ResidualViT for Efficient Temporally Dense Video Encoding

Supplementary Material

We provide the following additional information:

1. **Token Dropping Strategies:** We present the different token dropping strategies that can be adopted in the token reduction module in Section 1.
2. **Motion-based Token-dropping Strategy:** Insights into the motion-based token-dropping strategy are provided in Section 2, explaining the additional RAM requirements and the pre-processing of raw motion vectors.
3. **Zero-shot Grounding Algorithm:** A thorough explanation of the implementation details of the zero-shot temporal grounding algorithm is presented in Section 3.
4. **Additional Comparison for Short-Form NLTVG:** Additional analysis for the task of natural language temporal video grounding on the Charades-STA and ActivityNet-Caption datasets are available in Section 4.
5. **Feature Comparison under Full Supervision Setup:** As an additional test of the quality of our ResidualViT features, we investigate the accuracy of CG-DETR [34] when replacing the original CLIP features with our ResidualViT ones in Section 5.
6. **Additional Comparison for Long-Form NLTVG:** Additional analysis for the task of natural language temporal video grounding on the MAD dataset is available in Section 6.
7. **Additional Comparison for AD:** Additional results for the task of automatic audio descriptions on the MAD dataset are available in Section 7.
8. **Additional Comparison for TAL:** Additional results for the task of temporal action localization on the ActivityNet-v1.3 dataset are available in Section 8.
9. **Supplementary Ablations:** In Section 9, we conduct additional ablations for ResidualViT on the Charades-STA dataset, exploring different token reductions strategies as presented in Section 1 and discussing the role of token-dropping probability. Additionally, we investigate two distinct strategies for computational savings: token merging and reduction of the spatial resolution of the input frames. We ablated the design of the distillation approach and showcased how different distillation objectives can achieve competitive accuracy. Moreover, we ablate the interleave factor during distillation training. Finally, we report two additional ablations on the MAD dataset, investigating the main components and the interleave factor N.
10. **Video Encoding Latency:** Section 10 empirically validates the wall-clock timings of ResidualViT, demonstrating significant time savings compared to a standard ViT model, despite requiring two forward passes.
11. **Additional task - Action Recognition:** In this supplementary experiment, we test the accuracy of CLIP features against ResidualViT features on the task of action recognition on the Kinetics 400 dataset. Results are reported in Section 11.
12. **Limitations and Discussion:** In Section 12 we discuss the inherent limitations of our solution.
13. **Qualitative Results:** We conclude with a showcase of several qualitative results in Section 13, highlighting the practical effectiveness of our approach.

1. Token Dropping Strategies

In Section 3.1 we introduced the ResidualViT architecture, which consists of the token reduction module ($\mathcal{R}$), the residual tokenizer ($\mathcal{A}$), and the transformer encoder ($\mathcal{E}_{\mathcal{V}}$). Here, we explore four practical implementations of the token reduction module when adopting the token dropping strategy [6, 13, 16, 31].

For a given frame x_t, which is transformed into a set of tokens $\mathcal{T}$, each strategy retains $(1-p) \times |\mathcal{T}|$ tokens, where p is the token reduction probability. Figure 1 visually depicts the four token reduction approaches we investigate. (i) The

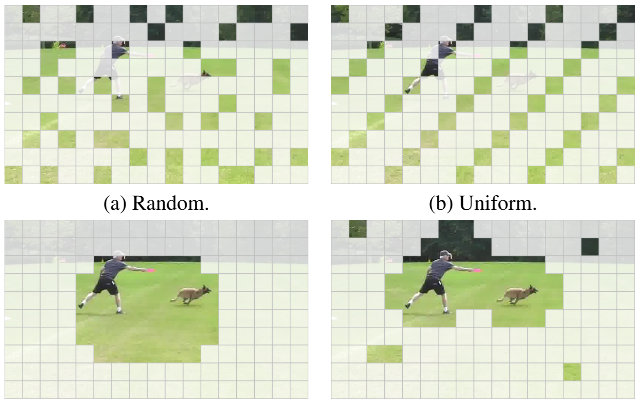

(a) Random. (b) Uniform.
(c) Center. (d) Motion.

Figure 1. **Token reduction strategies.** We implement three data-independent token reduction strategies (a-c) and one data-dependent one (d).

random strategy randomly samples tokens from the set. Conversely, (ii) the **uniform** strategy selects tokens from patches that are evenly distributed across the 2D grid of image patches, ensuring that the selected patches are spaced at regular intervals throughout the frame. (iii) The **center** strategy is designed to retain tokens of patches from the center of the frame. This strategy takes into consideration that, when shooting a video, we tend to center the frame around the subject or action being recorded. Finally, we design a data-dependent (iv) **motion** strategy. This strategy further exploits the characteristics of video data, which describes how characters, objects, and scenes evolve in time. We argue that motion is a valuable source of information readily accessible from encoded video files, providing information on which parts of the frame at time step $t + k$ differ from the frame at time step t. Consequently, we discard tokens representing patches with minimal motion, assuming their change relative to previous frames is negligible, and their information can be recovered through the residual token. Notably, by prioritizing tokens associated with regions of higher motion, ResidualViT is well-suited to handle fast-moving content. See Section 2 for additional details about motion preprocessing and memory overhead.

2. Implementation Details for Motion-Based Token Reduction Strategy

Motion is a valuable and readily available source of information for determining which spatial regions of a frame have changed with respect to the previous one. To harness this information, our method employs a compressed video reader [1] that extracts motion vectors directly from compressed video streams. Nevertheless, it is important to acknowledge that motion vectors extracted from raw video data typically exhibit a moderate level of noise, attributable to the inherent sparsity and optimization mechanisms of standard video compression techniques. To counteract this effect and derive a more reliable motion estimation, we compute the average motion across a short temporal window surrounding a target frame x_t. Specifically, we construct a set of motion vectors $M_v = \{m_i\}_{i=t-W_M/2}^{t+W_M/2}$, where each vector $m_i \in \mathbb{R}^{H' \times W' \times C'}$ corresponds to the motion information of frame i. Here, $H' = H/4$ and $W' = W/4$ are the reduced height and width dimensions, respectively, and $C' = 4$ signifies the channels in the motion vector, capturing the $(\Delta x, \Delta y)$ displacement of pixels with respect to adjacent frames (previous and following ones). The parameter W_M denotes the size of the temporal window over which the motion is aggregated. As we are interested in the magnitude of the motion and not its direction, we compute the average L_1 norm along dimension C' in the window W_M. Note that, at the start of the video ($t < W_M$) and at the end ($t > T - W_M$, where T is the timestamp of the last frame), the window is reduced so that only the available motion vectors are aggregated, avoiding the need for padding.

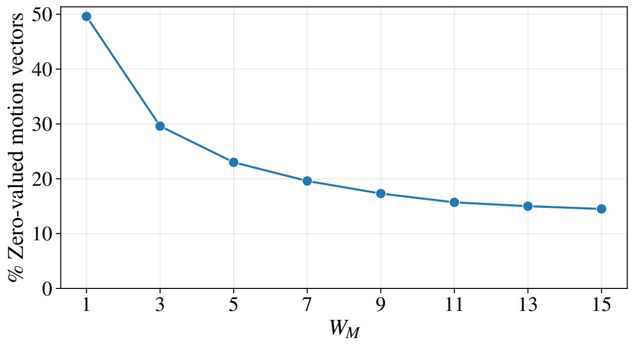


Figure 2. **Optimal W_M hyperparameter value.** The plot shows the average percentage of zero-valued motion vectors on the Charades-STA dataset as the aggregation window size W_M varies. The trend flattens beyond $W_M = 11$, indicating diminishing returns. Therefore, we choose $W_M = 11$ as our default parameter.

We then upsample the computed motion magnitudes to the frame resolution (H, W) and select the $1 - p$ frame tokens with the highest motion magnitudes at their patch's spatial location.

In our implementation, we set the motion window size $W_M = 11$. This setting implies that we incur an additional RAM memory consumption that is proportional to the cost of storing a frame in memory. We can estimate the memory cost as follows. The memory consumption of a frame can be expressed as $M_F{=}H{\times}W{\times}3$, while for motion vectors $M_{Mv}{=}(H/4){\times}(W/4){\times}4{\times}M_v$, resulting in a total memory cost $M_F + M_{Mv}/M_F = {\sim}1.9{\times}$. Note that this memory overhead does not affect GPU memory availability as the motion vectors are not required to be moved to such a device for processing. To determine the value of W_M, we measure the average percentage of zero-value motion vectors in the Charades-STA dataset. As shown in Figure 2, we find that for $W_M = 1$, roughly 50% of motion vectors are zero while considering $W_M = 11$ reduces this value to less than 15%. We do not observe a significant reduction beyond $W_M = 11$. For simplicity, we keep this parameter constant across datasets.

Finally, note that we utilize motion information to identify frame patches that have likely undergone significant transformations relative to preceding frames. This strategy enables us to provide the transformer encoder of ResidualViT ($\mathcal{E}_S$) with patches expected to exhibit less redundancy with previous frames. Importantly, this approach does not supply the encoder with motion information, meaning the network remains unaware of explicit motion patterns.

3. Zero-shot Grounding Algorithm

The task of natural language video grounding involves temporally localizing a natural language description within a single video. Given the fine-grained temporal localization requirements of the task, dense frame sampling and encoding are indispensable, making it an ideal testbed for our efficient ResidualViT approach. This section details feature encoding and describes the motivations for addressing the task in a zero-shot setting.

We argue that the zero-shot setting holds valuable properties. Firstly, algorithms evaluated in a zero-shot manner are not prone to be affected by the inherent biases of the downstream datasets, which have shown to be a danger for this task [38, 45, 56]. Additionally, models exhibiting strong zero-shot capabilities typically demonstrate enhanced generalization to unseen datasets, thereby increasing their versatility and utility. Secondly, from a practical standpoint, relying on multiple specialized models for each new dataset can severely limit the scalability and versatility of systems. In contrast, a unified model that excels in zero-shot settings streamlines system architecture and boosts scalability and adaptability. Such models simplify the maintenance and deployment of deep learning applications and readily adjust to new challenges without the need for extensive retraining. Third, the zero-shot approach promotes environmental sustainability. This approach significantly curtails the computational demands by drastically reducing the necessity for ongoing retraining on possibly extensive datasets, thus lowering energy consumption and the associated carbon footprint. Employing large pre-trained models in a zero-shot manner optimizes their efficacy while minimizing further environmental impacts. We strive to pursue zero-shot evaluation in this work for all these reasons.

Visual Encoding. Our algorithm begins with encoding a set of video frames $\mathcal{X} = \{x_t\}_{t=1}^{n_v}$ through a designated visual encoder (either a standard ViT or our ResidualViT). This process generates a series of frame features $\{f_t\}_{t=1}^{n_v}$. When employing ResidualViT, in line with the approach illustrated in Figure 2a, we utilize a sliding window mechanism that concurrently processes $N+1$ frames. The first frame in each window is encoded by the foundation model encoder $\mathcal{E}_{\mathcal{V}}$, with the resulting features stored for subsequent use. The following N frames are processed by encoder $\mathcal{E}_{\mathcal{S}}$ (Figure 2b), which takes as input the frame tokens and the cached feature of the first frame of the window. The residual feature is first transformed into the residual token via the residual tokenizer. Subsequently, in the reduction module, frame tokens are reduced according to a particular strategy and token reduction probability p. Finally, these sparse visual tokens are concatenated with the residual token and forwarded to the visual encoder $\mathcal{E}_{\mathcal{V}}$.

Language Encoding. The language encoder is kept frozen throughout our experiments and initialized with CLIP weights corresponding to the specific version of the visual encoder (ViT-B/32, ViT-B/16, or ViT-L/14). To solve the task, each sentence s is first tokenized and then processed through the language encoder to derive a single sentence feature g_l.

Grounding Algorithm. For the grounding task, cosine similarity between each frame embedding and the sentence embedding is calculated, creating a temporal sequence of similarity scores $\{S_t\}_{t=1}^{n_v}$. We post-process the similarity profiles with a moving average smoothing operation with window size W_{MA}.

Finally, inspired by methods in prior work such as [7, 24], we implement a watershed algorithm [42] for moment prediction. In this step, group consecutive timesteps where the similarity scores exceed a given threshold, effectively delineating temporally contiguous segments. The start and end timesteps of these segments constitute our moment predictions. Multiple predictions are sorted based on the highest frame-sentence similarity in their span.

For short-video datasets, such as Charades-STA and ActivityNet-Captions, we compute the threshold as a scaled average of the scores, given by $\frac{\alpha}{n_v}\sum_{t=1}^{n_v} S_t$, where α is a scaling factor. Conversely, for the long-form MAD dataset, we normalize the scores to the range $[0, 1]$ and apply a fixed threshold β, an approach that mitigates the influence of low-relevance similarities in longer sequences that can otherwise skew the average similarity score. Section 13 presents several qualitative results showcasing the aforementioned similarity profile.

4. Additional Short-Form NLTVG Comparisons

For completeness, Table 1 provides a summary comparing state-of-the-art grounding methods, categorized into fully supervised [2, 8, 27, 30, 36, 44, 53, 54, 57], weakly supervised [5, 19, 58], pseudo-supervised [7, 10, 15, 22, 37, 50, 59], and zero-shot techniques [29, 33] within the context of short video setups.

In fully supervised settings, models are trained using video data, corresponding sentences, and temporal boundaries. In contrast, weakly supervised approaches eliminate the need for temporal annotations. Our work is more closely aligned with settings where textual or temporal labels are unavailable. In these scenarios, prior approaches have leveraged off-the-shelf concept detectors (e.g., for objects, actions, and scenes) [10, 37, 50], which are used to automatically generate pseudo-annotations (sentences and temporal boundaries) for downstream tasks. These pseudo-annotations are then used to train grounding models. Other sources of pseudo-supervision include pretrained visual-language embeddings [22], commonsense knowledge [15, 46], and captioning methods [59]. Furthermore, methods

	Supervision	Use Downstream Task Data	Charades-STA R@1 ↑		Avg. Cost Feature/sec ↓ (GFLOPs)	ActivityNet-Captions R@1 ↑		Avg. Cost Feature/sec ↓ (GFLOPs)
			IoU=0.5	IoU=0.7		IoU=0.5	IoU=0.7	
2D-TAN [57]	Full	✓	39.8	23.3	**74.2**	44.0	27.4	**19.3**
CPNet [27]	Full	✓	$\underline{60.3}$	$\underline{38.7}$	638.3	40.6	21.6	$\underline{38.5}$
CRaNet [47]	Full	✓	**60.9**	**41.3**	$\underline{296.8}$	**47.3**	**30.3**	**19.3**
WSTG [5]	Weak	✓	27.3	12.9	**38.5**	23.6	–	$\underline{38.5}$
CRM [19]	Weak	✓	34.8	16.4	638.3	32.2	–	**23.2**
CPL [58]	Weak	✓	**49.2**	**22.4**	$\underline{445.2}$	**31.4**	–	115.5
U-VMR [10]	Pseudo	✓	20.1	8.3	$\underline{289.5}$	26.4	11.6	962.5
PSVL [37]	Pseudo	✓	31.3	14.2	638.1	30.1	14.7	$\underline{38.5}$
PZVMR [50]	Pseudo	✓	33.2	18.5	638.1	$\underline{31.3}$	**17.8**	$\underline{38.5}$
CORONET [15]	Pseudo	✓	34.6	17.9	638.1	28.2	12.8	$\underline{38.5}$
LFVL [22]	Pseudo	✓	$\underline{37.2}$	$\underline{19.3}$	638.1	**32.6**	$\underline{15.4}$	$\underline{38.5}$
SPL [59]	Pseudo	✓	**40.7**	**19.6**	**166.5**	27.2	15.0	83.3
UniVTG [29]	Zero-Shot	✗	25.2	10.0	70.0	–	–	–
MR-FVLM [33]	Zero-Shot	✗	**42.9**	20.1	1370.0	27.9	11.6	370.0
CLIP (B/32)	Zero-Shot	✗	35.9	18.7	$\underline{13.2}$	27.8	**13.9**	$\underline{4.4}$
ResidualViT (B/32)	Zero-Shot	✗	34.2	17.7	**6.1**$_{(-53\%)}$	27.3	13.7	**2.0**$_{(-53\%)}$
CLIP (B/16)	Zero-Shot	✗	37.7	21.2	50.7	28.1	$\underline{13.8}$	16.9
ResidualViT (B/16)	Zero-Shot	✗	37.8	21.0	22.4$_{(-56\%)}$	27.5	$\underline{13.8}$	7.5$_{(-56\%)}$
CLIP (L/14)	Zero-Shot	✗	**42.9**	**24.1**	233.4	**29.1**	$\underline{13.8}$	77.8
ResidualViT (L/14)	Zero-Shot	✗	$\underline{41.5}$	$\underline{23.8}$	102.6$_{(-56\%)}$	$\underline{28.3}$	13.5	34.2$_{(-56\%)}$

Table 1. **Short video state-of-the-art comparison.** We compare our approach against state-of-the-art methods using different levels of supervision. Our ResidualViT reduces the cost of frame encoding by 56% while closely retaining the performance of the CLIP model. The best method in each block of directly comparable methods is bolded, and the second-best method is underlined.

employing complex proposal schemes, such as feature clustering [15, 22, 37] or sliding windows [50], are often paired with strategies for supervised feature refinement. Although these methods do not rely on manually annotated labels, they still adapt model parameters using the training dataset for the target downstream task, which is why we categorize them as *pseudo-supervised*.

In contrast, our zero-shot approach (described in Section 3) can be directly compared to zero-shot methods such as UniVTG [29] and MR-FVLM [33], all of which avoid training on task-specific datasets.

For each method in Table 1, we report the grounding accuracy on the Charades-STA [11] and ActivityNet-Captions [23] datasets, alongside the average embedding cost per second. Previous methods have used visual backbones such as ResNet152, C3D, BLIP, VGG-19, and I3D [4, 14, 26, 43, 49], with respective costs of 11.6, 38.5, 55.5, 143.7, and 148.4 GFLOPs per feature.

Table 1 also reports the grounding accuracy using the vanilla CLIP and our ResidualViT features across different backbones. For ResidualViT, we employ motion-based token reduction with a probability of $p = 85\%$ and set the interleave parameter to $N = 2$. For the grounding algorithm, we set W_{MA}=15 and α=1.0 for Charades-STA, W_{MA}=15 and α=0.95 for ActivityNet-Captions.

As highlighted in the main paper, our ResidualViT closely matches CLIP's grounding accuracy while reducing frame encoding costs by approximately 56% across all ViT backbones. Despite not being trained on the downstream task data, our method still achieves competitive accuracy when compared to prior approaches that train on both datasets. Specifically, for the Charades-STA dataset, our approach offers the best cost vs. accuracy trade-off among all pseudo-supervised methods. For the ActivityNet-Captions dataset, our method with the B/16 backbone matches or surpasses the accuracy of three pseudo-supervised methods, while maintaining a lower computational cost.

5. Feature Comparison under Full Supervision Setup

In this section, we focus on a representative fully supervised baseline for Natural Language Temporal Video Grounding to evaluate the accuracy gap between CLIP and ResidualViT features. For this experiment, we selected CG-DETR [34], a recent and well-performing publicly available baseline that natively utilizes CLIP features for the Charades-STA dataset. The results of our experiments are

	Features	Charades-STA R@1 ↑ IoU=0.3	R@1 ↑ IoU=0.5	R@1 ↑ IoU=0.7	mIoU ↑	Avg. Cost Feature/sec ↓ (GFLOPs)
CG-DETR	CLIP (B/32)	63.6	49.7	26.8	43.8	4.4
CG-DETR	ResidualViT (B/32)	62.2	48.2	26.4	42.5	$2.0_{(-53\%)}$
CG-DETR	CLIP (B/32) + SlowFast	69.6	57.1	34.5	49.0	40.5
CG-DETR	ResidualViT (B/32) + SlowFast	69.2	56.5	34.0	48.7	38.1
CG-DETR*	CLIP (B/32) + SlowFast	70.4	58.4	36.3	50.1	40.5

Table 2. **Frame feature comparisons in full supervision setup.** This table compares the performance of the baseline CG-DETR [34] on the Charades-STA dataset under two setups: (i) using either CLIP (B/32) or ResidualViT (B/32) alone, and (ii) combining SlowFast features with either CLIP (B/32) (as in the original manuscript [34]) or ResidualViT (B/32). Our ResidualViT achieves a 53% reduction in frame encoding cost while closely maintaining the accuracy of the original setup. We denote with the symbol $*$ the accuracy as presented in the original paper (last row). Note that all other rows have been trained from scratch using the original codebase.

presented in Table 2, and we maintained all hyperparameters as defined by the official implementation. Notably, features were extracted at a rate of one frame per second. For all rows except the last one, we train CG-DETR from scratch. The last row reports the accuracy as presented in the original paper. We find that we cannot fully reproduce those results using the default settings.

We begin by comparing the accuracy when using only CLIP features versus ResidualViT features, as shown in the first two rows of the table. For ResidualViT, we set N=2 and p=85%. ResidualViT achieves a reduction in encoding cost of approximately 53% while maintaining accuracy close to the CLIP features. Specifically, we observe a marginal drop of 1.4% (relative 2.2%) for R@1-IoU=0.3, an absolute drop of 1.5% (relative 3.0%) for R@1-IoU=0.5, and an absolute drop of 0.4% (relative 1.5%) for R@1-IoU=0.7. These results indicate that, with an average relative accuracy drop of only 2.2%, we can achieve more than a 50% reduction in encoding cost.

Additionally, we evaluated the accuracy of CG-DETR in its original configuration, where CLIP features are channel-wise combined with SlowFast [9] features. This setup significantly increases computational cost, as SlowFast features alone are estimated at 36.1 GFLOPs per feature. While the addition of SlowFast features can boost average accuracy on average of approximately 7.0%, it comes with a 9.2$\times$ increase in computational cost, representing an unfavorable trade-off. Nonetheless, when SlowFast features are combined with ResidualViT features, the computational cost is reduced by approximately 6%, with only a 0.5% absolute drop (relative 1%) in average accuracy, providing once again a favorable balance between accuracy and cost reduction.

6. Additional Long-Form NLTVG Comparisons

In this section, we present additional grounding results for the long-form MAD dataset. Table 3 builds on Table 3 from the main paper by incorporating results from supervised state-of-the-art methods and zero-shot watershed accuracy using CLIP features.

We begin by emphasizing that our zero-shot watershed-based grounding algorithm, detailed in Section 3, significantly outperforms the proposal-based method introduced by [45]. By comparing rows 9 and 10 of the table, where both algorithms utilize the same visual backbone (CLIP ViT-B/32), we isolate and evaluate their individual contributions. Our zero-shot watershed-based approach demonstrates superior accuracy, with relative improvements ranging from 43% to 128%. Remarkably, our zero-shot results are comparable with, or even surpass, several fully supervised methods listed in rows 1 through 8.

Table 3 also enables a direct comparison of different backbone features while keeping the grounding algorithm fixed, thereby contrasting CLIP with our ResidualViT. For ResidualViT, we utilize configurations of $N = 2$, $p = 85\%$, and a center token dropping strategy, resulting in an embedding cost reduction of 53% to 56%. For the grounding algorithm, we set W_{MA}=7 and β=0.7.

When using the ViT-B/32 backbone (rows 10-11), ResidualViT reduces computational costs by approximately 53%, with an average accuracy degradation of just 0.1% compared to CLIP weights, a negligible decrease. Similarly, employing the ViT-B/16 backbone (rows 12-13) ResidualViT achieves a 56% reduction in computation with respect to CLIP, accompanied by an average accuracy drop of 0.5%. For the larger ViT-L/14 backbone, the average accuracy drop is 1.5%, with the most significant decrease occurring for the less stringent metric (R@1 IoU=0.1). We hypothesize that the slightly larger accuracy drop observed on

Grounding Algorithm	Use Downstream Task Data	Features	Visual Backbone	MAD						Avg. Cost Feature/sec ↓ (GFLOPs)
				R@1 ↑			R@5 ↑			
				IoU=0.1	IoU=0.3	IoU=0.5	IoU=0.1	IoU=0.3	IoU=0.5	
DenoiseLoc [53]	✓	CLIP	ViT-B/32	1.1	0.9	0.5	4.1	3.3	2.2	21.8
2D-TAN [57]	✓	CLIP	ViT-B/32	3.2	2.5	1.6	11.9	9.3	5.7	21.8
Moment-DETR [25]	✓	CLIP	ViT-B/32	3.6	2.8	1.7	13.0	9.9	5.6	21.8
VLG-Net [44]	✓	CLIP	ViT-B/32	3.6	2.8	1.7	11.7	9.3	6.0	21.8
CONE [17]	✓	CLIP	ViT-B/32	8.9	6.9	4.1	20.5	16.1	9.6	21.8
SOONet [39]	✓	CLIP	ViT-B/32	11.3	9.0	5.3	23.2	19.6	13.1	21.8
SnAG [35]	✓	CLIP	ViT-B/32	10.4	8.5	5.5	24.4	20.3	13.4	21.8
RGNet [12]	✓	CLIP	ViT-B/32	12.4	9.5	5.6	25.1	18.7	10.9	21.8
Proposals [45]	✗	CLIP	ViT-B/32	6.6	3.1	1.4	15.1	9.9	5.4	21.8
Watershed	✗	CLIP	ViT-B/32	8.7	5.5	3.2	21.1	13.0	7.3	21.8
Watershed (ours)	✗	ResidualViT	ViT-B/32	8.6	5.4	3.1	20.5	12.6	6.9	$10.2_{(-53\%)}$
Watershed	✗	CLIP	ViT-B/16	10.8	6.8	3.9	24.5	15.2	8.5	84.3
Watershed (ours)	✗	ResidualViT	ViT-B/16	10.1	6.4	3.7	23.5	14.6	8.1	$37.3_{(-56\%)}$
Watershed	✗	CLIP	ViT-L/14	13.3	8.6	5.0	28.5	18.2	10.3	389.2
Watershed (ours)	✗	ResidualViT	ViT-L/14	10.7	7.3	4.3	24.4	16.6	9.3	$171.0_{(-56\%)}$

Table 3. **Long-form video state-of-the-art comparison.** ResidualViT outperforms the previous art both in accuracy and computational cost on the challenging long-form MAD dataset. In these experiments, ResidualViT was configured with N=2, a token dropping probability p=85%, and the center token dropping strategy.

the MAD dataset is due to frequent shot transitions ($\sim$1k per movie), which disrupt the temporal correlations between I- and P-frames, unlike Charades-STA, which contains no such transitions. Nonetheless, these results demonstrate that ResidualViT offers an excellent accuracy-to-cost reduction trade-off across all ViT variants within the MAD dataset.

7. Additional Automatic Audio Description Setting

This section presents further results on the task of Automatic Audio Description. Unlike the results in the main paper (Table 4 in Section 4), we include 64 contextual Audio Descriptions as additional input to the GPT-2 model. Table 4 details the accuracy of both CLIP and ResidualViT features across various backbone sizes without audio context.

In this setup, both the AudioAD baseline equipped with CLIP or ResidualViT features exhibit comparable accuracy. Notably, with the ViT-B/32 backbone, ResidualViT shows an average accuracy drop of 2.2%. However, for the ViT-B/16 and ViT-L/14 backbones, ResidualViT model outperforms CLIP by 1.6% and 1.9%, respectively, while being 56% more efficient.

This experiment further highlights the excellent accuracy vs cost tradeoff achieved by ResidualViT with respect to the CLIP model.

	BertS ↑	R-L ↑	C ↑	M ↑	S ↑	Avg. Cost Feature/sec ↓ (GFLOPs)
CLIP (B/32)	23.8	$\underline{13.0}$	$\underline{17.7}$	5.7	$\underline{5.0}$	21.8
ResidualViT (B/32)	23.9	12.8	17.0	5.5	4.9	$10.2_{(-53\%)}$
CLIP (B/16)	**25.0**	12.9	17.5	5.4	**5.1**	84.3
ResidualViT (B/16)	24.3	**13.2**	**18.0**	$\underline{5.8}$	$\underline{5.0}$	$37.3_{(-56\%)}$
CLIP (L/14)	$\underline{24.5}$	$\underline{13.0}$	17.0	5.6	4.8	389.2
ResidualViT (L/14)	24.2	**13.2**	16.9	**5.9**	**5.0**	$171.0_{(-56\%)}$

Table 4. **Automatic Audio Description.** ResidualViT provides nearly identical captioning quality to CLIP across all backbone sizes and metrics at a cheaper encoding cost.

8. Additional TAL Comparisons

This section presents further results on the task of Temporal Activity Localization. Table 5 complements the results presented in the main paper with comparisons against additional backbones. The ActionFormer [55] baseline is trained from scratch for all sets of features.

We find that all commonly used backbones (*i.e.*, TSN, TMS, SlowFast, VideoSwin, VideoMAE, and InternVideo2) impose a much higher computational cost for feature extraction. This is due to their temporal modeling design, which makes them particularly costly. Conversely, frame-based encoders such as CLIP and ResidualViT are not subjected to such high computational demands yet provide competitive accuracy for the task.

In particular, contrasting ResidualViT (L/14) against TSM-R50 and TSN-R50, we observe that for equivalent

Backbone	mAP ↑ (%)	Avg. Cost Feature/sec ↓ (GFLOPs)
TSM-R50 [28]	34.51	123.3
TSN-R50 [51]	34.64	385.1
SlowFast-R101 [9]	35.95	247.9
VideoSwin-B [32]	35.60	675.0
VideoSwin-L [32]	35.91	2238.8
VideoMAE-H [48]	36.96	4470.0
InternVideo2-6B [52]	**38.95**	5137.5
CLIP (B/32)	34.05	4.4
ResidualViT (B/32)	33.42	$2.0_{(-53\%)}$
CLIP (B/16)	34.40	16.9
ResidualViT (B/16)	33.76	$7.5_{(-56\%)}$
CLIP (L/14)	**34.83**	77.8
ResidualViT (L/14)	34.46	$34.2_{(-56\%)}$

Table 5. **Temporal Action Localization.** ResidualViT provides competitive accuracy at a fraction of the CLIP computational cost. When comparing against additional backbones, ResidualViT provides a good cost vs. accuracy tradeoff with an orders-of-magnitude cheaper model and comparable mAP.

mAP, ResidualViT is 3.6-11.2× more efficient. This finding showcases the excellent accuracy vs. cost tradeoff achieved by ResidualViT.

9. Additional Ablations

In this section, we delve deeper into the design choices of ResidualViT by performing ablation studies on its token reduction mechanisms and distillation strategy. We begin by testing several designs for token-dropping strategies as presented in Section 1 and discussing the role of token-dropping probability. Next, we explore an alternative approach to the token reduction module by replacing token-dropping with a token merging strategy [3]. We then assess the impact of reducing input frame resolution on the total number of tokens, providing insights into its effectiveness as a computational saving technique. Finally, we investigate an alternative distillation objective that eliminates the need for language annotations.

Note that, while semantically aware token reduction strategies [6] could be incorporated, we leave this for future work due to their additional computational demands (*i.e.*, complex token relevance computation at each level of the transformer encoder).

Token Reduction Module Ablation - Token Drop Strategy. Here, we ablate the different token reduction strategies presented in Section 1. In Table 6, we contrast the grounding accuracy of the CLIP model (first row) against our ResidualViT encoder. The lowest grounding accuracy is achieved by the center token reduction strategy with relative drops (vs. the CLIP model, first row) in the range

	Drop Strategy	Charades-STA R@1 ↑ IoU=0.5	IoU=0.7	Avg. Cost per Feature ↓ (GFLOPs)	Memory Cost per Feature ↓ (normalized)
ViT-L/14	–	42.9	24.1	233.4	1×
ViT-L/14	Random	40.8	23.3	102.6	1×
ViT-L/14	Uniform	39.6	22.5	102.6	1×
ViT-L/14	Center	38.6	21.1	102.6	1×
ViT-L/14	Motion	41.5	23.8	102.6	1.9×

Table 6. **Token reduction strategy ablation for ResidualViT.** We ablate four different token reduction strategies on the Charades-STA dataset. For all, we fix the token reduction probability to 85%. Memory cost is normalized according to the baseline memory footprint.

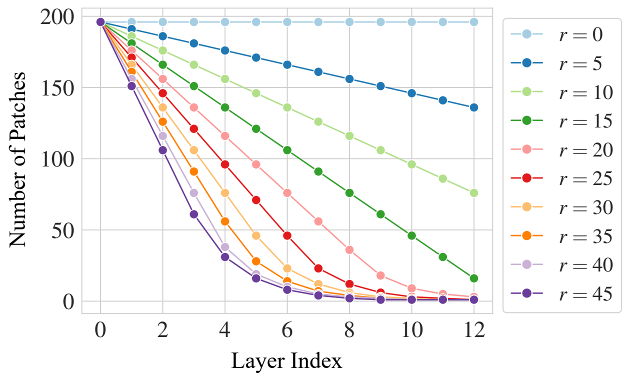
	Drop Strategy	Charades-STA R@1 ↑ IoU=0.5	IoU=0.7	Avg. Cost per Feature ↓ (GFLOPs)	Memory Cost per Feature ↓ (normalized)
ViT-L/14	–	42.9	24.1	233.4	1×
ViT-L/14	Random	20.8	9.5	102.6	1×
ViT-L/14	Uniform	21.0	10.6	102.6	1×
ViT-L/14	Center	25.8	13.2	102.6	1×
ViT-L/14	Motion	28.5	14.5	102.6	1.9×

Table 7. **Token reduction strategy ablation for CLIP.** We ablate four different token reduction strategies on the Charades-STA dataset. For all, we fix the token reduction probability to 85%. Memory cost is normalized according to the baseline memory footprint.

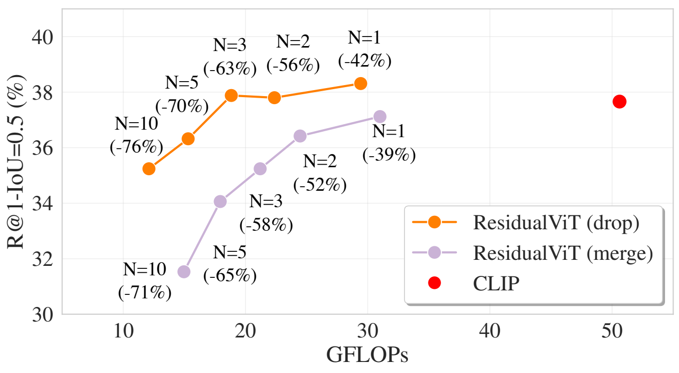
of 10% − 12%. Uniform sampling produces slightly better accuracy with relative drops in the range of 6% − 7%. The second-best performing method is random, which decreases the drop to 3% − 5%. Finally, the motion-based strategy closely matches the grounding accuracy of the CLIP baseline with a relative drop in the range of 1% − 3%. Given the fixed token reduction probability, all settings result in a cost reduction of 56% with respect to the naive CLIP frame encoding baseline.

Additionally, in Table 7, we report the accuracy when the different token reduction strategies are applied to the CLIP model. In this case, we observe much wider differences between different token reduction strategies, where random and uniform strategies perform the worst with a relative accuracy drop in the range of 50% − 60%. The center token reduction strategy provides better accuracy, reducing the losses to 40% − 45%, while motion provides the best trade-off with a 3%4 − 40% drop.

It is important to observe that our model (Table 6) provides a certain level of resilience to the type of token reduction strategy compared to the baseline CLIP model (Table 7). This finding suggests that for ResidualViT, token reduction strategies that avoid motion computation can serve as viable alternatives, especially in scenarios with limited



(a) Tokens decay profile for different r factors.



(b) Drop vs Merge ablation.

Figure 3. **Token dropping vs merging**. (a) We illustrate the relationship between the ViT layer index and the number of tokens resulting from the token merging operation for several canonical merging factors (r). (b) We compare the cost (GFLOPs) vs performance (R@1-IoU=0.5) for CLIP and ResidualViT. We present CLIP without any token reduction strategy (**red**), against our ResidualViT when the token reduction is token dropping (**orange**) or token merging (**purple**). The ablation can conclude that token merging is less favourable due to lower performance at a comparable cost reduction.

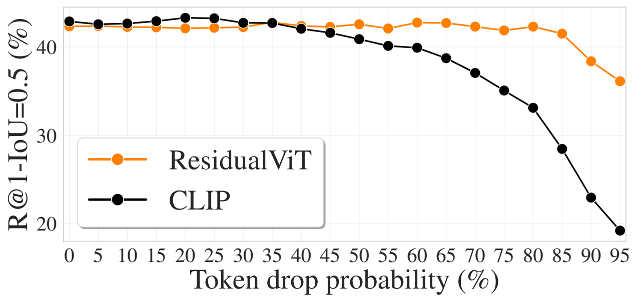


Figure 4. **Token drop probability ablation**. We showcase the performance of CLIP (**black**) and our ResidualViT (**orange**) when progressively increasing the token drop probability.

memory or restricted computational resources. We attribute this finding to the learnable temporal residual connection, which enables the model to effectively compensate for the discarded tokens.

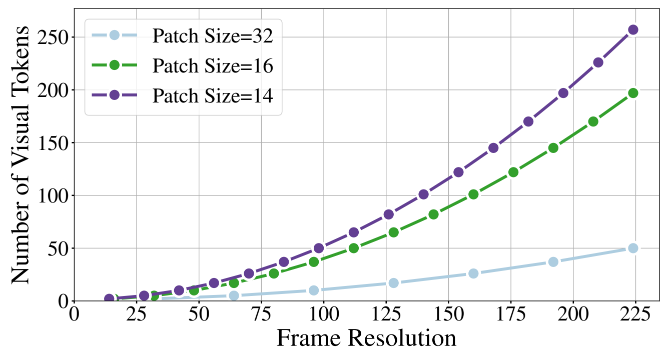
Token reduction probability. Here, we assess how varying the token reduction probability affects the accuracy of both the baseline CLIP model and our ResidualViT model. As depicted in Figure 4, the CLIP model (**black**) demonstrates a degree of robustness to the dropped tokens, maintaining relatively stable grounding accuracy until the token reduction probability reaches $35-40\%$. Beyond this setting, we observe a gradual decline in accuracy, which becomes more pronounced when the probability exceeds 80%. In contrast, thanks to our model design, ResidualViT (**orange**) exhibits a higher tolerance to dropped tokens, retaining relatively high grounding accuracy up to p=85% of dropped tokens.

Token Reduction Module Ablation - Token Merging. Our ResidualViT is agnostic to the implementation of the token reduction method. Therefore, we ablate replacing the token dropping strategy [16, 31] with token merging [3], which has shown promising results in reducing the inference time of pre-trained ViT models.

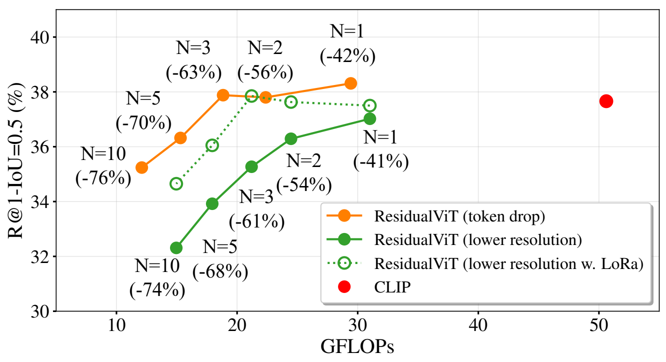
This solution opts for merging a fixed number of tokens per layer, denoted by the r parameter. Within each transformer block, the set of frame tokens at layer l, denoted as $\mathcal{T}^l$, is divided into two subsets: $\mathcal{T}^l_{\textbf{odd}}$, containing tokens at odd indices, and $\mathcal{T}^l_{\textbf{even}}$, containing tokens at even indices. A bipartite matching is computed over the two sets by calculating the cosine similarity between the *key* embeddings of tokens derived from the self-attention mechanism. The r edges of the bipartite graph characterized by the highest similarity define the assignment. The connected tokens are then merged together via a weighted sum, where each token weight represents how many tokens were previously aggregated in it. Note that neither the `[CLS]` token nor the residual token is merged with the frame tokens. Following the bipartite assignment, the maximum number of token mergers per layer is limited to half of the total number of tokens available at layer l ($min(|\mathcal{T}^l|/2, r)$).

This token-reduction strategy has the potential to reduce the information loss that affects the token dropping strategy, as the content of the tokens is retained even if their number is reduced. However, it presents other limitations. (i) Due to the progressive nature of the merging operation (after each transformer layer), to achieve a comparable cost reduction to token dropping, the r parameter must be large. (ii) When the r factor is moderately large, the majority of the tokens are merged together. This effect is showcased in Figure 3a, where we see that for higher values of r, the number of tokens reduces to one quite early in the network (*e.g.*, around the depth of layer 8 for $r = 45$).

In Figure 3b, we conduct a comparative analysis of the



(a) Number of visual tokens vs. frame resolution.



(b) Token Drop vs. lower resolution input ablation.

Figure 5. **Token drop vs lower resolution**. (a) We illustrate the relationship between frame resolution and number of tokens as a function of three canonical patch sizes. (b) We compare the cost (GFLOPs) vs performance (R@1-IoU=0.5) for CLIP and ResidualViT. We present CLIP without any token reduction strategy (**red**), against our ResidualViT with token drop (**orange**) or with lower input resolution (**green**). For the lower resolution setting, we additionally explore using LoRa [18] adapters to finetune the input 2D convolution that implements the patchyfication operation.

token dropping and token merging strategies. For both strategies, we employ the ViT-B/16 backbone model. We set $p = 85\%$ and used the motion-based strategy for token dropping. We set $r = 40$ for token merging. We report R@1-IoU=0.5 grounding accuracy on Charades-STA. We compare the CLIP baseline in **red** against ResidualViT equipped with token dropping (**orange**) or token merging (**purple**).

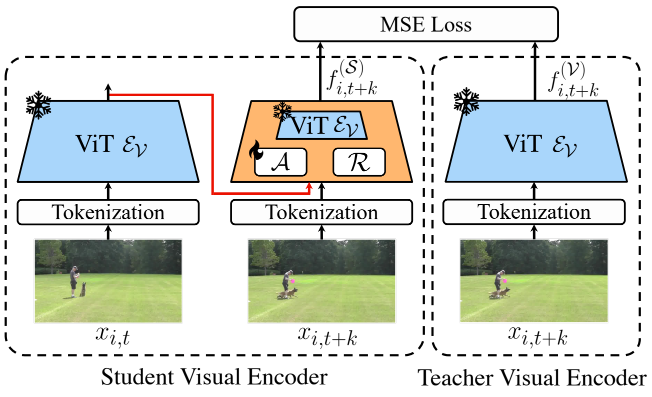
Figure 3b shows token merging achieves overall lower grounding accuracy and incurs a significantly higher computation cost for its highest grounding accuracy setting ($\sim$30 GFLOPs for token merging with N=1 versus $\sim$17 GFLOPs for token dropping with N=3). Nonetheless, both strategies are capable of effectively reducing the cost with respect to the CLIP baseline (**red**). This result validates our design choices for the token reduction module of our ResidualViT.

Spatial Resolution Ablation. An alternative to directly manipulating the number of frame tokens involves adjusting the spatial resolution of input frames. This strategy has proven effective in dual-branch architectures [9], where one branch processes a few high-resolution frames, and the other handles many low-resolution frames. In this section, we compare our approach against this strategy.

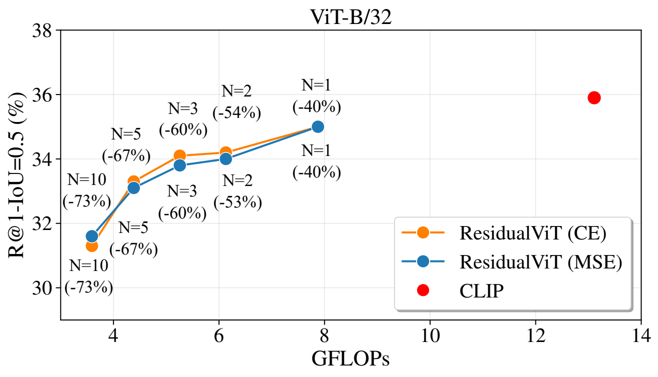
In particular, we forward the full resolution I-frame to the ViT encoder $\mathcal{E}_{\mathcal{V}}$ and N low-resolution P-frames to the ResidualViT encoder $\mathcal{E}_{\mathcal{S}}$. The number of tokens $|\mathcal{T}|$ for an input frame with resolution (H, W) is calculated as $|\mathcal{T}| = \frac{H \times W}{P^2}$, where P denotes the patch size. Consequently, reducing the frame resolution directly decreases the total number of tokens produced from the frame. We provide the relationship between frame resolution, patch size, and number of tokens in Figure 5a, examining trends across three canonical patch sizes: $P \in \{14, 16, 32\}$.

Subsequently, in Figure 5b, we contrast the accuracy of two variations of ResidualViT. One variant employs token dropping (**orange**), while the other utilizes a reduced input frame resolution (**green**), both using the ViT-B/16 backbone model. For the token dropping variant, we set $p = 85\%$, and for reduced resolution, we adjust the spatial dimensions to $H = W = 96$ pixels (as opposed to the default $H = W = 224$). These modifications yield comparable reductions in computational cost, as demonstrated by the alignment of the data points along the x-axis. However, our results indicate that reducing the input resolution is less effective than employing token dropping in terms of accuracy (y-axis). Note that, in all experiments where the token reduction module is modified, we re-train the residual tokenizer to ensure consistent performance evaluation.

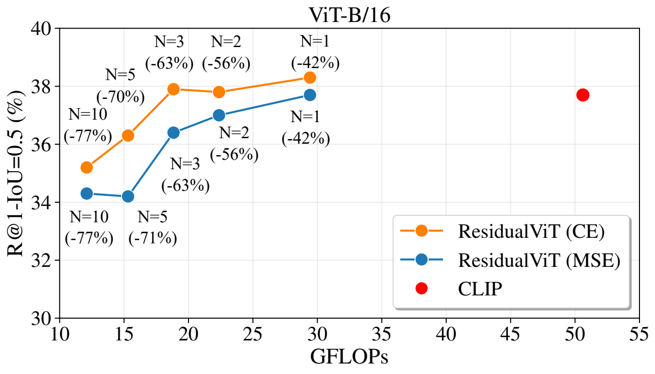
We hypothesize that reducing the input frame resolution compromises the quality of the token representations inputted to the transformer. The process of converting image frames into tokens is implemented through a 2D convolution where both the kernel size and stride are set to the patch size. Previous research has indicated that although convolutional kernels can generalize to different resolutions, substantial changes in resolution can negatively impact accuracy [20, 41]. In our experiments, to match the computational cost reductions observed with the token reduction strategy, we decreased the resolution of inputs to the ResidualViT encoder by a factor of four. To address the resulting resolution mismatch, we explored fine-tuning the 2D convolutional layers using LoRa adapters [18]. This adjustment helps account for the impact of lower-resolution inputs on token representation quality. Our findings show that incorporating LoRa adapters with lower-resolution inputs improves accuracy across all N values and achieves accuracy comparable to the token drop strategy for $N = 3$. How-



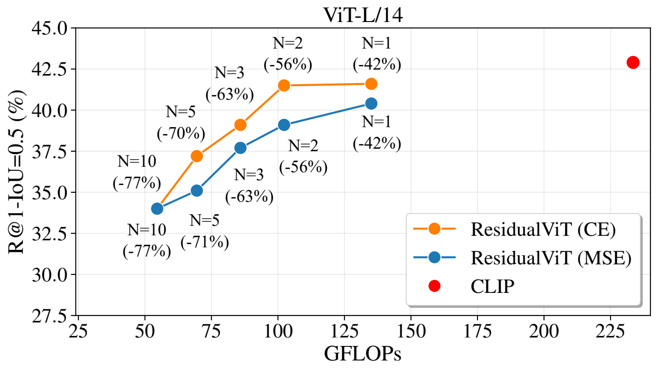
(a) Distillation pipeline.



(b) Downstream performance comparison ViT-B/32 backbone.



(c) Downstream performance comparison ViT-B/16 backbone.



(d) Downstream performance comparison ViT-L/14 backbone.

Figure 6. **Distillation loss ablation.** We ablate replacing the CE loss (Equation 2) with a Mean Square Error (MSE) loss. (a) Depicts the distillation pipeline when the MSE loss is used. (b-c) Summarizes the downstream performance comparison for the three different backbone sizes (ViT-B/32, ViT-B/16, ViT-L/14). The **red** represents CLIP's performance, while the **orange** and **blue** curves represent the performance of ResidualViT on the Charades-STA dataset when the distillation uses the original CE loss or the MSE loss respectively. We perform this ablation adopting the ViT-B/32 backbone. We conclude that the MSE loss, which does not require language annotations, produces near-identical results.

ever, the token drop strategy consistently outperforms this approach while maintaining the advantage of not requiring any weight modifications to the encoder $\mathcal{E}_\mathcal{V}$.

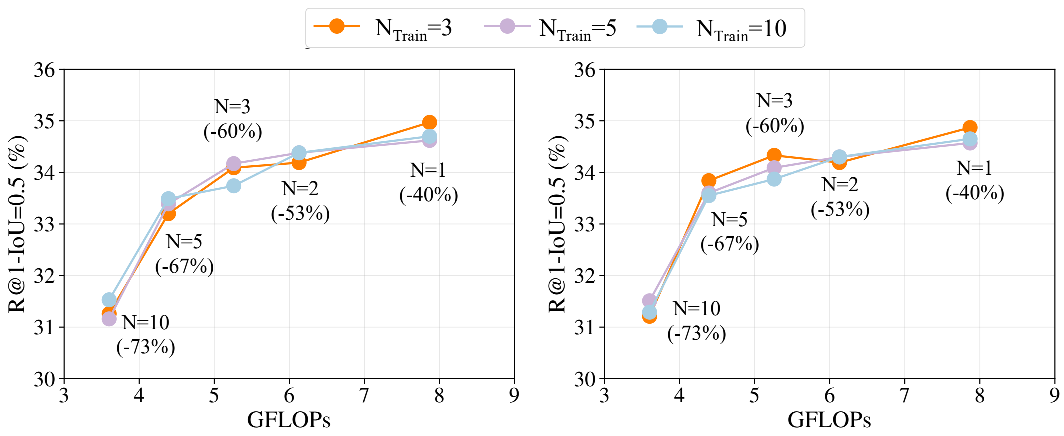
Distillation Strategy. To evaluate our distillation approach, we replace the Cross-Entropy (CE) loss (Equation 2) with a Mean Squared Error (MSE) loss, computed between frame features as $||f_{i,t+k}^{\mathcal{S}} - f_{i,t+k}^{\mathcal{V}}||_2$.

This alternative setup, illustrated in Figure 6a, removes the need for language annotations, reducing training costs by approximately 10% for ViT-B/32, 3% for ViT-B/16, and 1.5% for ViT-L/14. However, we emphasize that our primary focus is on the efficient deployment of a trained model (forward inference), rather than optimizing training efficiency.

Figure 6b presents the results for ViT-B/32, where both loss functions achieve comparable accuracy, indicating that our distillation method remains effective regardless of the loss choice. However, in Figure 6c and Figure 6d, the CE loss consistently outperforms the MSE loss, supporting our choice of a language-supervised CE approach as the optimal strategy. For consistency, all training and testing hyperparameters remain unchanged across these ablations, ensuring that accuracy differences stem solely from the choice of the loss function.

Training Interleave Factor (N_{Train}) Ablation. In this section, we evaluate how varying the interleave factor (N_{Train}) during training impacts ResidualViT's accuracy and computational cost for different N values during inference. Additionally, we explore whether different frame sampling strategies during training affect the model's final accuracy. We consider two distinct sampling approaches: (a) Sample N_{Train} frames per training video at a constant frame rate. (b) Extract N_{Train} frames at a constant frame rate, but randomly subsample the frames before inputting them into the network. Our findings are summarized in Figure 7, where we present the accuracy vs. cost trade-off on the Charades-STA dataset using the B/32 backbone.

In this experiment, we train ResidualViT with varying $N_{\text{Train}} \in 3, 5, 10$ and test these models with $N \in 1, 2, 3, 5, 10$. Note that $N_{\text{Train}} = 3$ is our default setting,



(a) Train with N_{Train} frames sampled with constant FPS.

(b) Randomly subsample from N_{Train} frames extracted at constant FPS.

Figure 7. **Training interleave factor (N_{Train}) ablation.** We compare the accuracy of ResidualViT when three different values of $N_{Train} \in \{3, 5, 10\}$ are used, and two different frame sampling strategies are implemented. In particular, we investigate (a) using all N_{Train} frames sampled at a constant FPS$=1.0$, and (b) sampling a random number of frames from the N_{Train} frames extracted at a constant FPS$=1.0$. Results are reported on the Charades-STA dataset using the B/32 backbone.

used for all other results in the manuscript.

Focusing on Figure 7(a), we observe that different values of N_{Train} produce very similar results, with $N_{\text{Train}} = 10$ showing slightly better accuracy for $N = 5$ and $N = 10$ compared to models trained with $N_{\text{Train}} = 3$.

Figure 7(b) supports the same conclusion. In this case, no clear advantage is observed for larger N_{Train}, as accuracy remains very similar across all configurations. Interestingly, $N_{\text{Train}} = 3$ (our default setting) shows slightly better accuracy for $N = 1$, $N = 3$ and $N = 5$.

Frame Rate Ablation. In this section, we evaluate the accuracy-cost trade-off between frame rate and computational cost for CLIP and ResidualViT on the Charades-STA dataset.

Figure 8 illustrates the accuracy of both models on the NLTVG task as the frame rate varies from 0.5 to 3.0 (our default value). At the default frame rate of 3.0, CLIP achieves an R@1-IoU=0.5 score of 35.9, while ResidualViT achieves 34.2—a slight accuracy drop, but with an approximate 53% reduction in encoding cost. As the frame rate decreases, both methods exhibit a steady decline in accuracy. However, it is noteworthy that ResidualViT at FPS=3 incurs a lower cost than CLIP at FPS=2 while achieving comparable accuracy. Additionally, ResidualViT at FPS=2 outperforms CLIP at FPS=1, with similar computational cost.

Finally, we observe that the decrease in accuracy for ResidualViT as the FPS decreases becomes steeper than for CLIP. We believe that this is due to the large temporal gap between consecutive frames, which hinders the ability of the residual tokenizer to provide valuable information when computing P-features.

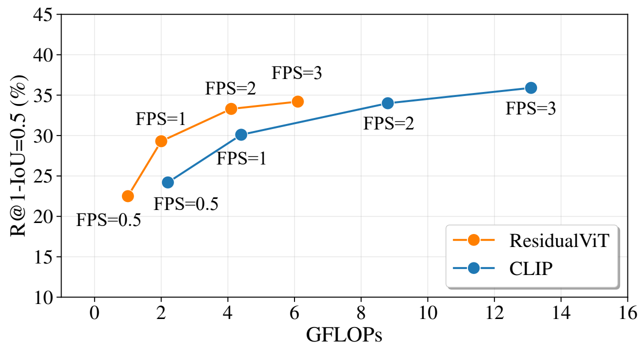


Figure 8. **Frame rate ablation.** We compare CLIP (**blue**) and ResidualViT (**orange**) features on the Charades-STA dataset for varying frame rates. The figure presents the accuracy (y-axis) vs. cost (x-axis) trade-off.

Architecture Ablation on the MAD dataset. Table 8 presents the ablation of the main architecture components of our model on the MAD [45] dataset. This ablation setup is equivalent to the one presented in Table 1 of the main paper.

The first row (**a**) reports the accuracy of the original CLIP model, which requires 21.8 GFLOPs/sec for feature computation. Applying the token reduction strategy uniformly across all frames (**b**) results in an 85% reduction in computational cost but leads to substantial accuracy losses of over 55% across all IoU thresholds. Introducing our interleave strategy with N=2 (**c**) significantly improves grounding accuracy while using only 47% of the original compute budget, with a relative accuracy drop of approximately 25%. Finally, incorporating the residual tokenizer

	Token Reduction	Interleave Factor	Residual Tokenizer (Distilled)	MAD R@1 ↑ IoU=0.3	MAD R@1 ↑ IoU=0.5	Avg. Cost Feature/sec ↓ (GFLOPs)
a.				5.5	3.2	21.8
b.	✓			2.3	1.4	$4.4_{(-85\%)}$
c.	✓	✓		4.2	2.6	$10.2_{(-53\%)}$
d.	✓	✓	✓	5.4	3.1	$10.2_{(-53\%)}$

Table 8. **Architecture ablation.** We ablate the main components of our architecture: the token reduction module, the interleave factor, and the distilled residual tokenizer. We set the token reduction probability p to 85%, $N = 2$, and use the ViT-B/32 backbone on the MAD dataset.

learned via distillation (**d**) adds virtually no computational overhead and nearly matches the original CLIP accuracy, with only a 0.1% absolute drop in accuracy.

These results are consistent with the findings in Table 1 of the main paper and further reinforce the effectiveness of each architectural component.

Interleave Factor N and Benefits of Distillation on the MAD dataset. In Figure 9, we explore the relationship between grounding accuracy and computational cost as we vary the number of interleaved frames (N) on the MAD dataset. Here, the baseline CLIP model is shown in **red**, while our ResidualViT, applied with and without the distilled residual tokenizer module, is shown in **orange** and **blue**, respectively. We vary $N \in \{1, 2, 3, 5, 10\}$. This ablation setup mirrors the one presented in the main paper for the Charades-STA dataset.

We observe that for N=1 and N=2, ResidualViT achieves accuracy on par with CLIP while reducing frame encoding costs by 40% and 53%, respectively. Notably, removing the residual tokenizer (**blue**) leads to a larger accuracy drop, highlighting the importance of distillation. Increasing N beyond 2 yields diminishing returns as cost savings start to plateau around 60%, while accuracy declines. This accuracy drop is attributed to the growing difficulty of predicting the CLIP feature at time t+N from frame t, as CLIP similarity decreases with larger temporal gaps, especially in MAD, which contains more diverse content. Since ResidualViT approximates f_{t+N} using a token subset and the residual tokenizer, lower similarity makes the prediction harder. This can be mitigated by retaining more tokens, but at higher computational cost. As shown in Figure 9, N=2 provides the best trade-off between efficiency and accuracy.

These results align with those in Figure 4 of the main paper and further validate the contributions of both the interleave encoding strategy and the residual tokenizer module.

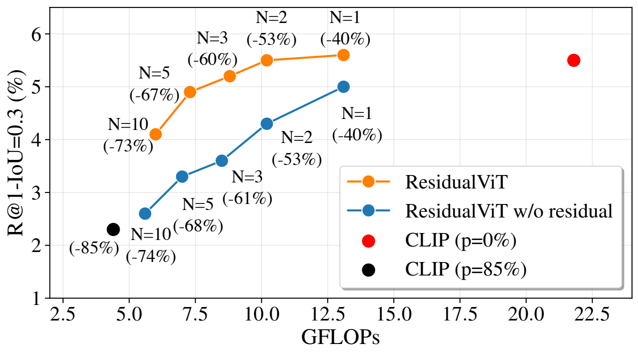


Figure 9. **Interleaving frames** (N). Our ResidualViT (**orange**) closely retains CLIP's (**red**) performance for N=1 and 2 while reducing cost by up to 53%.

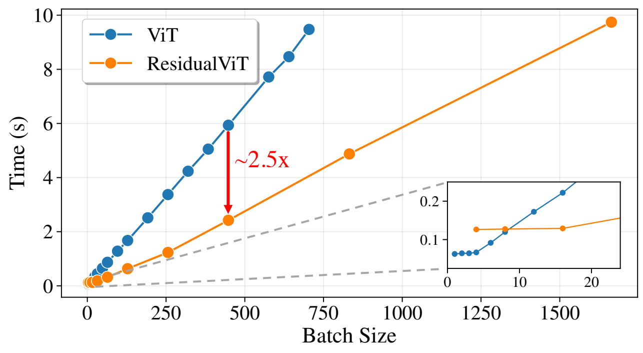


Figure 10. **Inference time comparison**. When varying the batch size, we showcase the runtime difference of a standard ViT (**blue**) against our ResidualViT (**orange**). We demonstrate that our approach is ∼2.5× faster than a standard ViT. Moreover, for the same time budget (*i.e.*, 10 seconds), we can accommodate ∼2.5× more samples in the batch without incurring Out Of Memory issues.

10. ResidualViT Runtime

In our manuscript, we have focused on characterizing the computational cost reductions in terms of GFLOPs. However, our system introduces a dependency where P-feature computation relies on the prior computation of I-features. Specifically, the I-features are first processed through the ViT encoder $\mathcal{E}_\mathcal{V}$, followed by the computation of P-features via the ResidualViT encoder $\mathcal{E}_\mathcal{S}$, which also incorporates the residual token. This design necessitates two sequential forward passes through distinct encoders, prompting us to examine the encoding latency costs inherent to this approach. One possible way to mitigate the latency due to this sequential dependency is via parallel processing via batching of the I-features, followed by batching of the P-features.

In Figure 10, we present the forward pass wall-clock latency as a function of batch size, comparing the timings for a standard ViT-L/14 model and our ResidualViT, which employs the same ViT-L/14 backbone. For each batch size, the total time for ResidualViT is calculated as the sum of the time taken to compute the I-features and the time to process the P-features.

The graph indicates that our ResidualViT is more time-efficient than the ViT baseline, benefiting from our design optimized for efficient video encoding. In practice, our architecture requires roughly 2.5 times less wall-clock time to encode frames into features across most batch sizes. Additionally, when the encoding time is constrained, *e.g*. 10 seconds, the baseline model can process a batch size of approximately 700 frames, whereas ResidualViT can handle a batch size of about 1700 frames.

Note that, in the regime of small batch sizes (*i.e*., ≤ 8), highlighted in the zoomed box in Figure 10, the ViT model proves more economical compared to ResidualViT. Nonetheless, it is crucial to remember that our focus is on efficiently encoding numerous video frames for dense tasks, making ResidualViT the preferred choice under these conditions.

These experiments were performed using a single NVIDIA V100 GPU. Timings for each batch size were obtained by averaging results from 100 consecutive forward passes to ensure statistical reliability. To guarantee precise timing measurements, we employed the PyTorch function `torch.cuda.synchronize()`, which halts the execution of the code until all pending GPU operations are completed. This function is critical for avoiding discrepancies in timing due to asynchronous GPU execution.

11. Additional Task: Action Recognition

In this section, we evaluate the task of action recognition by examining the accuracy gap between CLIP and ResidualViT ($N = 2$, $p = 85\%$) features, using the ViT-B/32 backbone for both models. The experiments are conducted on the Kinetics-400 dataset [21] in a zero-shot setting.

The accuracy comparisons are presented in Table 9. In particular, we investigate the accuracy trends and total encoding costs as the number of frames increases.

We observe that ResidualViT delivers competitive accuracy compared to CLIP features, with a minimum gap of 0.8% for Accuracy@1 at 3 frames and a maximum gap of approximately 3.2% for Accuracy@1 at 4 frames. Similar trends are observed for Accuracy@5. However, when analyzing the accuracy versus total encoding cost, ResidualViT demonstrates a clear advantage: with 4 frames and a total cost of 12.8 GFLOPs, it outperforms CLIP with 3 frames and a total cost of 13.2 GFLOPs for both Accuracy@1 and Accuracy@5. Furthermore, ResidualViT with 7 frames and a total encoding cost of 21.2 GFLOPs achieves nearly identical accuracy to CLIP with 5 frames, which has a higher total cost of 22.0 GFLOPs.

For the zero-shot setup of this experiment, each frame is encoded using either CLIP or ResidualViT, and the resulting visual feature representations are averaged. Classification is performed by combining the class labels with prompt templates provided by the CLIP baseline[1] and encoding the

Number	CLIP		Total encoding	ResidualViT		Total encoding
of Frames	Acc@1 ↑	Acc@5 ↑	cost (GFLOPS)	Acc@1 ↑	Acc@5 ↑	cost (GFLOPS)
1	44.5	72.3	4.4	44.5	72.3	4.4
2	45.0	73.0	8.8	43.4	71.1	6.4
3	43.9	71.5	13.2	43.1	70.7	8.4
4	48.1	76.0	17.6	44.8	73.0	12.8
5	46.5	74.5	22.0	45.1	73.5	14.8
6	48.7	76.8	26.4	45.5	73.9	16.8
7	47.6	75.9	30.8	44.4	72.9	21.2
8	49.3	77.1	35.2	46.5	74.9	23.2
9	48.2	76.7	39.6	46.2	74.8	25.2
10	49.3	77.4	44.0	46.5	75.1	29.6

Table 9. **Action Recognition.** We report accuracy at 1 (Acc@1) and accuracy at 5 (Acc@5) for CLIP and ResidualViT ($N = 2$, $p = 85\%$) features on the Kinetics 400 [21] dataset under a zero-shot setting.

text using the language encoder. All prompt features per class are then averaged, and cosine similarity between the visual and text representations for each class is computed. The classes are ranked by their similarity scores, and the accuracy metric is computed accordingly.

12. Limitations and Discussion

We acknowledge several technical limitations of our approach. First, our method is specifically designed for the Vision Transformer (ViT) architecture, making it less applicable to other architectures, such as convolutional or recurrent neural networks. Nonetheless, we argue that transformer-based models have proven to be among the most versatile and scalable options in the deep learning landscape, supporting their continued adoption and adaptation.

Second, ResidualViT is optimized for dense video processing tasks, which may limit its efficacy in scenarios that benefit from sparse frame sampling, such as action recognition or video retrieval. For such applications, the semantic continuity captured by the residual token across temporally distant frames may not be sufficient, suggesting a potential area for future research.

Third, our token drop strategy relies on motion information derived from the video's compressed representation, which may fail to capture subtle or small-scale movements, such as those occurring in crowded scenes or during fine-grained interactions. This limitation highlights the need for more nuanced motion-aware selection strategies in future work.

Fourth, our method is tailored to reduce the computational burden of video frame encoding in temporally dense tasks. Other aspects of fine-grained video understanding (*e.g*., precise spatial localization) remain open directions for future investigation.

Lastly, our solution's effectiveness heavily relies on

[1]Prompt templates can be found here: https://github.com/openai/CLIP/blob/main/data/prompts.md#kinetics700

the quality of the underlying large pre-trained foundation model, such as CLIP [40]. Consequently, any inherent biases or limitations in the pre-trained model's weights could adversely affect our method's accuracy.

Note that we hypothesize that the WebVid training does not introduce new information beyond what is already in CLIP as the distillation trains only the residual tokenizer so that ResidualViT approximates CLIP features (Section 3.2). Moreover, less than 0.3% of the parameters differ between the two models. We leave for future exploration the full fine-tuning of CLIP on WebVid which has the potential to improve accuracy on downstream tasks. Nonetheless, our approach can seamlessly benefit from this fine-tuned representation instead of CLIP.

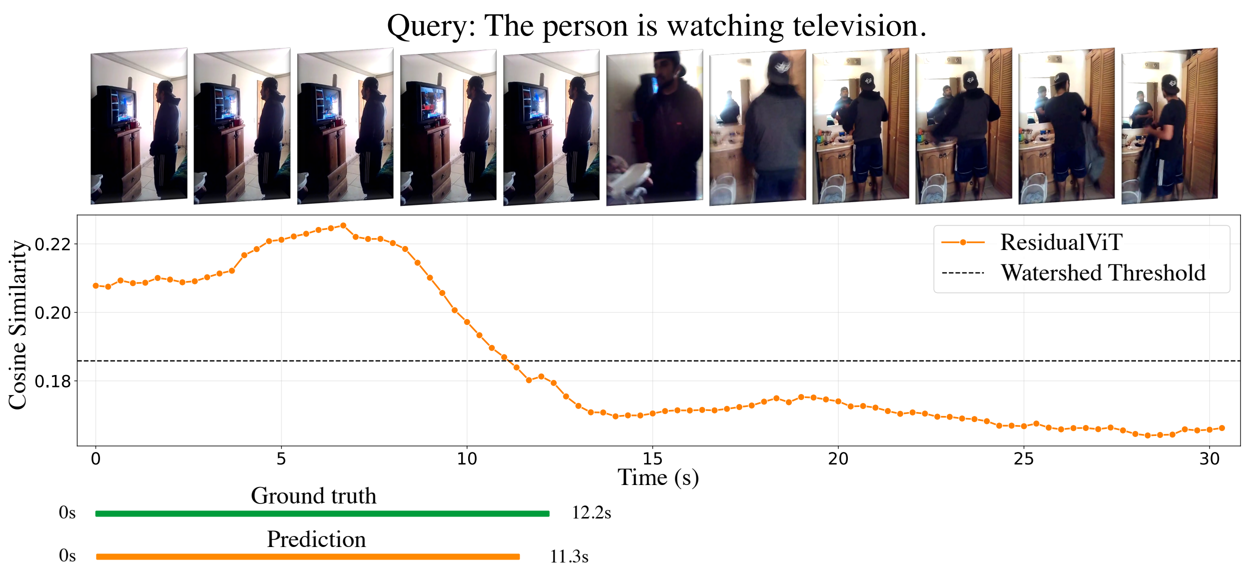
13. Qualitative Results of Natural Language Video Grounding

In Figure 11-13, we present a series of qualitative results from the Charades-STA dataset, demonstrating the efficacy of our zero-shot grounding baseline in identifying relevant event boundaries within video content. In each example, we first show a subset of the video frames along with the textual query on top. Then, we illustrate the temporal sequence of similarity scores $\{S_t\}_{t=1}^{n_v}$ produced by computing the cosine similarity between each frame feature and the sentence feature. We also show the watershed threshold, which is used to determine the start and end moment predictions as detailed in Section 3. For each example, the figure also illustrates the top-1 predicted temporal segment (**orange**) and the ground truth annotation (**green**).

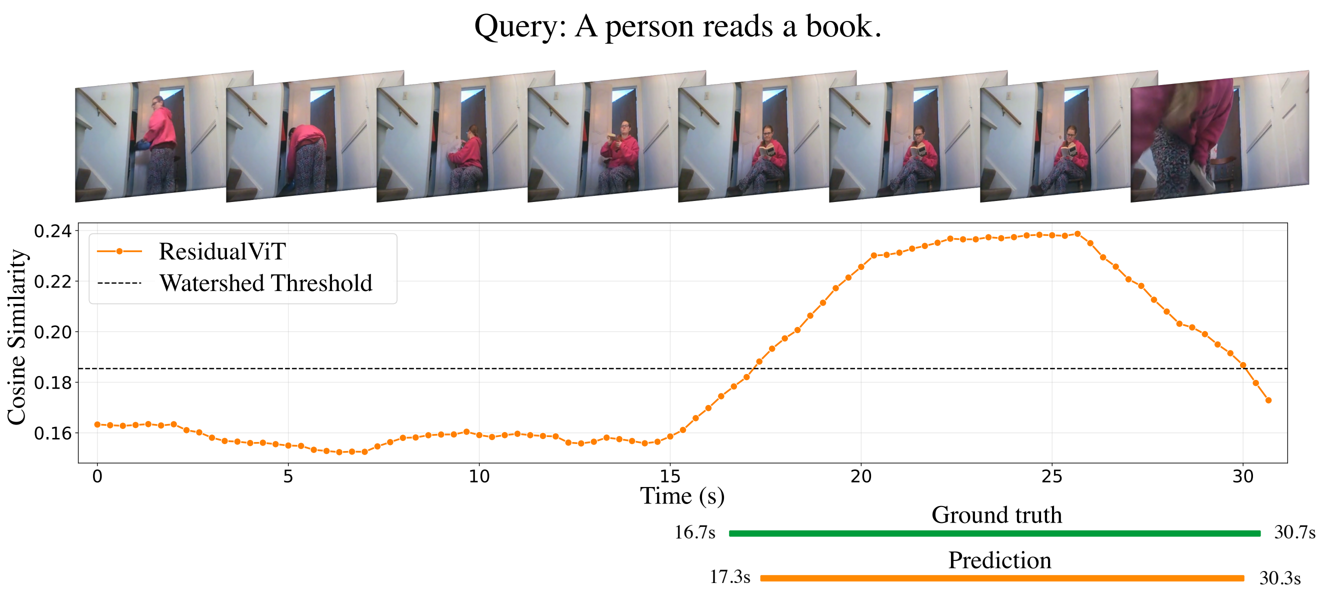
In the examples depicted in Figures 11(a-b) and 12(a-b), our algorithm is capable of discriminating subtle frame differences and produces very precise temporal boundaries that provide an IoU > 0.9 with the ground truth. In example 11a, the feature representations of the frames and the sentence provide higher similarity when the television is present, in accordance with the query *"The person is watching television"*. Similarly, in example 11b, the algorithm can distinguish whether the person is holding a book despite the high resemblance among all frames, correctly predicting the temporal span relative to the textual query *"A person reads a book"* with IoU $= 0.93$. The cosine similarity profile in example 12a clearly differentiates between the section of the video in which the person is eating a sandwich and when they are simply smiling at the camera, predicting the grounding of the action *"A person is eating a sandwich"*, achieving IoU $= 0.98$. Example 12b presents a challenging scenario,*"A person is fixing a light"*, where the model needs to recognize the light's transition from off to on. Despite these complexities, our method provides a correct prediction with an IoU $= 0.95$.

Nonetheless, our approach can provide meaningful predictions that, however, do not align well with the ground truth moment. We detail one such example in Figure 13a. For the query *"A person puts a coffee cup on a shelf"*, we predict a temporal span that is correctly centered to the ground truth span but is twice as long as the ground truth moment, yielding an IoU of approximately 0.5. However, if we pay attention to the video frames, one could argue the prediction is still correct, as it begins when the person opens the cabinet and finishes after the person has placed the coffee cup in it.

Lastly, in Figure 13b, we depict an example in which our proposed solution fails. The action described by the query *"Person undressing by the shelf beside the doorway"* shows a long duration, effectively producing a high similarity response for a good part of the video. This, in turn, affects the watershed threshold, which is proportional to the average similarity scores. Due to the high value of the threshold, our algorithm produces an incorrect prediction that does not overlap with the ground truth.

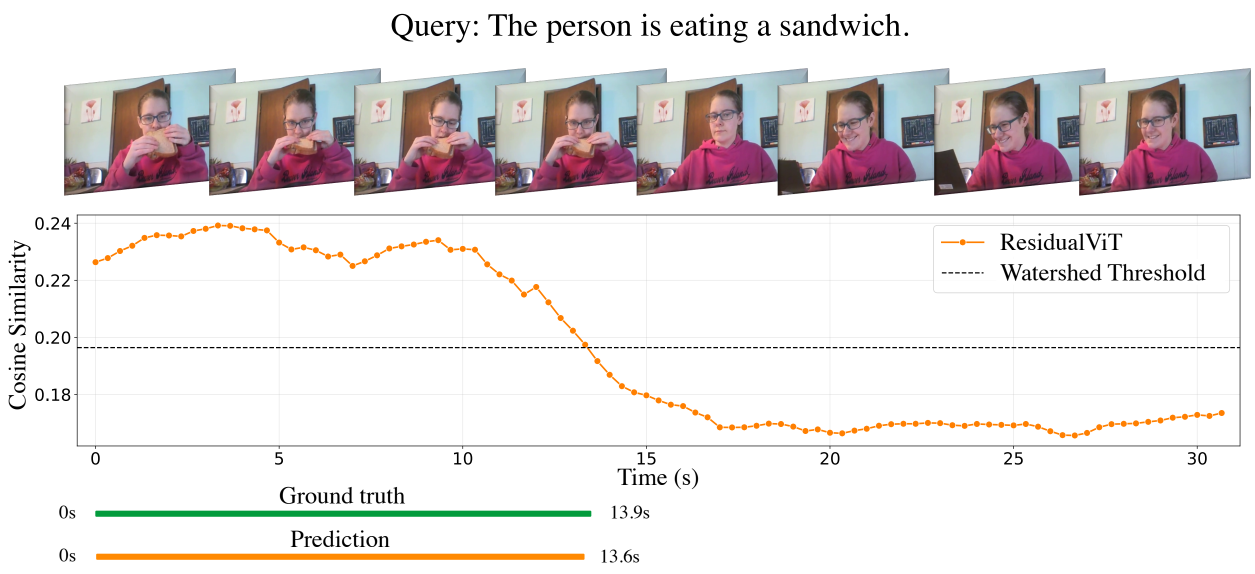


(a) **Grounding example.** We observe an IoU $= 0.93$ between the ground truth moment and the predicted one.

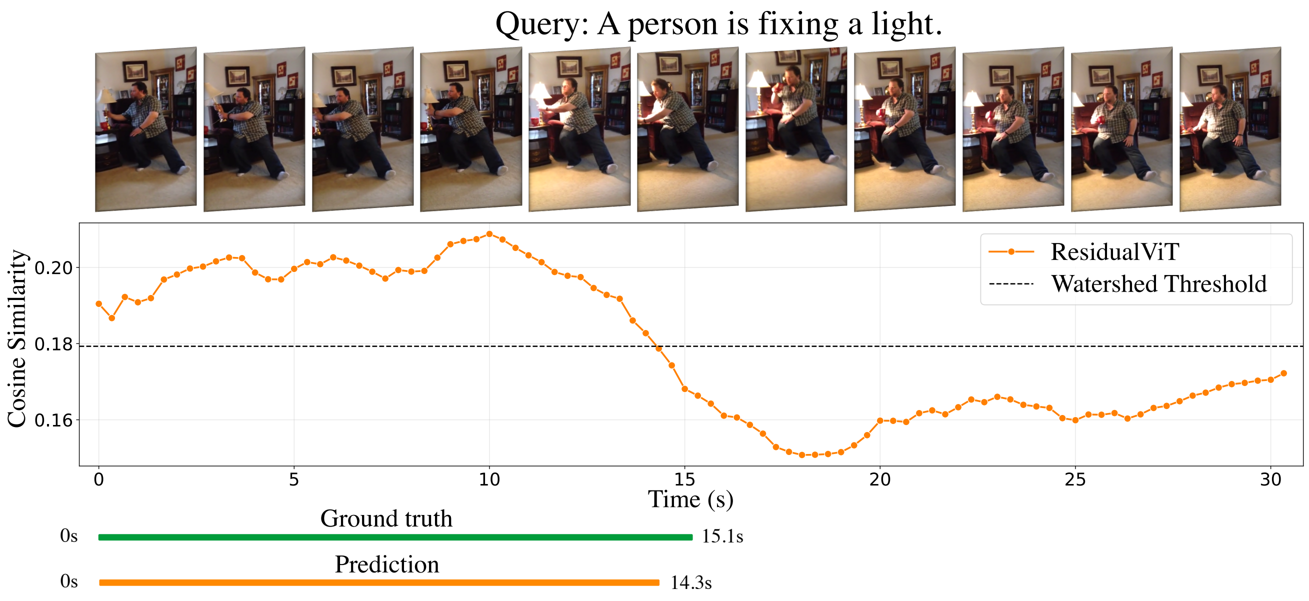


(b) **Grounding example.** We observe an IoU $= 0.93$ between the ground truth moment and the predicted one.

Figure 11. **Qualitative results.** We present two different examples in which our zero-shot algorithm can effectively ground the sentence in the video. We showcase the comparison between the ground truth annotation (**green**) and our top-1 prediction (**orange**).

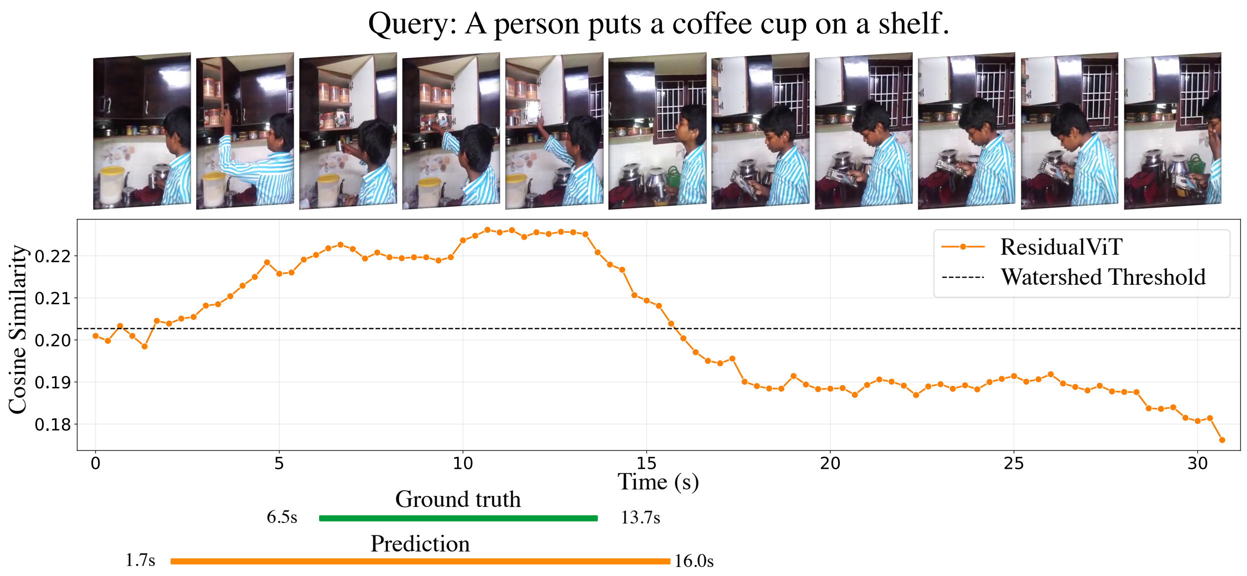


(a) **Grounding example.** We observe an IoU $= 0.98$ between the ground truth moment and the predicted one.

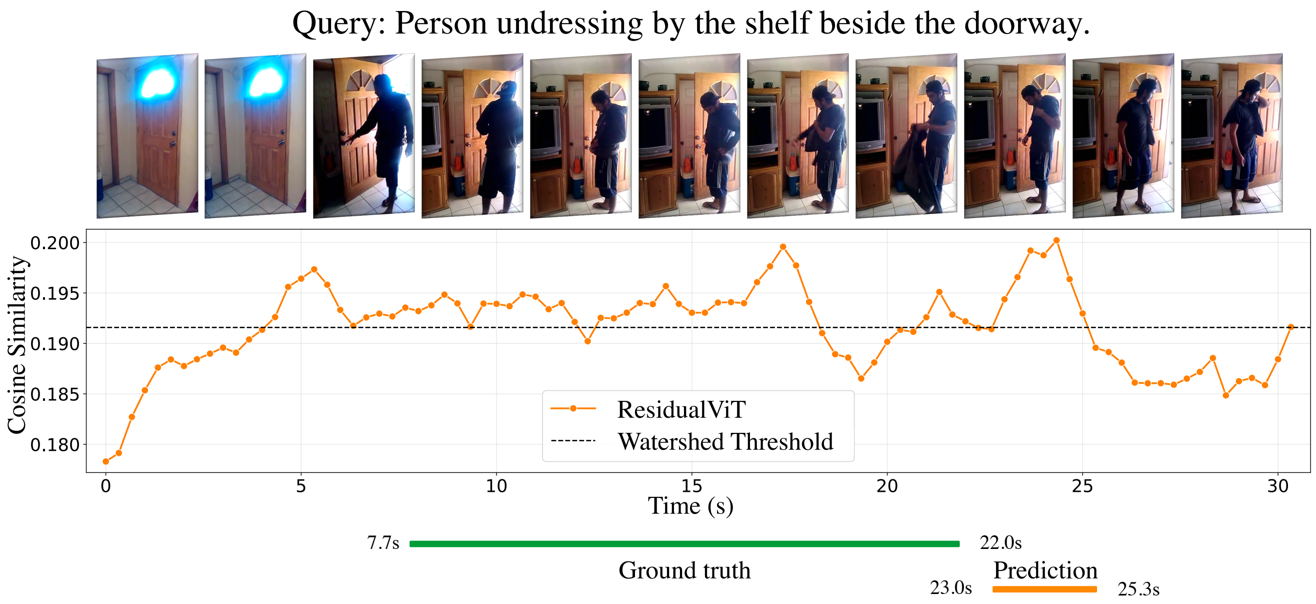


(b) **Grounding example.** We observe an IoU $= 0.95$ between the ground truth moment and the predicted one.

Figure 12. **Qualitative results.** We present two different examples in which our zero-shot algorithm can effectively ground the sentence in the video. We showcase the comparison between the ground truth annotation (**green**) and our top-1 prediction (**orange**).



(a) **Grounding example.** An IoU $= 0.5$ is observed between the temporal annotation and our prediction.



(b) **Grounding example.** Our prediction does not overlap with the ground truth moment.

Figure 13. **Qualitative results.** We present two different examples in which our zero-shot algorithm can effectively ground the sentence in the video. We showcase the comparison between the ground truth annotation (**green**) and our top-1 prediction (**orange**).

References

[1] AcherStyx. Compressed video reader, 2020. Accessed: 2024-02. 2

[2] Wayner Barrios, Mattia Soldan, Alberto Mario Ceballos-Arroyo, Fabian Caba Heilbron, and Bernard Ghanem. Localizing moments in long video via multimodal guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13667–13678, 2023. 3

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 7, 8

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 4

[5] Zhenfang Chen, Lin Ma, Wenhan Luo, Peng Tang, and Kwan-Yee K Wong. Look closer to ground better: Weakly-supervised temporal grounding of sentence in video. *arXiv preprint arXiv:2001.09308*, 2020. 3, 4

[6] Shuangrui Ding, Peisen Zhao, Xiaopeng Zhang, Rui Qian, Hongkai Xiong, and Qi Tian. Prune spatio-temporal tokens by semantic-aware temporal accumulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16945–16956, 2023. 1, 7

[7] Anuj Diwan, Puyuan Peng, and Ray Mooney. Zero-shot video moment retrieval with off-the-shelf models. In *Transfer Learning for Natural Language Processing Workshop*, pages 10–21. PMLR, 2023. 3

[8] Victor Escorcia, Mattia Soldan, Josef Sivic, Bernard Ghanem, and Bryan Russell. Finding moments in video collections using natural language. *arXiv preprint arXiv:1907.12763*, 2019. 3

[9] 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. 5, 7, 9

[10] Junyu Gao and Changsheng Xu. Learning video moment retrieval without a single annotated video. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1646–1657, 2021. 3, 4

[11] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275, 2017. 4

[12] Tanveer Hannan, Md Mohaiminul Islam, Thomas Seidl, and Gedas Bertasius. Rgnet: A unified retrieval and grounding network for long videos. *arXiv preprint arXiv:2312.06729*, 2023. 6

[13] Joakim Bruslund Haurum, Sergio Escalera, Graham W Taylor, and Thomas B Moeslund. Which tokens to use? investigating token reduction in vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 773–783, 2023. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[15] Meghana Holla and Ismini Lourentzou. Commonsense for zero-shot natural language video localization. *arXiv preprint arXiv:2312.17429*, 2023. 3, 4

[16] Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*, 2022. 1, 8

[17] Zhijian Hou, Wanjun Zhong, Lei Ji, Difei Gao, Kun Yan, Wing-Kwong Chan, Chong-Wah Ngo, Zheng Shou, and Nan Duan. Cone: An efficient coarse-to-fine alignment framework for long video temporal grounding. *arXiv preprint arXiv:2209.10918*, 2022. 6

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 9

[19] Jiabo Huang, Yang Liu, Shaogang Gong, and Hailin Jin. Cross-sentence temporal and semantic relations in video activity localisation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7199–7208, 2021. 3, 4

[20] Suresh Prasad Kannojia and Gaurav Jaiswal. Effects of varying resolution on performance of cnn based image classification: An experimental study. *Int. J. Comput. Sci. Eng*, 6(9): 451–456, 2018. 9

[21] 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. 13

[22] Dahye Kim, Jungin Park, Jiyoung Lee, Seongheon Park, and Kwanghoon Sohn. Language-free training for zero-shot video grounding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2539–2548, 2023. 3, 4

[23] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-Captioning Events in Videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 4

[24] Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. *Advances in Neural Information Processing Systems*, 34: 11846–11858, 2021. 3

[25] Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. In *Advances in Neural Information Processing Systems*, pages 11846–11858. Curran Associates, Inc., 2021. 6

[26] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022. 4

[27] Kun Li, Dan Guo, and Meng Wang. Proposal-free video grounding with contextual pyramid network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1902–1910, 2021. 3, 4

[28] 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. 7

[29] Kevin Qinghong Lin, Pengchuan Zhang, Joya Chen, Shraman Pramanick, Difei Gao, Alex Jinpeng Wang, Rui Yan, and Mike Zheng Shou. Univtg: Towards unified video-language temporal grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2794–2804, 2023. 3, 4

[30] Daizong Liu, Xiaoye Qu, Xiao-Yang Liu, Jianfeng Dong, Pan Zhou, and Zichuan Xu. Jointly cross-and self-modal graph attention network for query-based moment localization. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4070–4078, 2020. 3

[31] Yue Liu, Christos Matsoukas, Fredrik Strand, Hossein Azizpour, and Kevin Smith. Patchdropout: Economizing vision transformers using patch dropout. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3953–3962, 2023. 1, 8

[32] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 7

[33] Dezhao Luo, Jiabo Huang, Shaogang Gong, Hailin Jin, and Yang Liu. Zero-shot video moment retrieval from frozen vision-language models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5464–5473, 2024. 3, 4

[34] WonJun Moon, Sangeek Hyun, SuBeen Lee, and Jae-Pil Heo. Correlation-guided query-dependency calibration in video representation learning for temporal grounding. *arXiv preprint arXiv:2311.08835*, 2023. 1, 4, 5

[35] Fangzhou Mu, Sicheng Mo, and Yin Li. Snag: Scalable and accurate video grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18930–18940, 2024. 6

[36] Jonghwan Mun, Minsu Cho, and Bohyung Han. Local-Global Video-Text Interactions for Temporal Grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[37] Jinwoo Nam, Daechul Ahn, Dongyeop Kang, Seong Jong Ha, and Jonghyun Choi. Zero-shot natural language video localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1470–1479, 2021. 3, 4

[38] Mayu Otani, Yuta Nakashima, Esa Rahtu, and Janne Heikkilä. Uncovering hidden challenges in query-based video moment retrieval. *arXiv preprint arXiv:2009.00325*, 2020. 3

[39] Yulin Pan, Xiangteng He, Biao Gong, Yiliang Lv, Yujun Shen, Yuxin Peng, and Deli Zhao. Scanning only once: An end-to-end framework for fast temporal grounding in long videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13767–13777, 2023. 6

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 14

[41] Mats L Richter, Wolf Byttner, Ulf Krumnack, Anna Wiedenroth, Ludwig Schallner, and Justin Shenk. (input) size matters for cnn classifiers. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30*, pages 133–144. Springer, 2021. 9

[42] Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1-2):187–228, 2000. 3

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[44] Mattia Soldan, Mengmeng Xu, Sisi Qu, Jesper Tegner, and Bernard Ghanem. Vlg-net: Video-language graph matching network for video grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3224–3234, 2021. 3, 6

[45] Mattia Soldan, Alejandro Pardo, Juan León Alcázar, Fabian Caba, Chen Zhao, Silvio Giancola, and Bernard Ghanem. Mad: A scalable dataset for language grounding in videos from movie audio descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5026–5035, 2022. 3, 5, 6, 11

[46] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 3

[47] Xin Sun, Jialin Gao, Yizhe Zhu, Xuan Wang, and Xi Zhou. Video moment retrieval via comprehensive relation-aware network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. 4

[48] 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. 7

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 4

[50] Guolong Wang, Xun Wu, Zhaoyuan Liu, and Junchi Yan. Prompt-based zero-shot video moment retrieval. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 413–421, 2022. 3, 4

[51] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 7

[52] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Jilan Xu, Zun Wang, et al. InternVideo2: Scaling video foundation mod-

els for multimodal video understanding. *arXiv preprint arXiv:2403.15377*, 2024. 7

[53] Mengmeng Xu, Mattia Soldan, Jialin Gao, Shuming Liu, Juan-Manuel Pérez-Rúa, and Bernard Ghanem. Boundary-denoising for video activity localization. *arXiv preprint arXiv:2304.02934*, 2023. 3, 6

[54] Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. Dense Regression Network for Video Grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[55] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022. 6

[56] Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. Towards debiasing temporal sentence grounding in video. *arXiv preprint arXiv:2111.04321*, 2021. 3

[57] Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. Learning 2d temporal adjacent networks for moment localization with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12870–12877, 2020. 3, 4, 6

[58] Minghang Zheng, Yanjie Huang, Qingchao Chen, Yuxin Peng, and Yang Liu. Weakly supervised temporal sentence grounding with gaussian-based contrastive proposal learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15555–15564, 2022. 3, 4

[59] Minghang Zheng, Shaogang Gong, Hailin Jin, Yuxin Peng, and Yang Liu. Generating structured pseudo labels for noise-resistant zero-shot video sentence localization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14197–14209, 2023. 3, 4