Don't Look Twice: Faster Video Transformers with Run-Length Tokenization

Rohan Choudhury [1] **Guanglei Zhu** [1] **Sihan Liu** [1] **Koichiro Niinuma** [2]
Kris M. Kitani [1] **László A. Jeni** [1]
[1] Carnegie Mellon University [2] Fujitsu Research
rchoudhu@andrew.cmu.edu

Abstract

Transformers are slow to train on videos due to extremely large numbers of input tokens, even though many video tokens are repeated over time. Existing methods to remove such uninformative tokens either have significant overhead, negating any speedup, or require tuning for different datasets and examples. We present Run-Length Tokenization (RLT), a simple approach to speed up video transformers inspired by run-length encoding for data compression. RLT efficiently finds and removes 'runs' of patches that are repeated over time prior to model inference, then replaces them with a single patch and a positional encoding to represent the resulting token's new length. Our method is *content-aware*, requiring no tuning for different datasets, and *fast*, incurring negligible overhead. RLT yields a large speedup in training, reducing the wall-clock time to fine-tune a video transformer by 30% while matching baseline model performance. RLT also works *without any training*, increasing model throughput by 35% with only 0.1% drop in accuracy. RLT speeds up training at 30 FPS by more than 100%, and on longer video datasets, can reduce the token count by up to 80%. Our project page is at https://rccchoudhury.github.io/projects/rlt/.

1 Introduction

Vision transformers [11] have enjoyed enormous success in modeling images and videos due to their scaling properties and minimal inductive bias. Unfortunately, training these models on videos, which generally have orders of magnitude more tokens than images, is significantly more expensive. One contributing factor is that video transformers tokenize videos by splitting them into uniformly sized spatiotemporal patches [2, 3], then embed them into a latent token space. As a result, the number of tokens depends only on the video's length and resolution. Researchers are thus forced to work with very short videos (<10s), as well as significantly downsample them to low frames-per-second (FPS) and low spatial resolution.

One promising solution to this problem is to reduce the number of input tokens. Compared to language input, videos are significantly less dense in information; many works observe that videos consist mostly of redundant or uninformative tokens [15, 39, 43]. However, existing methods that aim to reduce input tokens to vision transformers have had limited adoption. Learned pruning methods [34, 52] reduce model complexity measured by GFLOPS, but either incur significant overhead during training, or require padding to handle changing numbers of tokens, negating any speed-up during training. Random masking [1, 27], though fast, decreases accuracy and thus requires more training time to match performance. Moreover, although methods like random masking and Token Merging [5] do lead to wall-clock speedups, they are not content-aware: they only remove a fixed number of

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0882

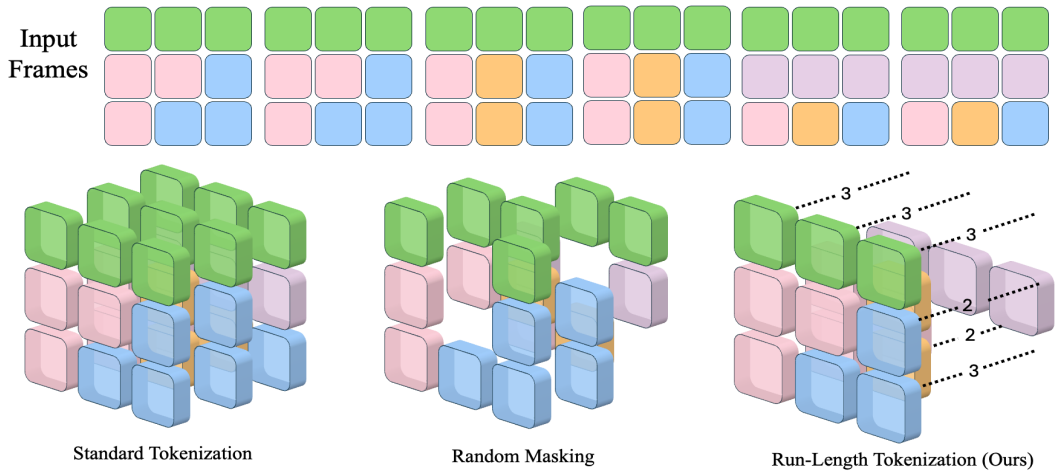

Figure 1: **Toy Example.** Given a set of input frames, with each square representing a patch, standard tokenization always produces the same number of tokens. RLT compares temporally consecutive patches and removes redundant ones, storing a single token and the run-length instead.

tokens per video, and will reduce the same number of tokens from a high-speed, high-action clip as from a still image repeated over time.

We argue that content-awareness can help more effectively reduce the number of input tokens. As an example, imagine an hour-long video of a lecture. Most of the frames are exactly the same over time, displaying a single slide. Existing methods would produce the same number of tokens from this as from an hour of motion-heavy GoPro footage, even though the two videos have significantly different amounts of content. On the other hand, video compressors, such as H.264 and H.265 [46, 41], are explicitly content-aware: rather than encoding frames independently, they encode pixel differences between consecutive frames, drastically reducing video size when there is no change.

We propose Run-Length Tokenization (RLT), which combines a simpler version of this idea with classical run-length encoding to tokenize videos for transformers. Our insight is that we can efficiently identify 'runs' of input patches that are repeated over time, enabling us to reduce the number of tokens based on the video content. When tokenizing the video, we compare consecutive patches in time and group together patches with sufficiently small differences. We then remove the "repeated" patches, and treat the remaining tokens as having variable length. Similar to how the string `aaaabb` can be run-length encoded as `a4b2`, we can add length information to each of the tokens, which incurs no additional overhead while retaining some of the information lost from removing the redundant tokens. Despite its simplicity, RLT works remarkably well - with it, we can fine-tune a video transformer in 40% faster wall-clock time than baseline ViTs while matching performance.

Our contributions are as follows: we (1) propose RLT, an alternative method to tokenize videos for vision transformers, (2) thoroughly compare its performance and compare RLT's speed to prior methods, finding significant improvements, (3) evaluate RLT's performance on high-FPS and longer videos, and (4) ablate design choices and qualitatively visualize RLT's output. We believe RLT can be a key step to significantly accelerate and further scale video understanding.

2 Related Work

Video Transformers. Vision Transformers [11] have been successfully adapted to video [2, 3, 13, 26, 38] but are generally trained and evaluated on short (<10s) video clips with relatively few frames. To efficiently handle videos, many works incorporate video-specific inductive biases in their architectures [29, 22, 56], such as memory [49, 35], compression cues [48], or modified attention mechanisms [30, 53]. Other methods, especially in video generation, project the video to a smaller latent space [20] and then split it into patches. We instead use the standard ViT formulation but apply a different tokenization scheme, reducing the number of input tokens to improve speed while maintaining performance.

https://doi.org/10.52202/079017-0882

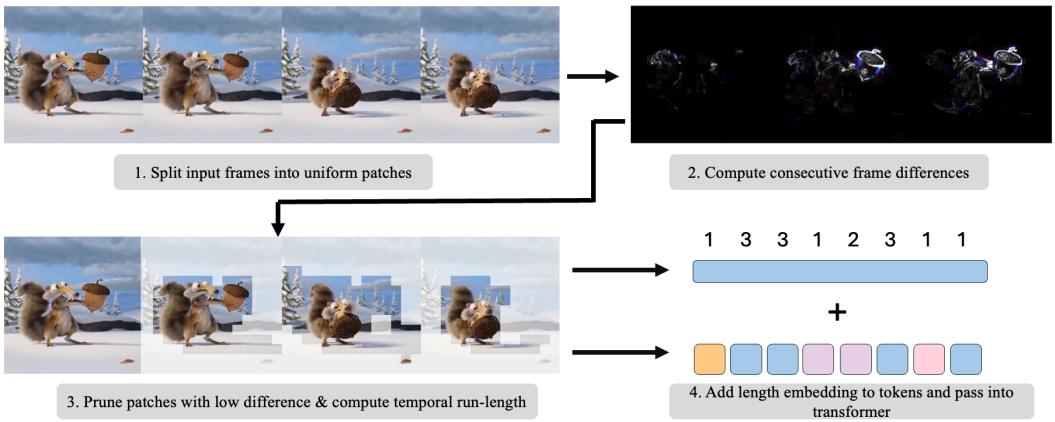

Figure 2: **RLT Overview.** RLT works by comparing temporally consecutive patches, and retaining those with L1 difference above a threshold τ. The remaining tokens are augmented with a length encoding to signify their 'run-length' and passed to the transformer.

Video Tokenization. Prior to vision transformers, video architectures were designed to take in a fixed size input [14, 40, 55]. However, transformers can handle arbitrary numbers of input tokens [44], and training on variable-sized inputs is standard in language modeling [23]; this has been used to train vision transformers with variable resolutions [4, 10]. However, video transformers still generally use the spatiotemporal patch tokenization scheme introduced in [2, 3] which is *content-agnostic*: the number of tokens depends only on the video's length and resolution. Some works attempt to reduce input size by compressing the video to a latent space, then tokenizing [12, 33, 6], but the number of tokens still depends strictly on the input video dimensions. On the other hand, standard video compressors like HEVC [41] and AVC [46] are *content-aware*: they actively consider the differences between consecutive frames for more efficient compression. Our work applies this idea to video transformers by condensing static tokens and tracking their length.

Faster ViTs with Fewer Tokens. Several works have attempted to remove uninformative tokens from vision transformers. One line of work identifies such tokens either through learned modules or attention scores [34, 52, 28, 21], and prunes them at each layer. Although transformers can handle variable sized inputs, these methods require padding as token counts change unpredictably with each layer. Other works combine tokens instead of pruning them ([5, 37, 31, 51]). Most of these works require training a model for pruning or merging, with the exception of Token Merging [5], which demonstrates strong results at inference time. Inspired by the success of masked pre-training ([17, 45, 43, 15]), another line of work uses random masking to speed up training. Although masking leads to worse performance after the same number of batches, the dramatic speedup enables training for more epochs in less time [1, 10, 27, 50]. In contrast, our method matches the performance of base models with the same amount of data with large speedups, and can be stacked with random masking for even more speed benefits. Closely related to our method are EVEREST [18] and STA [36] which both exploit temporal similarity to identify redundant tokens. However, like [5] both these works require setting a constant number of tokens to remove from each video, while RLT can remove varying numbers of tokens based on the video content.

3 Method

Consider a vision transformer that takes as input a video $V \in \mathbb{R}^{C \times T \times H \times W}$. The standard tokenization scheme splits V into a set $\mathbf{P}$ of uniformly sized, non-overlapping patches, each with size $C \times D_x \times D_y \times D_t$, with P_t called the tubelet size. These patches are projected to a lower dimension d_{embed} with an MLP $\mathcal{E}$, resulting in N_P tokens, with each corresponding to a distinct spatiotemporal location. This results in the same number of tokens for any input video that has the same size.

In contrast, our goal is to to identify input patches that are extremely similar, then compress these redundant patches, increasing throughput and training time. Our approach is illustrated in Figure 2. In particular, we focus on *temporally consecutive* patches, those which have the same x, y location

https://doi.org/10.52202/079017-0882

and differ by one timestep. These "static patches" correspond to visual content that does not change or move over time, and such tokens can be easily compressed.

3.1 Removing Static Patches

Token Similarity. Unlike prior works, we aim to reduce the number of total input tokens by comparing *patches* rather than tokens. By operating on patches, we do not need to run the patch embedding $\mathcal{E}$ or any layer of the model. As a result, we do not need to freeze parts of the model or propagate gradients through the pruning operation, which would require padding and negate potential speedups. This contrasts with prior works which progressively prune or combine tokens after each layer in the transformer. Furthermore, by identifying redundant patches, we can pre-compute the token distributions of various datasets and sizes of examples, allowing us to employ techniques like example-packing [23]. Finally, operating on visual patches is more interpretable and is similar to the heuristics used by video encoders [41, 46].

We next define a criterion for determining whether two consecutive patches are static. Consider two temporally consecutive patches P_1, P_2 that correspond to spatial location (x, y) and temporal locations t_1, t_2 with $t_2 = t_1 + D_t$. For tubelet sizes with value $P_t > 1$, each patch consists of multiple frame crops, so that $P_1 = [P_{xy}^{t_1}, P_{xy}^{t_1+1}, ...P_{xy}^{t_1+D_t-1}]$. Given a threshold τ, we consider P_1 and P_2 static if

$$\|P_{xy}^{t_2+D_t-1} - P_{xy}^{t_1}\|_1 < \tau \tag{1}$$

with $P_{xy}^{t_2+D_t-1}$ being the temporally last spatial crop of in P_2 and $P_{xy}^{t_1}$ the first spatial crop of P_1. This operation compares the "start" of the P_1 to the "end" of P_2, with the idea being that if the first crop of token P_1 matches the last crop of token P_2, the patches in between likely match as well. Notably, τ is a hyperparameter that needs to be tuned, but is *dataset-agnostic*; it simply encodes how much change between patches is allowed before they are considered different. τ controls the trade-off between speed and accuracy; while higher values reduce significantly more tokens, they treat tokens that are perceptibly different as being the same, reducing accuracy. We use $\tau > 0$ since imperceptible artifacts can occur, and follow standard procedure by running ImageNet normalization before comparing patches. We typically use $\tau = 0.1$, and provide experiments and visualizations on its effect in Section 4.3 and Appendix B.

Pruning Procedure. To identify all static tokens, we run the prior comparison on all pairs of temporally consecutive patches in $\mathbf{P}$ obtaining their differences and only retaining those with difference less than τ. We always include the entirety of the first frame since there is no previous patch to compare it to. This results in a binary mask M_{static}, which we can then apply with

$$\mathbf{P}' = \mathbf{P} \circ M_{\text{static}} \tag{2}$$

with P' containing $N_{P'}$ tokens and P consisting of N_P tokens. Note that $N_{P'} \leq N_P$ is always true; with RLT, we can never have more tokens than in the standard tokenization procedure, so the worst-case performance matches the standard vision transformer. RLT also incurs essentially no overhead as the entire process can be implemented entirely with parallelizable PyTorch [32] operations on the GPU, so training and inference are strictly faster.

The simplicity of RLT is a major advantage: in contrast to other methods, we can take advantage of transformers' ability to handle variable input sizes, and do not need to provide any additional padding. Because we make no changes to the model itself, a video transformer using RLT can make use of hardware optimizations like Flash Attention [8, 9] and memory efficient kernels [25].

Notably, the pruning procedure is *content-aware*: some videos with large amounts of static content will result in significantly fewer input tokens than videos with significant amounts of camera or subject motion. This is a desired outcome, and we discuss how to handle training with dynamic input sizes in Section 3.3.

3.2 Run-length Positional Encoding

Although we have reduced the number of input patches, we know that each patch represents a 'run' of static patches, with length 1 corresponding to no static content, and length T corresponding to input time dimension length. Without information about the length of the 'run' of static patches, the transformer may not be able to compensate for information removed during the pruning procedure.

To address this, Bolya et al. [5] introduced Proportional Attention, which weights each token by the number of tokens in each group. On the other hand, we opt to let the model *learn* this information: we treat each token as having variable length that we can communicate through a new positional encoding. Specifically, we use a factorized encoding, described in Dehghani et al. [10], with one encoding ϕ_{xyt} containing positional information and the other ϕ_L corresponding to the length. We use a learnable length bias ϕ_L consisting of a single parameter of size (T, d_{embed}). For a given 'run' of repeated patches, we always retain the initial patch P_{xyt}, and thus can compute the new length ℓ_i as the distance from xyt to the nearest 1 entry in M_{static} along the t-axis. Concretely, for P_{xyt}

$$\ell_i = \min_{t'}(t' - t), \quad \text{where} \quad M_{\text{static}}(x, y, t') = 1, t' > t \tag{3}$$

This operation can also be efficiently implemented on the GPU, adding no overhead. Then, the full positional encoding becomes

$$\phi(T_i) = \phi_{xyt}(T_i) + \phi_L[\ell_i] \tag{4}$$

with the $\phi_L[\ell_i]$ representing the indexing operator. We add the positional encoding $\phi(T_i)$ to each token after running the patch embedding network $\mathcal{E}$. Unlike the pruning procedure, since we use a learnable length encoding ϕ_L, we propagate gradients to the positional embedding, enabling the model to learn how to optimally encode variable length tokens during fine-tuning.

3.3 Handling Dynamic Input Sizes

Since RLT is content-aware, the number of tokens varies significantly per example. Although transformers can natively handle any input size [44], prior methods like DynamicViT [34] or A-ViT[52] produce different numbers of tokens at each layer; this requires padding or attention masking to handle batched inference during training. In our case, only the input token count is variable, but the number of tokens stays constant throughout the network, closer to the setting of NaViT [10]. Furthermore, since we know the input size before running the network, we can employ *example packing* [23], an idea from language modeling where multiple inputs with variable sizes are packed together, and tokens from individual examples attend only to each other.

At training time, the input to the transformer consists of a batch of tokenized videos, $V_1, V_2, ...V_B$, each with size $T_1, T_2, ...T_B$. Rather than pass an input $(B, \max_i T_i, d_{\text{embed}})$ to the network, we concatenate the video tensors to produce $V' = V_1 \oplus V_2 \oplus V_3 ... V_B$, resulting in input size $(1, \sum_{i=1}^{B} T_i, d_{\text{embed}})$. We then construct a block-diagonal attention mask so that tokens only attend to other tokens from the same video, which we add during the attention operation. Since every token in V' is attending only to tokens from the same example, this does not reduce throughput and is also compatible with existing hardware-efficient attention implementations. To compute the class prediction in action recognition, we split each example out and compute its prediction as the mean of each example token, as in [43]. We then project it to dimension N_C, resulting in output of size (B, N_C) to which we can apply standard cross-entropy losses during training.

We note that typically example packing results in a constant number of input tokens, with a variable number of input examples. A key difference between RLT and Dehghani et al. [10] is that data augmentations such as RandAugment [7] can alter the visual content and thus number of tokens of input videos, rendering greedy example packing strategies inapplicable during data loading. We opt to use a constant number of examples per GPU, with high enough batch size sufficiently reducing variance in input size.

4 Experimental Results

To analyze RLT's impact on performance and speed, we conduct several experiments on standard action recognition tasks. We measure the speedup on model training at several scales in Section 4.1 as well as RLT's effect as a drop-in addition at inference time in Section 4.2. We perform ablations in Section 4.3, then evaluate RLT's effect on higher FPS videos and long video datasets in Section 4.4. Finally, we provide qualitative visualizations in Section 4.5.

4.1 Training

In Table 1 we evaluate RLT's impact on the performance of video transformers during training and its resulting speedup. We fine-tune ViT-B and ViT-L from pre-trained VideoMAE [43, 45] checkpoints,

https://doi.org/10.52202/079017-0882

	Kinetics-400			Something-Something-v2		
Model	Acc	FT time(8 GPU)	Speedup	Acc	FT time(8 GPU)	Speedup
ViT-B	80.1	14.4h	1.0×	70.3	10.1h	1.0×
ToMe$_{r_{64}}$	80.0	13.4h	1.1×	69.7	9.4h	1.1×
Random (0.7)	79.2	10.2h	1.4×	69.3	7.2h	1.4×
RLT (Ours)	**80.1**	**10.2h**	**1.4×**	**70.2**	**7.2h**	**1.4×**
ViT-L	84.8	21.6h	1.0x	74.3	15.2h	1.0×
ToMe	84.4	18.3h	1.2×	74.3	12.9h	1.2×
Random	83.1	15.4h	1.4×	74.3	10.8h	1.4×
RLT (Ours)	**84.7**	**15.4h**	**1.4×**	**74.4**	**10.8h**	**1.4×**

Table 1: **Training results on action recognition.** RLT significantly reduces fine-tuning time with comparable performance to the baseline on both Kinetics-400 and Something-Something-v2.

comparing the speed and performance with standard tokenization, random masking, and RLT. We evaluate random masking by removing k tokens, with k being the mean number of tokens pruned by RLT on a given dataset. For the most fair speed comparison, all evaluated models are trained with mixed-precision, memory-efficient attention and Flash Attention where possible using an 8xH100 node, as well as the optimized data loader from AVION [54] to avoid data loading bottlenecks. We use the standard Vision Transformer rather than more complex architectures such as TimesFormer [3] or MViT [26]; we found that it was significantly simpler and more efficient, matching observations from Ryali et al. [38]. We limit our analysis to fine-tuning due to computational constraints. We compare against the baseline vision transformer, as well as Token Merging and STA [36]. We also include a random masking baseline where the masking fraction is set to the average number of tokens removed by RLT, which is a stronger baseline than using a fixed standard fraction such as 0.5.

Compared to standard tokenization, RLT achieves a speed-up of up to 40%, even with heavily optimized implementations. RLT achieves the best trade-off between performance and speed, with better performance than random masking while achieving the same speedup. This demonstrates that the choice of which tokens to remove makes a nontrivial difference, and that properly identifying redundant tokens is important.

Compared to standard tokenization, RLT achieves a speed-up of up to 40%, even with heavily optimized implementations. RLT achieves the best trade-off between performance and speed, with better performance than random masking while achieving the same speedup. In particular, RLT is much faster to train than Token Merging since it is compatible with hardware-optimized implementations such as Flash-Attention [8, 9]. Unlike random masking, RLT matches the performance of the baseline ViT after the same number of training batches, while random masking requires significantly more epochs to catch up. RLT matches baseline performance across multiple scales, indicating that RLT does not degrade performance while considerably accelerating training.

4.2 Inference-Time Results

Although RLT was designed to speed up training, it can be used as a drop-in replacement for standard tokenization, similar to Token Merging[5]. In Table 2 we compare the top-1 accuracy, GFLOPs and throughput with RLT to standard tokenization and Token Merging [5]. We also compare against random masking for completeness, although it is intended only for training time [27]. For the most fair comparison, we randomly mask out P tokens for each example, where P is the mean number of tokens used by RLT; for Kinetics-400 and SSv2 this was $P = 0.72$. We do not compare to learned pruning methods like A-ViT [52] since those only present results on images. We measure throughput in clips-per-second, with each model running on a single clip at a time. In practice, video models are evaluated on multiple temporal and spatial crops; following VideoMAE[43] we measure GFLOPs on single clip and measure accuracy with 4 temporal and 3 spatial crops.

Across model sizes, RLT consistently delivers the best tradeoff between speed and accuracy. The benefit becomes more pronounced as model size increases, as at larger parameter counts, the attention operation begins to dominate the computation. Compared to baselines, RLT is significantly faster than Token Merging and outperforms all other baselines on accuracy. Token Merging cannot make use of Flash Attention and other optimizations due to its reliance on a weighted attention operation, slowing

	Kinetics-400				Something-Something-v2			
Model	Acc	GFLOPS	Clips/s	Speedup	Acc	GFLOPS	Clips/s	Speedup
ViT-B	80.5	180	31.4	1.0×	70.8	180	31.4	1.0×
ToMe$_{r_{64}}$	80.4	131	34.4	1.09×	69.1	131	34.4	1.09×
STA$_{r_{64}}$	80.4	131	34.4	1.09×	69.1	131	34.4	1.09×
Random	80.1	120	**53.0**	**1.68**×	69.3	120	**53.0**	**1.68**×
RLT (Ours)	**80.6**	**120**	52.6	1.67×	**69.8**	**120**	52.6	1.67×
ViT-L	84.8	598	11.5	1.0×	74.3	598	11.5	1.0×
STA$_{r_{64}}$	80.4	308	34.4	1.09×	69.1	308	34.4	1.09×
ToMe$_{r_{64}}$	84.3	**285**	**19.3**	**1.68**×	73.6	**285**	**19.3**	**1.68**×
Random	84.1	405	18.8	1.63×	73.3	405	18.8	1.63×
RLT (Ours)	**84.6**	405	18.71	1.62×	**74.1**	405	18.71	1.62×
ViT-H	86.8	1192	6.65	1.0×	-	-	-	-
ToMe$_{r_{32}}$	86.1	766	8.51	1.27×	-	-	-	-
STA$_{r_{64}}$	80.4	**611**	34.4	1.09×	-	-	-	-
Random	85.1	816	9.66	1.45×	-	-	-	-
RLT (Ours)	**86.3**	816	**9.66**	**1.45**×	-	-	-	-

Table 2: **Inference-only results on action recognition.** With batch size 1, RLT with $\tau = 0.1$ consistently achieves the closest performance to the baseline, comparable or faster than Token Merging or random masking. We omit ViT-H results on Something-Something-v2 due to lack of existing pre-trained checkpoints.

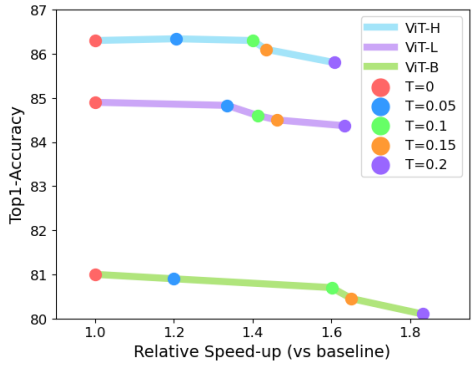

Figure 3: **Varying Difference Threshold.** When comparing the tradeoff between speedup factor and accuracy, RLT is close to baseline performance for low values of τ, with a sharp drop-off after $\tau = 0.1$.

Model	Acc	FT Time
ViT-B	80.1	14.4h
RLT (no length)	80.1	10.2h
RLT	80.1	10.2h
RLT (no length, w/random)	79.3	8.1h
RLT (w/random)	79.8	8.1h
ViT-L	84.8	21.6h
RLT (no length)	84.6	15.4h
RLT	84.6	15.4h
RLT (no length, w/random)	84.2	11.3h
RLT (w/random)	83.3	11.3h

Table 3: **Effect of length encoding.** When fine-tuning with RLT only, length encoding has minimal effect, but helps significantly when combined with random masking.

it down in comparison to RLT. Although worse than RLT, random masking performs surprisingly well, likely due to the fact that most tokens in videos are redundant. Random masking can also be combined with RLT for further speed benefits, with smaller resulting performance gaps than in [27]. However, achieving the optimal performance-throughput tradeoff with random masking requires tuning for each dataset, while RLT is natively content-aware, achieving higher accuracy at similar speeds without tuning. Similarly, Token Merging [5] requires changing the r parameter based on the model size and is not content aware, limiting its speed-up in highly static videos.

4.3 Ablations

We ablate our design choices for RLT in Figure 3 and Table 3, measuring the impact of the difference threshold and length encoding design choices at multiple model scales during training.

https://doi.org/10.52202/079017-0882

Dataset	FPS	#Tokens	RLT
K400	7.5	3.8×10^8	2.7×10^8 (-29%)
K400	15	7.5×10^8	4.8×10^8 (-36%)
K400	30	1.5×10^9	8.2×10^8 (-45%)
SSv2	7.5	2.6×10^8	1.8×10^8 (-31%)
SSv2	15	5.2×10^8	3.2×10^8 (-38%)
SSv2	30	1.0×10^9	5.7×10^8 (-48%)
EK-100	3.5	1.1×10^8	7.2×10^7 (-36%)
COIN	30	9.8×10^9	2.8×10^9 (-71%)
Breakfast	15	1.3×10^9	2.7×10^8 (-79%)

Table 4: **Per-Dataset Token Reduction.** RLT reduces tokens significantly across datasets, with higher reductions on higher FPS. On long-video datasets like COIN and Breakfast with mostly static content, RLT achieves almost 80% reduction, demonstrating its promise for scaling training.

Model	FPS	Acc	FT Time	
ViT-L	7.5	84.8	21.6h	
RLT	**7.5**	**84.6**	**15.4h**	**1.41×**
ViT-L	15	85.8	45.2h	
RLT	**15**	**85.8**	**27.4h**	**1.72×**
ViT-L	30	86.3	110h	
RLT	**30**	**86.2**	**52.3h**	**2.1×**
ViT-L	7.5	74.3	15.1h	
RLT	**7.5**	**74.4**	**10.8h**	**1.39×**
ViT-L	15	75.4	41.4h	
RLT	**15**	**75.3**	**24.1h**	**1.7×**
ViT-L	30	76.1	99.8h	
RLT	**30**	**76.1**	**47.5h**	**2.0×**

Table 5: **Training at higher FPS.** RLT enables training efficiently for higher FPS, allowing us to go beyond the standard low FPS paradigm. As FPS increases, RLT delivers larger and larger speed-ups over the baseline for training, with no decrease in accuracy.

Difference Threshold. The only tunable hyperparameter in RLT is the threshold τ, which controls the sensitivity to change between temporally consecutive tokens. Lower values of τ indicate higher sensitivty to change. We vary tau and compare the final action recognition accuracy vs. throughput and wall-clock time for several configurations, both for training and inference. These results are shown in Figure 3. We find that using $\tau = 0.1$ offered the best tradeoff in speed and performance: it matches the baseline performance while delivering a 37% speedup in training. Lower values of τ lead to similar performance, but with less of a speedup, while high values deliver larger speedups at a cost to performance. We attribute this to the existence of a 'difference cut-off': at some point, the tokens are too different to be grouped together, and the resulting tokens do not obey the assumptions made by RLT. We also note that τ is *dataset-agnostic*: it simply describes how much pixel difference is needed to consider two 16x16 patches different, and the same value of τ leads to different reductions across datasets based on the video content.

Length Encoding. We ablate the effect of our length encoding mechanism in Table 3. When using RLT by itself, length encoding has minimal effect. However, when combining RLT with random masking, we note a clear improvement. Due RLT's structured and predictable pruning, length encoding may be unnecessary: the transformer is able to mostly understand the length of various tokens by their associated spatial positional encoding. However, once random masking is introduced, the structure is removed, and the length encoding adds crucial information. Since including the length encoding is strictly more information and has no negative effect, we default to including it.

4.4 Longer Videos and Higher FPS

Standard action recognition datasets consist of short clips with downsampled FPS; an input example typically spans 2 seconds. One potential advantage of RLT is that by reducing the total number of tokens, training becomes more tractable for both longer videos and higher FPS. We evaluate the effect of training with RLT in Table 5 on action recognition datasets with higher FPS along with their training time. As before, we fine-tune these models from pre-trained VideoMAE checkpoints. Although these checkpoints were pre-trained at 7.5 FPS, we can still compare with the baseline performance to observe differences in training time or quality. Similar to the result from Table 1, we find that ViTs trained with RLT can match performance but train significantly faster, with the speed-up increasing with the FPS.

We next analyze the number of total tokens in RLT compared to the baseline for several video datasets in Table 4, including datasets with longer videos as well as higher FPS. Matching the result from Table 5, at higher FPS, RLT consistently reduces the tokens by a higher proportion. This matches

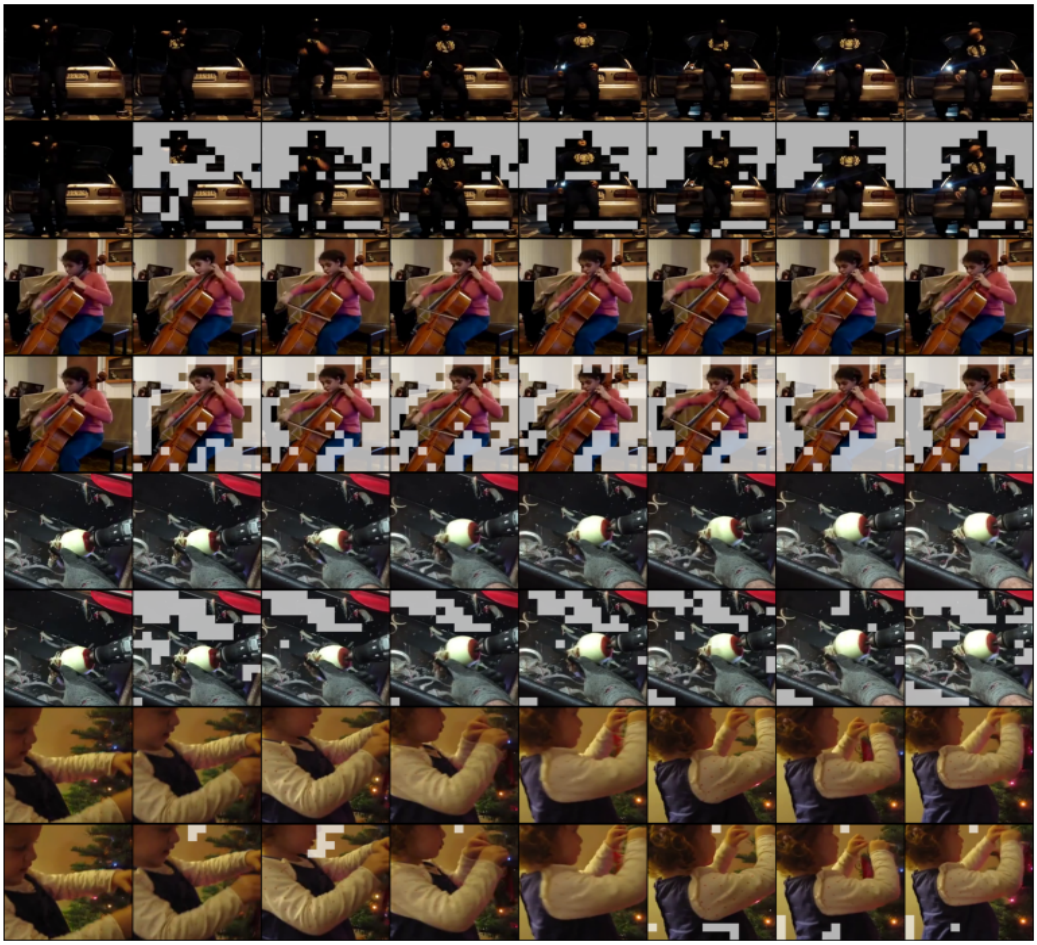

Figure 4: **Sample Visualizations.** Tokens that are compressed are visualized in gray. RLT retains tokens that change between frames while removing redundant tokens. In the top example, RLT captures the static background, and in the bottom example, due to camera motion and the motion of the girl, almost no tokens are modified. Video visualizations are available at the project page.

our intuition, since tokens between two redundant tokens at lower FPS are likely to be similar and also be removed. Furthermore, on longer video datasets, RLT can reduce the number of tokens by significantly larger margins, with reductions of up to 80% on COIN and Breakfast. These datasets in particular consist of videos filmed with fixed cameras and largely static backgrounds, demonstrating RLT's potential to drastically speed up transformers on these types of videos. Although in practice, researchers do not typically train on raw videos with large number of frames due to the heavy cost of video decoding on academic clusters, RLT presents a promising way to efficiently train on these videos at scale.

4.5 Visualizations

We provide some qualitative visualizations of the tokens RLT removes in Figure 4. As desired, input patches that are repeated over time are pruned by RLT. This intuitively matches with how humans often pay less attention to static tokens over time. In the top example, most of the background is black, with some motion taking place in the foreground. RLT is able to remove the constant black portions, drastically reducing the number of tokens. Similarly in the second example, RLT ensures that the tokens containing motion, with the boy's hands and instrument, are not modified, but prunes the static background. In the lower two examples, the person using the drill and the girl in the foreground move around significantly, reducing the amount of tokens that can be compressed. In such cases where there is significant subject or camera motion, RLT removes fewer tokens, resulting in similar token

https://doi.org/10.52202/079017-0882

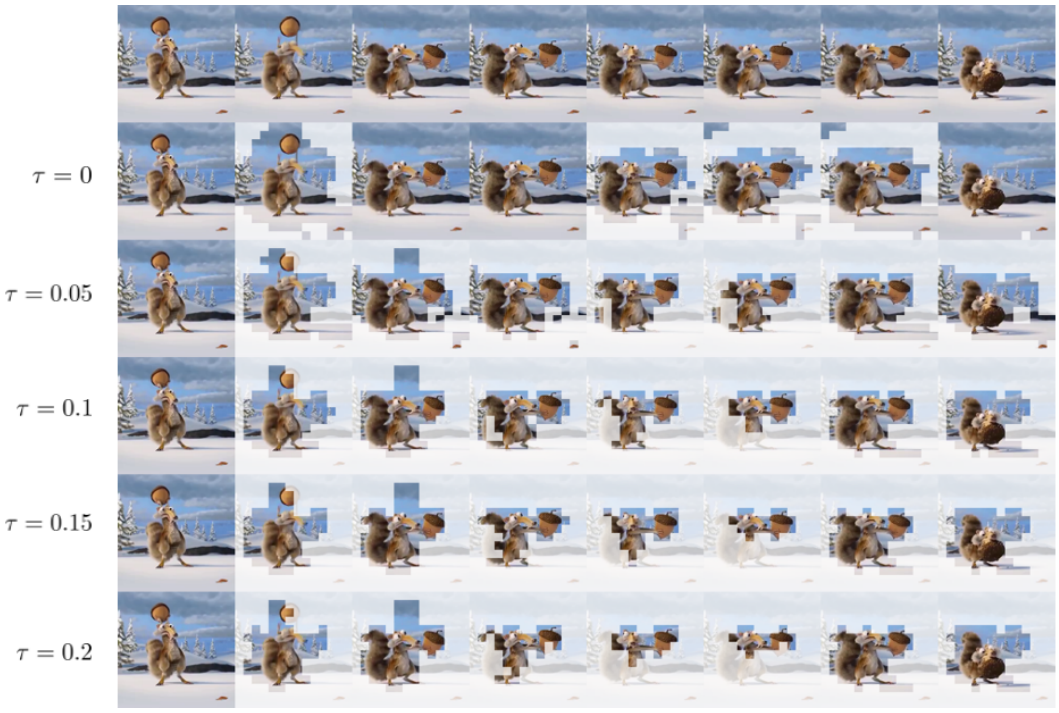

Figure 5: **Effect of τ.** With low values of τ, the clearest repeated patches are ablated, but imperceptible variations can prevent some visibly similar tokens from being pruned. Above $\tau = 0.1$, some tokens with slight movement are pruned.

counts to standard tokenization. However, the sensitivity of RLT to small perturbations and motion depends entirely on the τ hyperparameter. We provide further example visualizations and visualize the effect of different values of τ in Appendix B and on our project page. In Figure 5 we demonstrate the effect that the τ hyperparameter has on the input tokens. We see that as τ increases, more and more patches are included, and after $\tau = 0.1$, some patches that have change in them are pruned incorrectly. On the other hand, $\tau = 0$ includes many patches with essentially imperceptible change, which is also undesired.

5 Conclusion

Summary We present Run-Length Tokenization (RLT), a simple alternative to standard video tokenization for video transformers that replaces temporally redundant tokens with a single token of variable length. RLT decreases transformer training and inference wall-clock time by up to 40%m achieves a better speed-accuracy tradeoff than prior works, and is simple to implement and combine with other methods. RLT demonstrates strong results during finetuning, especially at higher FPS, and even works well when applied to models without any training.

Limitations Though RLT works well, it relies on a heuristic to compare temporally consecutive tokens, which could include extra tokens that are unused by the transformer. While RLT speeds up video transformers significantly, it cannot be used for dense vision tasks, such as point tracking or video generation, that require the same number of output tokens as input tokens; RLT reduces tokens before running the model and does not replace them. Furthermore, RLT does not handle camera motion well: in a video with constant camera motion, few tokens will be removed, leading to no speedup. Future work will be necessary to overcome these limitations, and we hope that RLT can inspire more research on efficient video transformers.

Acknowledgments and Disclosure of Funding

This work is supported by Fujitsu Research of America, and RC is supported by the NSF GRFP.

References

[1] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *Advances in Neural Information Processing Systems*, 34:24206–24221, 2021.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[4] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14496–14506, 2023.

[5] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

[6] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[8] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[10] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[13] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021.

[14] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019.

[15] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv:2205.09113*, 2022.

https://doi.org/10.52202/079017-0882

[16] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[18] Sunil Hwang, Jaehong Yoon, Youngwan Lee, and Sung Ju Hwang. Everest: Efficient masked video autoencoder by removing redundant spatiotemporal tokens.

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

[20] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European conference on computer vision*, pages 620–640. Springer, 2022.

[22] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6232–6242, 2019.

[23] Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2021.

[24] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014.

[25] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

[26] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[27] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. Scaling language-image pre-training via masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23390–23400, 2023.

[28] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022.

[29] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019.

[30] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[31] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. *arXiv preprint arXiv:2110.03860*, 2021.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[33] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[34] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

[35] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[36] Yong Ren, Chenxing Li, Manjie Xu, Wei Liang, Yu Gu, Rilin Chen, and Dong Yu. Sta-v2a: Video-to-audio generation with semantic and temporal alignment. *arXiv preprint arXiv:2409.08601*, 2024.

[37] Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*, 2022.

[38] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *International Conference on Machine Learning*, pages 29441–29454. PMLR, 2023.

[39] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

[40] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[41] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[42] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019.

[43] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

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

[45] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14549–14560, 2023.

[46] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

 https://doi.org/10.52202/079017-0882

[47] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[48] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6026–6035, 2018.

[49] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13587–13597, 2022.

[50] Zhirong Wu, Zihang Lai, Xiao Sun, and Stephen Lin. Extreme masking for learning instance and distributed visual representations. *arXiv preprint arXiv:2206.04667*, 2022.

[51] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18134–18144, 2022.

[52] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10818, 2022.

[53] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3008, 2021.

[54] Yue Zhao and Philipp Krähenbühl. Training a large video model on a single machine in a day. *arXiv preprint arXiv:2309.16669*, 2023.

[55] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 803–818, 2018.

[56] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 695–712, 2018.

A Implementation Details

Our code, demos and associated blog post are all located on our project page. In this section, we provide further details on implementation details of our experiments.

Architecture. All models used were based on the `timm` [47] Vision Transformer implementation, and all fine-tuning experiments were done with pre-trained checkpoints from VideoMAE [43] and VideoMAEv2 [45]. As mentioned in 3.3, we compute output predictions for action recognition by taking the mean across the output tokens, rather than producing a separate class token.

Baselines. The baselines we compared to are Token Merging [5] and random masking [27]. For all random masking experiments, we set the masking ratio ρ to match the mean RLT token reduction for the given dataset. For example, on Kinetics-400 at 7.5 FPS, RLT with $\tau = 0.1$ reduces the number of tokens by 28%, so we randomly drop 28% of the tokens during training. We use the recommended values of r from the Token Merging paper, except on ViT-H, where we use $r = 32$ due to the larger depth of the model.

Datasets. We train and evaluate RLT on Kinetics-400 (K400) [19] and Something-Something-v2 (SSv2) [16]. Both datasets are video classification datasets, with K400 having 400 classes and SSv2 having 174. K400 has 240k training examples and 40k test examples, while SSv2 has 170k training examples and 30k test examples. We also included experiments measuring the token reduction on the Breakfast [24] and COIN [42] datasets, both of which are smaller-scale datasets involving longer videos that range from 2-5 minutes. In particular, these datasets contain lots of fixed-camera videos with static backgrounds, leading to particularly high token reductions from RLT.

Training Recipe. We do not change hyperparameters when finetuning models with different tokenization strategies, as we found the provided set to be optimal in our experiments. We follow the recommended training recipes from VideoMAE for each model size, namely training for up to 100 epochs, with batch size 256, learning rate with warm-up to 1×10^{-3} for 5 epochs, then cosine annealing down to 1×10^{-6}. We also use RandAugment, random erasing, CutMix, and standard cropping/scaling and flipping. We do not use MixUp since it can severely affect the efficacy of RLT, and we found that removing it and only using CutMix did not affect our experiments. We also used random erasing with a single value rather than noise, enabling some of the erased tokens to be removed by RLT.

All experiments were conducted with 8xH100 Nvidia GPUs with 128 CPU cores, with 16 workers per GPU. The inference-time results were computed on a single GPU, along with the throughput and FLOPS analysis. One important detail is that data loading is often a bottleneck. We mainly relied on the fast video data loader from AVION [54], but NVIDIA DALI also works very well. However, we only recomend to use DALI on A100 or newer chips, as earlier generations have an insufficient number of dedicated decoder hardware. Each training run for the paper is specified in hours, but this does not include a few months of work testing and debugging. We used a single node for all work on this paper.

B More Visualizations

We include some additional visualizations here to qualitatively demonstrate which tokens RLT prunes, as well as to analyze the qualitative effect of varying the difference threshold τ. In each figure, the whitened patches represent those RLT identified as static, and that are not passed to the transformer. In Figure 6, we visualize a diverse range of samples and note that RLT consistently prunes out patches that repeat across consecutive frames. One case where RLT fails to remove many tokens is the 4th example from the top, which is from a ski jumper using a GoPro; the constant camera motion means that RLT is unable to identify almost any repeated patches.

We highly encourage readers to visit our project page for video visualizations that better convey the effect of RLT.

https://doi.org/10.52202/079017-0882

Figure 6: **More examples.** We visualize RLT's effect on videos ranging from TV shows, movies, action sequences on a GoPro, and sports. RLT consistently prunes out tokens that are repeated and static, and includes all patches that change between frames, retaining as much information as possible while cutting the number of tokens significantly.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claims in the abstract and introduction are concretely linked to results in our Results section, and aspirational goals are clearly denoted.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitations section at the end of the paper and discuss some of the weaknesses of our method that we hope future work can address.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-0882

Answer: [NA]

Justification: Our paper includes no theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our paper includes detailed technical instructions, lists of hyperparameters and model architectures and experimental results. We also include our code in this submission and will open-source the code upon releasing the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include the code in the submission which has all the necessary files to run the experiments from the paper. We have no dataset contribution and thus do not include any data.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We include data splits, hyperparameters and rational for choosing them in the Implementation details section of the appendix. This is also provided with the code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: Our experiments are deterministic, and none of the experiments require statistical significance analysis; we ensure all experiments are random-seeded and reproducible.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-0882

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include details about our compute usage in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We reviewed and adhere to the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The impact of our work is simply to speed up existing methods, in this case vision transformers. We do not see any clear way for this to be maliciously used or for this to directly negatively impact society and thus did not discuss this in the main text.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: Our experiments are relatively small-scale and based on existing pre-trained models, and thus requires no such safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All pre-trained models were cited for their authors, along with the codebases used. We reference these in the main text, supplement and code.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

https://doi.org/10.52202/079017-0882

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide details about our method in the main text and supplement, and the code contains documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We include no human subject research in our work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We have no user studies and thus did not need an IRB.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0882