z)\mathbf{n} + z\mathbf{h}$ would reduce to $\mathbf{h}' = 0.5\mathbf{h}$, halving the state at each update. By the time the last stage reads the state, the sample-level information that $W_c$ injected would be reduced to $0.5^{10} \approx 9.8 \times 10^{-4}$ of its value. We therefore leave the weights at zero and set the update-gate bias to $+6$, which puts the update gate at $z = \sigma(6) = 0.9975$ and makes the cell an identity map to within $0.25\%$ per stage. The bias is learnable, so the update gate relaxes on its own as training proceeds.

### D.4 FLOPs Accounting

**The computation of $\Phi(X)$.** Table 13 is the accounting behind $\Phi(X)$ in Eq. 9. The block term of $\Phi$ uses the number of tokens entering each block, and the scorer term uses the post-PreMerge token count at the corresponding reduction stage. The per-sample costs are evaluated before they are averaged, so a batch of unequal lengths enters as $\mathrm{mean}_b\, \mathcal{F}_{\mathrm{blk}}(N_b^{\ell})$ and not as $\mathcal{F}_{\mathrm{blk}}(\mathrm{mean}_b\, N_b^{\ell})$. The two quantities differ whenever the lengths are unequal, and the second, the cost evaluated after averaging, is smaller, because the attention term is quadratic and the mean of the squared lengths exceeds the square of the mean length.

The accounting is analytical rather than profiled because the anchor calibration needs it to be. At every epoch boundary, the solver evaluates the ratio at 64 candidate anchors, one per bisection iteration, using the lengths it predicts rather than a realized batch (Appendix D.5). A profiler cannot measure a sequence that has not been computed.

| Counted operation | Analytical proxy |
|---|---|
| *EVA block at token length $n$* | |
| EVA Q/K/V projections | $6nD^2$ |
| EVA attention output projection | $2nD^2$ |
| Attention $QK$ and $AV$ products | $4Dn^2$ |
| Softmax proxy | $5H_{\mathrm{blk}}n^2$ |
| Inner attention normalization | $5nD$ when enabled |
| SwiGLU MLP | $n(6DD_{\mathrm{mlp}} + 6D_{\mathrm{mlp}})$ |
| Two block normalizations and LayerScale | $12nD$ |
| *EVA block aggregate* | $\mathcal{F}_{\mathrm{blk}}(n) = a_{\mathrm{lin}}n + (4D + 5H_{\mathrm{blk}})n^2$ |
| *Linear coefficient* | $a_{\mathrm{lin}} = 8D^2 + 17D + 6DD_{\mathrm{mlp}} + 6D_{\mathrm{mlp}}$ |
| *SAVOR scorer at a reduction stage* | |
| Key projection | $2DD_s n^{\mathrm{pm}}$ |
| CLS query projection | $2DD_s$ (once per sample) |
| Feature-token dot products | $2D_s(n^{\mathrm{pm}} - 1)$ |
| *Scorer aggregate* | $\mathcal{F}_{\mathcal{R}}(n^{\mathrm{pm}}) = 2DD_s(n^{\mathrm{pm}} + 1) + 2D_s(n^{\mathrm{pm}} - 1)$ |

Table 13: Terms counted in $\Phi(X)$ for the reference EVA configuration. Lengths $n$ and $n^{\mathrm{pm}}$ include CLS. The aggregate rows assume the reference configuration, in which QKV is fused, the MLP is scaled SwiGLU, inner attention normalization is enabled, and the LayerScale term is present. The scorer key projection covers all $n^{\mathrm{pm}}$ post-PreMerge tokens, its query projection is computed once per sample, and its dot products consume one key per feature token. One multiply–accumulate counts as two FLOPs.

**Omitted terms.** The FLOPs proxy $\Phi$ measures the video encoder, so the text encoder and the retrieval losses stay outside it. The remaining omissions all lie inside the encoder, and each is there for one of two reasons: either the term is small next to a block, or no established convention says how to count it. They are the patch embedding and the final visual head, the PreMerge matcher and repacker, HFA with its FiLM and GRU, and RoPE. The oracle recomputation is left out because it exists only during training.

### D.5 Anchor Calibration

**The calibration problem.** HFA writes every budget as a bounded deviation from one shared anchor $\rho_a$, so the anchor is what those deviations are measured against. It is defined as the per-stage feature-token retention that would reproduce $\tau$ if every sample and every stage retained exactly that ratio. It has no closed form, because the ratio a stage is given is not the token count it returns. Three gaps stand between the two. PreMerge removes tokens before HFA sets its target, so the stage starts from $\widetilde{N}^{(i)}$, the length after PreMerge, rather than from $N^{(i)}$. The target $K_i$ is not what ATR returns, which is $K_{i,\mathrm{dir}} + A_i$ after the merges. And $A_i$ depends on the data, since it is bounded by the admissible pairs the pool holds.

**The length schedule without the anchor.** Without the shared anchor, each stage will adaptively set its own ratio. At $\tau$=0.7, the lengths entering all twelve blocks were $3137, 3137, 3137, 2555, 2073, 2073, \ldots, 2073$. The first three blocks keep the sample intact, the stages before the fourth and fifth blocks account for the whole of the reduction, and the length then holds at 2073 for the remaining eight blocks. The realized proxy ratio of that training run is $0.70$, at its target, so the ratio alone gives no sign of the imbalance.

**The required statistics.** Two of the three gaps need a measured statistic, each taken from the hard forwards of the previous epoch and exponentially smoothed.

- $\widehat{\pi}_i$, the PreMerge retention, the fraction of feature tokens a stage still has after PreMerge.
- $\chi_i$, the admissible-pair ratio, i.e., the ratio of the low-score pool that the sweep matches into admissible pairs. This is the statistic that matters on EK-100, where admissible pairs are scarce and a stage often realizes fewer merges than its budget allows.

Both smooth a ratio of epoch sums rather than an average of per-sample ratios, so that a sample with a large pool is not outweighed by many samples with small ones. A statistic is updated only where its denominator is positive, and otherwise keeps its previous value. Both are detached, so they set the anchor without carrying gradient, and $\chi_i$ additionally enters the soft token count of Appendix E.1.

**The anchor computation.** The solver replays the hard accounting (Appendix D.2) on a continuous single sample with the sample index dropped. For a candidate anchor $\rho$, it starts from $N^{(1)} = N_0$ and propagates a length schedule. Each stage is first shrunk by $\widehat{\pi}_i$, and the target $K_i$ and the merge budget $B_{i,\mathrm{mrg}}$ then follow from $\rho$ by the hard rule. Only the number of merges has to be estimated, since it depends on the data, and the solver estimates it as

$$A_i = \min\{B_{i,\mathrm{mrg}}, \chi_i U_i/2\}, \qquad N^{(i+1)} = \max\{2, 1 + K_{i,\mathrm{dir}} + A_i\}. \tag{17}$$

Counts stay continuous, so the schedule varies continuously with $\rho$. The anchor is then the root of $\Phi(\rho_a) = \tau$, found by bisection over $[\rho_{\min}, \rho_{\max}]$ with 64 fixed iterations and no early stop. $\Phi(\rho)$ is the expression in Eq. 9 taken at the predicted lengths, with the batch mean dropped and the cost terms and normalizer unchanged.

**The reference configuration.** Table 14 gives the calibration for the reference EK-100 checkpoint at $\tau$=0.5, substituting the two statistics of that configuration (Table 15) into the recurrence. The first two stages are written out in full, and the remaining nine repeat them, each taking the previous output length as its input.

The bisection reaches that anchor by probing the midpoint of the current bracket and keeping the half that still contains the target. $\Phi(\rho)$ returns $0.468$ at $\rho = 0.94$ and $0.549$ at $\rho = 0.96$. Those two values of $\rho$ lie on either side of the root, and the two returned values lie on either side of $\tau$. Starting from $[0.6, 1.0]$, twelve halvings take the bracket to $[0.94844, 0.94854]$. The remaining iterations of the fixed budget of 64 only refine digits beyond the last one quoted here. The root is unique when $\tau$ is reachable, because $\Phi$ rises monotonically in $\rho$. When $\tau$ is not reachable, the solver clamps to the nearest boundary and records the fact. The anchor is not a constant of the method. A stricter target lowers it, and a lower realized PreMerge retention $\widehat{\pi}_i$ raises it, since PreMerge has then already removed part of what the target requires and ATR can retain more of what remains.

# E Gradients and Objectives

The three subsections below cover how gradient crosses the discrete selection, where it flows through the recurrence (Table 16), and how the ranking objective is evaluated numerically, since the Pearson correlation it is built on is not numerically stable as written.

## E.1 Straight-Through Estimators

The direct-keep selection is a hard decision: the top $K^{\mathrm{h}}_{b,\mathrm{dir}}$ feature tokens are kept, and the remaining tokens enter the low-score pool, where they are merged in admissible pairs or dropped. Sorting, matching, and repacking have no derivative, so two quantities in this decision lack a gradient in the forward pass, i.e., the distance from the direct-keep boundary of the token probabilities $p_{b,j} = \sigma(s_{b,j})$, which gives the scorer its task-loss gradient, and the number of tokens the sample retains, which trains the allocator. Both gradients are built from straight-through estimators, which forward the hard value and backpropagate through its continuous replacement.

**Direct-keep gate.** The first quantity is the boundary between the directly kept tokens, the top $K^{\mathrm{h}}_{b,\mathrm{dir}}$ ranks, and the low-score pool. The gate compares the token probability $p_{b,j}$ with a threshold $\vartheta_b$ obtained from a local fit: we take the window of $w_b = \min\{N_b^{R_i,\mathrm{pm}} - 1, w_{\mathrm{STE}}\}$ ranks centered on the hard direct-keep count $K^{\mathrm{h}}_{b,\mathrm{dir}}$, and fit an affine function of rank to the detached probabilities

Table 14: The anchor calibration of Appendix D.5 on the reference EK-100 checkpoint at $\tau$=0.5, with the statistics of that configuration substituted. Counts are in feature tokens except $N$-lengths, which include CLS. Only the first two stages are shown, and the remaining nine repeat them.

**Input:** $\rho = 0.9485$, $f = 0.8$, $\tau = 0.5$, $N_0 = 3137$
the block costs $a_{\text{lin}} = 1.4181 \times 10^7$ and $4D + 5H_{\text{blk}} = 3132$
the scorer width $D_s = 192$
the statistics $\widehat{\pi}_i$ and $\chi_i$ of Table 15
**Denominator:** $L\mathcal{F}_{\text{blk}}(N_0) = 9.0369 \times 10^{11}$.

**Stage 1**, with $\widehat{\pi}_1 = 0.960854$ and $\chi_1 = 0.717309$:
$\widetilde{N}^{(1)} = 1 + (N_0 - 1)\widehat{\pi}_1 = 1 + 3136 \times 0.960854 = 3014.24$
$K_1 = \rho\,(\widetilde{N}^{(1)} - 1) = 0.9485 \times 3013.24 = 2858.06$
$B_1 = \min\{K_1(1-f),\ (\widetilde{N}^{(1)} - 1) - K_1\}$
$\quad = \min\{571.61,\ 155.18\} = 155.18$
$K_{1,\text{dir}} = K_1 - B_1 = 2702.87, \quad U_1 = (\widetilde{N}^{(1)} - 1) - K_{1,\text{dir}} = 310.36$
$A_1 = \min\{B_1,\ \chi_1 U_1/2\} = \min\{155.18,\ 111.31\} = 111.31$
$N^{(2)} = 1 + K_{1,\text{dir}} + A_1 = 2815.19$

**Stage 2**, with $\widehat{\pi}_2 = 0.964986$ and $\chi_2 = 0.387280$:
$\widetilde{N}^{(2)} = 1 + (2815.19 - 1) \times 0.964986 = 2716.65$
$K_2 = \rho\,(\widetilde{N}^{(2)} - 1) = 0.9485 \times 2715.65 = 2575.80$
$B_2 = \min\{K_2(1-f),\ (\widetilde{N}^{(2)} - 1) - K_2\}$
$\quad = \min\{515.16,\ 139.86\} = 139.86$
$K_{2,\text{dir}} = K_2 - B_2 = 2435.94, \quad U_2 = (\widetilde{N}^{(2)} - 1) - K_{2,\text{dir}} = 279.71$
$A_2 = \min\{139.86,\ 54.16\} = 54.16, \quad N^{(3)} = 1 + 2435.94 + 54.16 = 2491.10$

**Stages 3–11** repeat that block, taking $(\widehat{\pi}_i, \chi_i)$ from Table 15 and $N^{(i)}$ from the previous line as input.

**Accumulate:**
block 0, still at the dense length: $\mathcal{F}_{\text{blk}}(3137)/L\mathcal{F}_{\text{blk}}(N_0) = 1/12 = 0.083333$
stage 1 scorer: $\mathcal{F}_{\mathcal{R}}(3014.24)/L\mathcal{F}_{\text{blk}}(N_0) = 0.000985$
block 1, at the length stage 1 returned: $\mathcal{F}_{\text{blk}}(2815.19)/L\mathcal{F}_{\text{blk}}(N_0) = 0.071645$
$\cdots$ the remaining twenty terms follow the same pattern $\cdots$
twelve blocks + eleven scorers $= 0.49343 + 0.00656 = 0.49999$

Table 15: The reference EK-100 checkpoint at $\tau$=0.5: the two statistics the solver reads from the configuration, and the lengths it propagates from them at $\rho_a$=0.9485, starting from $N^{(1)} = N_0 = 3137$. Every other quantity in the propagation follows from these by the hard rule, as the text works through for stage 1. $N$-lengths include CLS.

| stage $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\widehat{\pi}_i$ | .9609 | .9650 | .9703 | .9683 | .9734 | .9721 | .9695 | .9710 | .9750 | .9792 | .9848 |
| $\chi_i$ | .7173 | .3873 | .4481 | .3891 | .3973 | .3201 | .3656 | .3067 | .3307 | .2562 | .2435 |
| $\widetilde{N}^{(i)}$ | 3014.2 | 2716.7 | 2417.2 | 2153.7 | 1922.7 | 1715.0 | 1518.9 | 1350.8 | 1202.3 | 1076.3 | 964.8 |
| $N^{(i+1)}$ | 2815.2 | 2491.1 | 2224.1 | 1975.1 | 1764.1 | 1566.7 | 1391.1 | 1233.1 | 1099.1 | 979.7 | 877.6 |

in that window. Evaluating the fit at the hard count gives the threshold $\vartheta_b$ that the forward pass uses; the soft direct-keep count $K^{\mathrm{s}}_{b,\mathrm{dir}}$ enters only through the straight-through correction $K^{\mathrm{h}}_{b,\mathrm{dir}} + (K^{\mathrm{s}}_{b,\mathrm{dir}} - \mathrm{sg}(K^{\mathrm{s}}_{b,\mathrm{dir}}))$, so the hard threshold stays detached from the allocator. The fit is local to the ranks whose order can change the selection. The gate combines a hard binary indicator with a soft gate,

$$g^{\mathrm{s}}_{b,j} = \sigma\left(\frac{p_{b,j} - \vartheta_b}{\tau_g}\right), \qquad g^{\mathrm{h}}_{b,j} = \mathbf{1}[\mathrm{rank}_{b,j} < K^{\mathrm{h}}_{b,\mathrm{dir}}], \qquad g_{b,j} = g^{\mathrm{h}}_{b,j} + g^{\mathrm{s}}_{b,j} - \mathrm{sg}(g^{\mathrm{s}}_{b,j}), \quad (18)$$

where the soft gate $g^{\mathrm{s}}_{b,j}$ converts the distance to the threshold into a differentiable weight, the hard binary indicator $g^{\mathrm{h}}_{b,j}$ is one for the tokens in the top $K^{\mathrm{h}}_{b,\mathrm{dir}}$ ranks and zero otherwise, and the combined gate $g_{b,j}$ forwards the hard value while backpropagating through the soft gate. Ranks are zero-based and sorted by decreasing logit, and the temperature $\tau_g$ controls the smoothness of the boundary; its annealing schedule and $w_{\mathrm{STE}} = 32$ are listed in Table 10.

**Gate integration.** The gate reaches the computation through two terms. The *scaling term* multiplies each directly kept feature token by its gate, $\tilde{\boldsymbol{x}}_{b,j} = g_{b,j}\boldsymbol{x}_{b,j}$. The *summed term* adds $S_b - \mathrm{sg}(S_b)$ to the feature vector of the directly kept token of rank zero, where $S_b = \sum_{j=1}^{N_b^{R_i,\mathrm{pm}}-1} g_{b,j}$ ranges over every feature token of the sample. In the forward pass, the gate equals the hard binary indicator, so both terms reduce to the identity and act only on the backward pass. Appendix E.2 analyzes what each of the two terms supervises.

**Soft token count.** The second quantity is the count of retained tokens. The soft token count introduced in Section 3.3 is computed as follows. The continuous target and its preferred direct-keep part are $K^{\mathrm{s}}_b = \mathrm{clip}_{[1,N_b^{R_i,\mathrm{pm}}-1]}\left((N_b^{R_i,\mathrm{pm}} - 1)\rho_b^{R_i}\right)$ and $D^{\mathrm{s}}_b = \min\{N_b^{R_i,\mathrm{pm}} - 1, \max\{1, fK^{\mathrm{s}}_b\}\}$. From these, the merge budget and the low-score pool follow the hard rule, $B^{\mathrm{s}}_{b,\mathrm{mrg}} = \min\{K^{\mathrm{s}}_b - D^{\mathrm{s}}_b, N_b^{R_i,\mathrm{pm}} - 1 - K^{\mathrm{s}}_b\}$ and $U^{\mathrm{s}}_b = N_b^{R_i,\mathrm{pm}} - 1 - K^{\mathrm{s}}_b + B^{\mathrm{s}}_{b,\mathrm{mrg}}$. The soft admissible-pair capacity $\widetilde{C}^{\mathrm{s}}_b$ follows the same pattern: it forwards the admissible-pair capacity $C^{\mathrm{h}}_b$ and backpropagates through the EMA-predicted admissible-pair capacity $\frac{1}{2}\chi_i U^{\mathrm{s}}_b$ (half the low-score pool, scaled by the admissible-pair ratio $\chi_i$). The expected merge count $A^{\mathrm{s}}_b$ is the smaller of the budget and $\widetilde{C}^{\mathrm{s}}_b$. The count that enters the FLOPs penalty is $n_{b,\mathrm{STE}} = n^{\mathrm{h}}_b + n^{\mathrm{s}}_b - \mathrm{sg}(n^{\mathrm{s}}_b)$, with the hard count $n^{\mathrm{h}}_b = 1 + K^{\mathrm{h}}_{b,\mathrm{dir}} + A^{\mathrm{h}}_b$, where $A^{\mathrm{h}}_b$ is the number of accepted merges, and the continuous estimate $n^{\mathrm{s}}_b = 1 + K^{\mathrm{s}}_b - B^{\mathrm{s}}_{b,\mathrm{mrg}} + A^{\mathrm{s}}_b$. The EMA for the admissible-pair ratio, the admissible-pair capacity, and the probabilities used for the local fit are detached; the two constructions enter the graph only through the continuous gate and count variables.

**The role of the EMAs.** Two detached, stage-level EMAs track realized PreMerge retention and the admissible-pair ratio. Both inform the anchor solver in Appendix D.5, and the admissible-pair-ratio EMA also enters the soft token count above. Updating them only at epoch boundaries separates fast content-dependent HFA decisions from slower population-level calibration, without treating the statistics as learnable parameters.

### E.2 Gradient Routing

Two properties of the routing in Table 16 show why the task loss cannot supervise ranking on its own.

The task loss reaches the scorer through the scaling and the summed terms defined above, and the two terms carry different information. The scaling term is selective: it gives a token-specific gradient to every feature token that survives directly. The summed term contributes the same scalar coefficient, $\partial\mathcal{L}/\partial S_b$, to the gate of every feature token of the sample, so it can adjust how much of a sample is retained but carries no ordering information about which tokens to prefer. Ranking supervision over the low-score pool therefore cannot come from the task loss alone; it is supplied by the ranking loss of Section 3.3.

The task loss cannot reach the feature of a dropped token at all, and the merge branch is only partially covered. Dropped tokens are physically absent from every later block. A merged centroid

| Component | Live differentiable path | Detached or non-differentiable |
|---|---|---|
| Task | Surviving and merged features; scorer via both gate terms; allocator via the threshold | Dropped features; hard ranks, pair indices, and repack destinations |
| FLOPs | HFA via the soft token count | Hard integer counts |
| Ranking | Scorer parameters only | Oracle, scorer inputs, backbone |
| Recurrence | Current-stage GRU, FiLM, and head parameters, through the paths listed above | Previous hidden state and realized retention (detached GRU inputs) |

Table 16: Gradient routing under one-step truncation. The live CLS initialization $W_c$ receives gradient only at the first reduction stage, because the first GRU update detaches its state input.

is differentiable with respect to its source features, but the pairing itself is discrete: the pair matcher has no differentiable counterpart, so no task gradient expresses whether a particular pair should have been merged. The ablation in Section 4.3 shows what removing the ranking loss costs.

The remaining rows follow from the forward construction. The FLOPs penalty acts on the allocator through the soft token count, while the hard integer counts are detached. The ranking loss trains the scorer parameters only; the oracle, the scorer inputs, and the backbone are detached from it. The recurrence is one-step truncated: the previous hidden state and the realized retention are detached, so gradient does not cross stage boundaries, but the GRU parameters themselves are not frozen and receive gradient from the update at every stage.

### E.3 Pearson Ranking Loss

This subsection documents two implementation details of the ranking loss defined in Section 3.3, i.e. how the oracle is computed and how the Pearson correlation is evaluated. The oracle uses detached copies of the Q and K weights of the consuming block so that gradient-checkpointing does not mix oracle activations into the saved graph. It is also recomputed on the post-PreMerge sequence, so merged tokens contribute keys derived from their centroid features and centroid coordinates rather than from either source token.

The Pearson reduction is evaluated in FP32. The norm floor $\epsilon_n$ is chosen per input precision. Each centered sum of squares is floored at $\epsilon_n^2$ before its square root is taken. Non-finite values are replaced by zeros for the reduction, and their samples are excluded, as are samples with fewer than two feature tokens or an oracle centered sum of squares at or below $\epsilon_n^2$. A nearly constant *predicted* score remains valid and receives a finite gradient. If no sample is valid, the ranking loss is zero.

## F Experimental Protocols

### F.1 Evaluation Protocol

**Retrieval and baseline protocol.** The EK-100 evaluator extracts video and text embeddings over the retrieval test split, aligns them with the relevance matrix supplied with the dataset, and computes both retrieval directions before averaging their mAP and nDCG. The varlen dense backbone disables reduction on the packed sequence. The original dense backbone uses the standard dense execution path. Both retain all feature tokens.

The target sweep holds every non-target setting fixed and applies the same calibration procedure at each target, so quantities computed from $\tau$, such as the solved anchor, differ between rows while nothing else does. Single-view K400 classification is not directly comparable with published multi-view evaluations.

## G Limitations

**Training memory does not follow FLOPs.** Physically removing tokens does reduce both the peak allocated memory and the reserved high-water mark of the CUDA caching allocator, but the saving is small and does not track the FLOPs reduction. At $\tau$=0.7 (Table 2) the reserved GPU memory is 14,234 MiB against 14,750 MiB for the dense model, the smallest margin in the sweep at 3.5%, and the largest margin is at $\tau$=0.4, where the reserved GPU memory falls to 12,636 MiB. Three effects

keep the saving small. First, the reduction machinery, including the ATR scorer, allocates per-stage buffers that a dense forward does not need. Second, the packed sequence changes length at every stage, which leaves the caching allocator holding segments it cannot reuse. Third, the first block of the backbone always processes the full dense sequence, so each forward carries one long sequence whose footprint has to coexist with the shortened ones, and that fragments the caching allocator further. A fixed allocation scheme, such as bucketing the packed length or keeping one workspace per bucket, would mitigate the latter two effects.

**The anchor is calibrated rather than jointly learned.** The shared anchor is solved by bisection outside the gradient path from epoch-smoothed statistics and recalibrated at epoch boundaries, while the learned deviations around the anchor respond during the forward pass to stage progress, compression debt, and PreMerge retention. The admissible-pair ratio informs the anchor solver and the soft token count instead. The learned deviations are measured against a fixed ratio in Table 3 and Appendix A.6. Two questions remain open. The first is whether the two-stage design of a calibrated anchor plus learned deviations is the best way to realize the adaptation. The second is how much of the reported accuracy gap comes from selection quality rather than from training effects, since the control is not compute-disadvantaged.

**The attention oracle is recomputed rather than observed.** The oracle logit is the attention that the consuming block would assign to each feature token. It is recomputed from the Q/K projections of that block, so the scorer receives a supervision target that the attention layer itself never reports. Training and inference share one attention path and therefore behave identically, and the recomputation costs one extra Q/K computation per reduction stage during training. That cost stays outside $\Phi(X)$. A training-only attention path would remove it at the price of maintaining two attention implementations.