On Pursuit of Designing Multi-modal Transformer for Video Grounding

Meng Cao[1], Long Chen[2]*, Mike Zheng Shou[3], Can Zhang[1], and Yuexian Zou[1,4]†
[1]School of Electronic and Computer Engineering, Peking University
[2]Columbia University [3]National University of Singapore [4]Peng Cheng Laboratory
mengcao@pku.edu.cn, zjuchenlong@gmail.com
mike.zheng.shou@gmail.com, {zhangcan, zouyx}@pku.edu.cn

Abstract

Video grounding aims to localize the temporal segment corresponding to a sentence query from an untrimmed video. Almost all existing video grounding methods fall into two frameworks: 1) Top-down model: It predefines a set of segment candidates and then conducts segment classification and regression. 2) Bottom-up model: It directly predicts frame-wise probabilities of the referential segment boundaries. However, all these methods are not *end-to-end*, *i.e.*, they always rely on some time-consuming post-processing steps to refine predictions. To this end, we reformulate video grounding as a set prediction task and propose a novel end-to-end multi-modal Transformer model, dubbed as **GTR**. Specifically, GTR has two encoders for video and language encoding, and a cross-modal decoder for grounding prediction. To facilitate the end-to-end training, we use a *Cubic Embedding* layer to transform the raw videos into a set of visual tokens. To better fuse these two modalities in the decoder, we design a new *Multi-head Cross-Modal Attention*. The whole GTR is optimized via a *Many-to-One* matching loss. Furthermore, we conduct comprehensive studies to investigate different model design choices. Extensive results on three benchmarks have validated the superiority of GTR. All three typical GTR variants achieve record-breaking performance on all datasets and metrics, with several times faster inference speed. Our project is available at GTR.

1 Introduction

Video grounding is a fundamental while challenging task for video understanding and has recently attracted unprecedented research attention (Chen et al., 2018, 2019b,a; Zhang et al., 2019a; Liu et al., 2018a; Yuan et al., 2021). Formally, it aims to identify the two temporal boundaries of the moment of interest based on an input untrimmed video and a natural language query. Compared to the conventional video action localization task (Shou et al., 2016; Zhao et al., 2017; Zhang et al., 2021), video grounding is more general and taxonomy-free, *i.e.*, it is not limited by the predefined action categories.

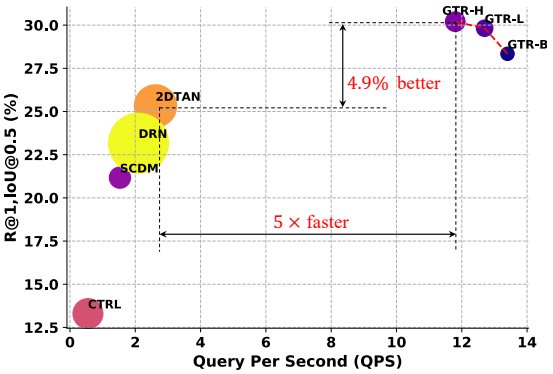

Figure 1: Performance comparisons on TACoS in terms of R@1, IoU@0.5 and Query Per Second (the number of queries that are retrieved each second during inference). Marker sizes are proportional to the model size. Our GTR-H is 4.9% better than 2D-TAN (Zhang et al., 2020b) with 5 times faster speed.

The overwhelming majority of the state-of-the-art video grounding methods fall into two frameworks: 1) Top-down models (Anne Hendricks et al., 2017; Gao et al., 2017; Ge et al., 2019; Chen et al., 2018; Zhang et al., 2019b; Liu et al., 2018b; Yuan et al., 2019a): They always use a propose-and-rank pipeline, where they first generate a set of moment proposals and then select the best matching one. To avoid the proposal bottlenecks and achieve high recall, a vast number of proposals are needed. Correspondingly, some time-consuming post-processing steps (*e.g.*, non-maximum suppression, NMS) are introduced to eliminate redundancy, which makes the matching process inefficient (cf. Figure 1). 2) Bottom-up models (Mun et al., 2020; Rodriguez et al., 2020; Zeng et al., 2020; Chen et al., 2020; Lu et al., 2019): They directly regress the two temporal boundaries of the referential segment from each frame or predict boundary probabilities frame-wisely. Similarly, they need post-processing steps to group or aggregate all frame-wise predictions.

*Work started when Long Chen at Tencent AI Lab.
†Corresponding author.

Although these two types of methods have realized impressive progress in video grounding, it is worth noting that they still suffer several notorious limitations: 1) For top-down methods, the heuristic proposal generation process introduces a series of hyper-parameters. Meanwhile, the whole inference stage is computation-intensive for densely placed candidates. 2) For bottom-up methods, the frame-wise prediction manner overlooks fruitful temporal context relationships, which strictly limits their performance. 3) All these methods are not *end-to-end*, which need complex post-processing steps to refine predictions, and easily fall into the local optimum.

In this paper, we reformulate the video grounding as a set prediction problem and propose a novel end-to-end multi-modal Transformer model **GTR** (video **G**rounding with **TR**ansformer). GTR has two different encoders for video and language feature encoding, and a cross-modal decoder for final grounding result prediction. Specifically, we use a *Cubic Embedding* layer to transform the raw video data into a set of visual tokens, and regard all word embeddings of the language query as textual tokens. Both these visual and textual tokens are then fed into two individual Transformer encoders for respective single-modal context modeling. Afterwards, these contextualized visual and textual tokens serve as an input to the cross-modal decoder. Other input for the decoder is a set of learnable segment queries, and each query try to regress a video moment by interacting with these contextualized tokens. To better fuse these two modalities in the decoder, we design a new *Multi-head Cross-Modal Attention* module (MCMA). The whole GTR model is trained end-to-end by optimizing a *Many-to-One Matching Loss* which produces an optimal bipartite matching between predictions and ground-truth. Thanks to this simple pipeline and the effective relationship modeling capabilities in Transformer, our GTR is both effective and computationally efficient with extremely fast inference speed (cf. Figure 1).

Since our community has few empirical experiences on determining the best design choice for multi-modal Transformer-family models, we conduct extensive exploratory studies on GTR to investigate the influence of different model designs and training strategies, including: (a) *Visual tokens acquisition*. We use a cubic embedding layer to transform raw videos to visual tokens, and discuss the design of the cubic embedding layer from three dimensions. (b) *Multi-modal fusion mechanism*. We propose six types of multi-modal fusion mechanisms, and compare their performance and computation cost thoroughly. (c) *Decoder design principles*. We explore some key design principles of a stronger multi-modal Transformer decoder, such as the tradeoff between depth and width, or the attention head number in different layers. (d) *Training recipes*. We discuss the influence of several training tricks. We hope our exploration results and summarized take-away guidelines can help to open the door for designing more effective and efficient Transformer models in multi-modal tasks.

In summary, we make three contributions in this paper:

1. We propose the first end-to-end model GTR for video grounding, which is inherently efficient with extremely fast inference speed.
2. By the careful design of each component, all variants of GTR achieve new state-of-the-art performance on three datasets and all metrics.
3. Most importantly, our comprehensive explorations and empirical results can help to guide the design of more multi-modal Transformer-family models in other multi-modal tasks.

2 Related Work

Video Grounding. The overwhelming majority of state-of-the-art video grounding methods are top-down models (Anne Hendricks et al., 2017; Gao et al., 2017; Ge et al., 2019; Liu et al., 2018a, 2020; Zhang et al., 2019a, 2020b; Chen et al., 2018; Yuan et al., 2019a; Wang et al., 2020; Xu et al., 2019; Xiao et al., 2021b,a; Liu et al., 2021). Although these top-down models have dominated the performance, they suffer from two inherent limitations: 1) The densely placed proposal candidates lead to heavy computation cost. 2) Their performance are sensitive to the heuristic rules (*e.g.*, the number and the size of anchors). Another type of methods is bottom-up models (Yuan et al., 2019b; Lu et al., 2019; Zeng et al., 2020; Chen et al., 2020, 2018; Zhang et al., 2020a). Some works (He et al., 2019; Wang et al., 2019) resort to Reinforcement Learning to guide the boundary prediction adjustment. However, all existing methods (both top-down and bottom-up) are not end-to-end and require complex post-processing steps. In this paper, we propose a end-to-end model GTR, which directly generates predictions with ultrafast inference speed.

Vision Transformer. Transformer (Vaswani et al., 2017) is a de facto standard language modeling

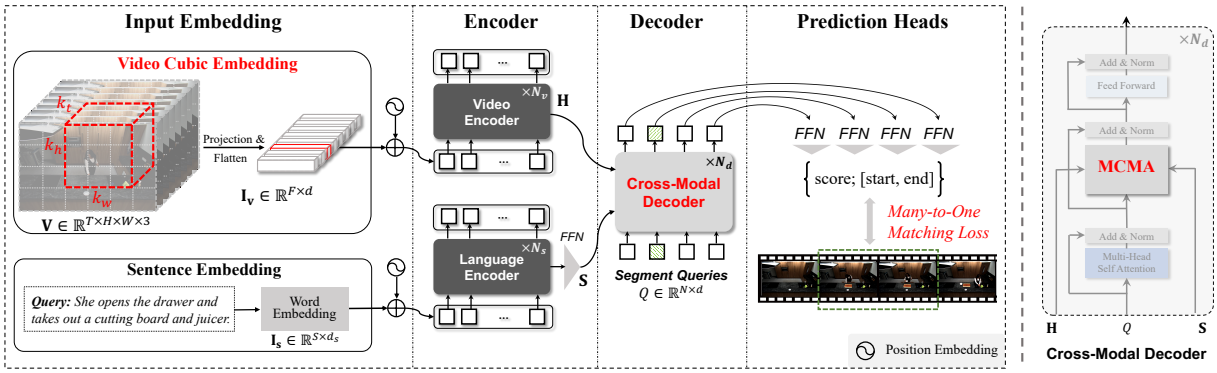

Figure 2: An overview of GTR. (1) Input Embedding transforms the video and language data to feature space. (2) Encoder is applied to encode global context. (3) Decoder contains a novel Multi-head Cross-Modal Attention module (MCMA) to fuse two modalities. (4) Prediction Heads generate grounding results optimized by a Many-to-One Matching Loss.

architecture in the NLP community. Recently, a pioneering object detection model DETR (Carion et al., 2020) starts to formulate the object detection task as a set prediction problem and use an end-to-end Transformer structure to achieve state-of-the-art performance. Due to its end-to-end nature, DETR regains the CV community attention about the Transformer, and a mass of vision Transformer models have been proposed for different vision understanding tasks, such as image classification (Wang et al., 2021), object detection (Carion et al., 2020; Zhu et al., 2021), tracking (Meinhardt et al., 2021; Xu et al., 2021; Chen et al., 2021; Sun et al., 2020), person re-id (He et al., 2021), image generation (Jiang et al., 2021), super resolution (Yang et al., 2020), and video relation detection (Gao et al., 2021). Unlike previous methods only focusing on the vision modality, our GTR is a multi-modal Transformer model, which not only needs to consider the multi-modal fusion, but also has few empirical experience for model designs.

3 Video Grounding with TRansformer

As shown in Figure 2, GTR yields temporal segment predictions semantically corresponding to the given query by four consecutive steps: (1) *Input Embedding*. Given a raw video and a query, this step aims to encode them into the feature space (*i.e.*, visual and textual tokens). 2) *Encoder*. Visual and textual token embeddings are enhanced with a standard Transformer encoder by modeling the intra-modality correlations. (3) *Cross-Modal Decoder*. Contextualized visual and textual token embeddings are fused by a Multi-head Cross-Modal Attention module (MCMA), and a set of learnable segment queries are fed into the decoder to interact with these two modal features. (4) *Prediction Heads*. A simple feed-forward network (FFN) is applied to predict final temporal segments.

3.1 Input Embedding and Encoder

Video Cubic Embedding. Aiming to build a pure Transformer model without the reliance on CNNs, ViT (Dosovitskiy et al., 2021) decomposes input images into a set of non-overlapping patches. To process video data, a straightforward solution is to apply this partition for each frame. However, this simple extension overlooks the frame-wise temporal correlations. Thus, we propose a Cubic Embedding layer which directly extracts 3D visual tokens from the height, width, and temporal dimensions respectively (cf. Figure 2).

Formally, given a raw video, we firstly use framerate $1/\gamma_\tau$ to sample the video, and obtain video clip $\mathbf{V} \in \mathbb{R}^{T\times H\times W\times 3}$. Then, we use a sampling kernel κ with shape (k_h, k_w, k_t) to transform the video into visual tokens, and the sampling kernel κ is propagated with stride size $s(s_h, s_w, s_t)$, where s_h, s_w, and s_t denote the stride size in the height, width, and temporal dimension respectively. Each sampled 3D visual patch is fed into a projection layer, and the output of the cubic embedding layer is a set of visual token embeddings $\mathbf{I_v} \in \mathbb{R}^{F\times d}$, where F is the number of visual tokens, and d is the dimension of the projected layer. In our experiments, we set d to the same hidden dimension of the Transformer. Apparently, $F = O_h \times O_w \times O_t$ where $O_h = \left\lfloor \frac{H-k_h}{s_h} + 1 \right\rfloor$, $O_w = \left\lfloor \frac{W-k_w}{s_w} + 1 \right\rfloor$, and $O_t = \left\lfloor \frac{T-k_t}{s_t} + 1 \right\rfloor$. Compared to the non-overlapping tokenization manners, our cubic embedding layer allows overlapping sampling, which implicitly fuses adjacent spatial-temporal context (More experiments and discussions about the cubic embedding layer are shown in Sec. 4.2). In

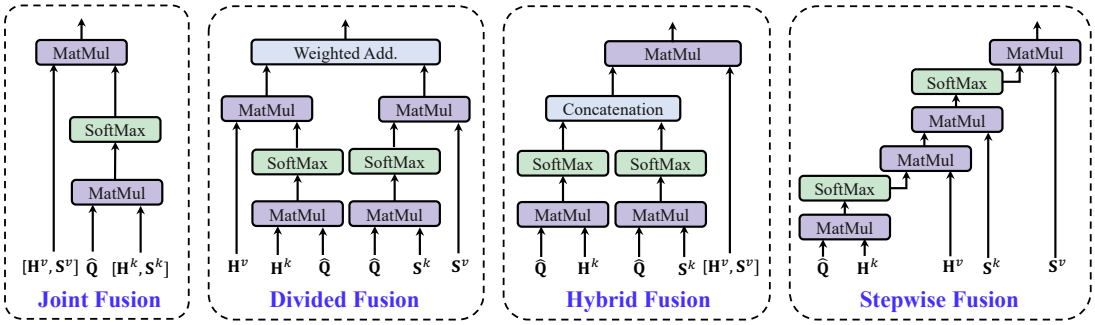

Figure 3: Different instantiations of the Multi-head Cross-Modal Attention module (MCMA).

particular, when setting $s_h = k_h$, $s_w = k_w$, and $s_t = k_t = 1$, our cubic embedding degrades into the prevalent patch embedding in ViT.

Sentence Embedding. For the language input, we first encode each word with pretrained GloVe embeddings (Pennington et al., 2014), and then employ a Bi-directional GRU to integrate the sentence-level embedding feature $\mathbf{I_s} \in \mathbb{R}^{S\times d_s}$, where S represents the length of the sentence query, and d_s is the dimension of textual token embeddings.

Encoder. We use two plain Transformer encoders to model the visual and textual intra-modality context, respectively. Specifically, for the video token embeddings $\mathbf{I_v}$, we apply an encoder with N_v layers to obtain their corresponding contextualized visual tokens $\mathbf{H} \in \mathbb{R}^{F\times d}$. Similarly, for the textual token embeddings $\mathbf{I_s}$, we use an encoder with N_s layers to obtain contextualized textual tokens $\mathbf{S}$. Another feed-forward network with two fully-connected layers is applied to adjust $\mathbf{S}$ to be the same channel with $\mathbf{H}$, *i.e.*, $\mathbf{S} \in \mathbb{R}^{S\times d}$.

3.2 Cross-modal Decoder

As shown in Figure 2, the inputs for cross-modal decoder consist of the visual features $\mathbf{H} \in \mathbb{R}^{F\times d}$, language features $\mathbf{S} \in \mathbb{R}^{S\times d}$, and a set of learnable *segment queries* $\mathbf{Q} \in \mathbb{R}^{N\times d}$. Each segment query $\mathbf{q}_i \in \mathbf{Q}$ tries to learn a possible moment by interacting with $\mathbf{H}$ and $\mathbf{S}$, and the whole decoder will decode N moment predictions in parallel. For accurate segment localization, video grounding requires modeling fine-grained cross-modal relations. To this end, we design a cross-modal decoder with a novel Multi-head Cross-Modal Attention module (MCMA). As shown in Figure 3, we propose several specific instantiations of MCMA[1]:

Joint Fusion. Given visual and language features $\mathbf{H}$ and $\mathbf{S}$, we firstly generate a set of modal-specific key ($\mathbf{H}^k$, $\mathbf{S}^k$) and value ($\mathbf{H}^v$, $\mathbf{S}^v$) pairs by linear transformations: $\mathbf{H}^k = \mathbf{H}\mathbf{W}_h^k$, $\mathbf{S}^k = \mathbf{S}\mathbf{W}_s^k$, $\mathbf{H}^v = \mathbf{H}\mathbf{W}_h^v$, $\mathbf{S}^v = \mathbf{S}\mathbf{W}_s^v$, where $\mathbf{W}_h^k$, $\mathbf{W}_s^k$, $\mathbf{W}_h^v$ and $\mathbf{W}_s^v \in \mathbb{R}^{d\times d}$ are all learnable parameters. Then joint fusion concatenates the two modalities before conducting the attention computing:

$$\mathbf{Y}_{\text{joint}} = \text{MHA}(\hat{\mathbf{Q}}, \mathbf{H}^k \otimes \mathbf{S}^k, \mathbf{H}^v \otimes \mathbf{S}^v),$$

where $\hat{\mathbf{Q}}$ is the enhanced segment query embeddings after the self-attention and $\otimes$ denotes the channel concatenation. MHA stands for the standard Multi-head Attention (Vaswani et al., 2017).

Divided Fusion. We provide a modality-specific attention computation manner, *i.e.,* Divided Fusion decomposes the multi-modal fusion into two parallel branches and the final results are summed up with learnable weights.

$$\mathbf{Y}_{\text{divided}} = \text{MHA}(\hat{\mathbf{Q}}, \mathbf{H}^k, \mathbf{H}^v) \oplus \text{MHA}(\hat{\mathbf{Q}}, \mathbf{S}^k, \mathbf{S}^v),$$

where $\hat{\mathbf{Q}}$, $\mathbf{H}^k$, $\mathbf{H}^v$, $\mathbf{S}^k$ and $\mathbf{S}^v$ are defined the same as in the joint fusion. $\oplus$ denotes the additive sum with learnable weights.

Hybrid Fusion. Hybrid Fusion offers a compromise between Joint Fusion and Divided Fusion. Specifically, the query-key multiplication is conducted separately while the query-value multiplication is still in an concatenation format. Suppose there are n_h self-attention heads. The query, key and value embeddings are uniformly split into n_h segments $\hat{\mathbf{Q}}_i \in \mathbb{R}^{N\times d_h}$, $\mathbf{H}_i^k$, $\mathbf{H}_i^v \in \mathbb{R}^{F\times d_h}$, $\mathbf{S}_i^k$, $\mathbf{S}_i^v \in \mathbb{R}^{S\times d_h}$, $\{i = 1, 2, \ldots, n_h\}$ along channel dimension, where d_h is the dimension of each head and equal to d/n_h. For each head, we apply hybrid fusion in the form:

$$\text{head}_i = \left(\sigma\left(\frac{\hat{\mathbf{Q}}_i \mathbf{H}_i^{k\top}}{\sqrt{d_v}}\right) \otimes \sigma\left(\frac{\hat{\mathbf{Q}}_i \mathbf{S}_i^{k\top}}{\sqrt{d_v}}\right)\right)(\mathbf{H}_i^v \otimes \mathbf{S}_i^v),$$

where σ is the softmax function. The outputs of all heads are then again concatenated along the channel dimension and a linear projection is finally applied to produce the final output as follows:

$$\mathbf{Y}_{\text{hybrid}} = (\text{head}_0 \otimes \text{head}_1 \otimes \ldots \text{head}_{n_h-1})\,\mathbf{W}^O,$$

[1]More details are left in the supplementary materials.

where $\mathbf{W}^O \in \mathbb{R}^{d\times d}$ is linear projection parameters.

Stepwise Fusion. This fusion manner implements the cross-modality reasoning in a cascaded way, *i.e.*, attention computation is performed between $\hat{\mathbf{Q}}$ and video features and then propagated to the sentence modality:

$$\mathbf{Y}_{\text{step}} = \text{MHA}(\text{MHA}(\hat{\mathbf{Q}}, \mathbf{H}^k, \mathbf{H}^v), \mathbf{S}^k, \mathbf{S}^v).$$

We further discuss more multi-modal fusion mechanisms in Sec. 4.2 and supplementary materials.

3.3 Many-to-One Matching Loss

Training: Based on the Cross-Modality Decoder output, a feed forward network is applied to generate a fixed length predictions $\hat{\mathcal{Y}} = \{\hat{y}_i\}_{i=1}^N$, where $\hat{y}_i = (\hat{\mathbf{b}}_i; \hat{c}_i)$ contains temporal segment predictions $\hat{\mathbf{b}}_i \in [0,1]^2$ and the confidence score $\hat{c}_i \in [0,1]$. The ground truth is denoted as $\mathcal{Y}$, which contains the segment coordinate $\mathbf{b} \in [0,1]^2$.

GTR applies set prediction loss (Carion et al., 2020; Stewart et al., 2016) between the fixed-size output sets and ground-truth. Notably, considering each language query only corresponds to one temporal segment, we adapt the *many-to-many* matching in (Carion et al., 2020) to the *many-to-one* version. Specifically, the loss computation is conducted in two consecutive steps. Firstly, we need to determine the optimum prediction slot via the matching cost based on the bounding box similarity and confidence scores as follows:

$$\begin{aligned} i^* &= \underset{i\in[0,N-1]}{\arg\min}\, \mathcal{C}_{\text{match}}\left(\hat{\mathcal{Y}}, \mathcal{Y}\right) \\ &= \underset{i\in[0,N-1]}{\arg\min}\left[-\hat{c}_i + \mathcal{C}_{\text{box}}(\mathbf{b}, \hat{\mathbf{b}}_i)\right]. \end{aligned} \tag{1}$$

In our many-to-one matching loss, the optimal match requires only one iteration of N generated results, rather than checking all possible permutations as in (Carion et al., 2020), which greatly simplifies the matching process. For $\mathcal{C}_{\text{box}}(\cdot,\cdot)$, we define $\mathcal{C}_{\text{box}} = \lambda_{\ell_1}\|b - \hat{b}_i\|_1 + \lambda_{\text{iou}}\mathcal{C}_{\text{iou}}(b, \hat{b}_i)$ with weighting parameters $\lambda_{\ell_1}, \lambda_{\text{iou}} \in \mathbb{R}$. Here $\mathcal{C}_{\text{iou}}(\cdot,\cdot)$ is a scale-invariant generalized intersection over union in (Rezatofighi et al., 2019). Then the second step is to compute the loss function between the matched pair:

$$\mathcal{L}_{\text{set}}(y, \hat{y}) = -\log \hat{c}_{i^*} + \mathcal{C}_{\text{box}}(\mathbf{b}, \hat{\mathbf{b}}_{i^*}), \tag{2}$$

where i^* is the optimal match computed in Eq. (1).

Inference: During inference, the predicted segment set is generated in one forward pass. Then the result with the highest confidence score is selected as the final prediction. The whole inference process requires no predefined threshold values or specific post-processing processes.

4 Experiments

We first introduce experimental settings in Sec. 4.1. Then, we present detailed exploratory studies on the design of GTR in Sec. 4.2. The comparisons with SOTA methods are discussed in Sec. 4.3, and we show visualization results in Sec. 4.4. More results are left in supplementary materials.

4.1 Settings

Datasets. We evaluated our GTR on three challenging video grounding benchmarks: 1) **ActivityNet Captions (ANet)** (Krishna et al., 2017): The average video length is around 2 minutes, and the average length of ground-truth video moments is 40 seconds. By convention, 37,417 video-query pairs for training, 17,505 pairs for validation, and 17,031 pairs for testing. 2) **Charades-STA** (Gao et al., 2017): The average length of each video is around 30 seconds. Following the official splits, 12,408 video-query pairs for training, and 3,720 pairs for testing. 3) **TACoS** (Regneri et al., 2013): It is a challenging dataset focusing on cooking scenarios. Following previous works (Gao et al., 2017), we used 10,146 video-query pairs for training, 4,589 pairs for validation, and 4,083 pairs for testing.

Evaluation Metrics. Following prior works, we adopt "R@n, IoU@m" (denoted as R_n^m) as the metrics. Specifically, R_n^m is defined as the percentage of at least one of top-n retrieved moments having IoU with the ground-truth moment larger than m.

Modal Variants. Following the practice of Visual Transformers or BERTs (Dosovitskiy et al., 2021; Devlin et al., 2019), we also evaluate three typical model sizes: GTR-Base (**GTR-B**, $N_v = 4$, $N_s = 4$, $N_d = 6$, $d = 320$), GTR-Large (**GTR-L**, $N_v = 6$, $N_s = 6$, $N_d = 8$, $d = 320$), and GTR-Huge (**GTR-H**, $N_v = 8$, $N_s = 8$, $N_d = 8$, $d = 512$).[1]

Implementation Details. For input video, the sampling rate $1/\gamma_\tau$ was set to $1/8$, all the frames were resized to 112×112, the kernel shape and stride size were set to (8, 8, 3) and (8, 8, 2), respectively. We used AdamW (Loshchilov and Hutter, 2017) with momentum of 0.9 as the optimizer. The initial learning rate and weight decay were set to 10^{-4}. All weights of the encoders and decoders were initialized with Xavier init, and the cubic embedding

Models	$R_1^{0.5}$	$R_1^{0.7}$	GFLOPs
GTR-B/8	**49.67**	**28.45**	**20.55**
GTR-B/12	44.32	24.62	8.71
GTR-B/16	43.14	23.93	5.44
GTR-B/24	42.01	23.82	1.96

(a) Spatial Configuration

k_t	$R_1^{0.5}$	$R_1^{0.7}$	GFLOPs
3	**49.67**	**28.45**	**20.55**
5	47.34	26.12	17.83
7	47.87	26.91	15.03
9	48.04	27.02	13.91

(b) Temporal Configuration

Models	Stride	$R_1^{0.5}$	$R_1^{0.7}$	GFLOPs
Temporal	(8, 8, 1)	49.73	28.51	41.05
	(8, 8, 2)	**49.67**	**28.45**	**20.55**
Spatial	(4, 4, 3)	44.54	23.12	54.51
	(6, 6, 3)	44.23	22.81	24.26
None	(8, 8, 3)	43.73	22.45	14.69

(c) Overlapping Exploration

Table 1: Input ablations for the GTR-B model[2] on ANet dataset (%).

Models	$R_1^{0.5}$	$R_1^{0.7}$	GFLOPs
Framewise	46.10	26.27	46.36
Cubic	**49.67**	**28.45**	**20.55**

Table 2: Cubic *vs.* Framewise Embedding on ANet(%).

Models	$R_1^{0.5}$	$R_1^{0.7}$	Param(M)	GFLOPs
Early Fusion	39.35	19.64	12.86	11.31
Conditional Fusion	35.25	14.57	12.42	10.60
Joint Fusion	42.51	21.53	16.62	13.91
Divided Fusion	46.91	26.21	25.16	21.02
Hybrid Fusion	**46.82**	**25.45**	**20.94**	**17.25**
Stepwise Fusion	**49.67**	**28.45**	**25.98**	**20.55**

Table 3: Multi-modal fusion comparisons on ANet.(%).

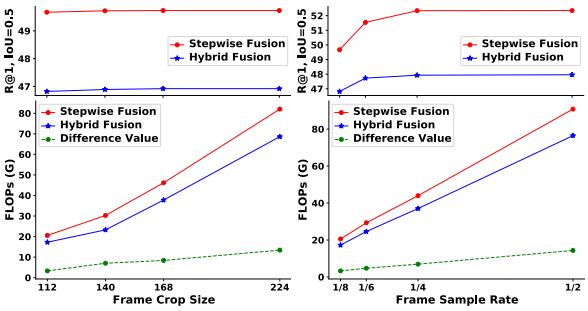

Figure 4: Top: $R_1^{0.5}$ *v.s.* Frame Crop Size & Sample Rate; **Bottom:** FLOPs *v.s.* Frame Crop Size & Sample Rate.

layer was initialized from the ImageNet-pretrained ViT (Dosovitskiy et al., 2021). We used random flip, random crop, and color jitter for video data augmentation. Experiments were conducted on 16 V100 GPUs with batch size 64.

4.2 Empirical Studies and Observations

In this subsection, we conducted extensive studies on different design choices of GTR, and tried to answer four general questions: **Q1:** How to transform a raw video into visual tokens? **Q2:** How to fuse the video and text features? **Q3:** Are there any design principles to make a stronger Transformer decoder? **Q4:** Are there any good training recipes?

4.2.1 Visual Tokens Acquisition (Q1)

Settings. In cubic embedding layer, there are two sets of hyper-parameters (kernel shape (k_w, k_h, k_t) and stride size (s_w, s_h, s_t)). We started our studies from a basic GTR-B model[2], and discussed design choices from three aspects: 1) *Spatial configuration.* We compared four GTR-B variants with different kernel spatial size $k_w, k_h = \{8, 12, 16, 24\}$, and denoted these models as GTR-B/*. Results are reported in Table 1 (a). 2) *Temporal configuration.* We compared four GTR-B variants with different kernel temporal depths $k_t = \{3, 5, 7, 9\}$. Results are reported in Table 1 (b). 3) *Overlapping exploration.* We conducted experiments with several different stride sizes (s_w, s_h, s_t) in Table 1 (c), which corresponds to three basic types (temporal overlapping, spatial overlapping, and non-overlapping).

Observations. 1) Models with smaller patch size (*e.g.*, GTR-B/8) achieve better performance yet at the cost of the dramatic increase of FLOPs. 2) Models with larger kernel temporal depth (k_t) will not always achieve better results. It is worth noting that our cubic embedding will degrade into the prevalent framewise partition in vision Transformers by setting $k_t = s_t = 1$. We further compared cubic embedding with this special case in Table 2, and the results show the superiority of our cubic embedding layer. 3) Compared to non-overlapping and spatial overlapping, temporal overlapping can help to improve model performance significantly. Besides, the performance is not sensitive to the *overlapping degree* in all overlapping cases.

Guides. *The temporal overlapping sampling strategy can greatly boost the performance of the Cubic Embedding layer at an affordable overhead.*

4.2.2 Multi-modal Fusion Mechanisms (Q2)

Settings. As mentioned in Sec. 3.2, we design four types of multi-modal fusion mechanism in the decoder (*i.e.*, joint/divided/hybrid/stepwise fusion), and we group all these fusion mechanisms as *late fusion*. Meanwhile, we also propose two additional fusion mechanisms[1]: 1) *Early fusion*: The multi-modal features are fused before being fed into the decoder. 2) *Conditional fusion*: The language features act as conditional signals of segment queries of the decoder. Results are reported in Table 3.

Observations. 1) All four late fusion models out-

[2]The baseline GTR-B is with the stepwise fusion strategy, and with $s_w = k_w = s_h = k_h = 8$, $k_t = 3$, $s_t = 2$.

Models	ActivityNet Captions				Charades-STA					TACoS					Param
	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	Feat.	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.3}$	$R_5^{0.5}$	QPS	
2D-TAN	44.51	26.54	77.13	61.96	VGG	39.81	23.25	79.33	52.15	37.29	25.32	57.81	45.04	2.36	93.37
DRN	45.45	24.36	77.97	50.30	I3D	53.09	31.75	89.06	60.05	—	23.17	—	33.36	1.82	108.34
SCDM	36.75	19.86	64.99	41.53	I3D	54.44	33.43	74.43	58.08	26.11	21.17	40.16	32.18	1.28	93.82
QSPN	33.26	13.43	62.39	40.78	C3D	35.60	15.80	79.40	45.40	20.15	15.23	36.72	25.30	1.14	95.03
CBP	35.76	17.80	65.89	46.20	C3D	36.80	18.87	70.94	50.19	27.31	24.79	43.64	37.40	1.92	98.55
2D-TAN*	45.21	27.35	77.60	62.32	VGG	40.32	23.63	80.22	52.57	37.89	25.99	58.23	45.21	2.36	93.37
DRN*	46.34	24.92	78.12	50.93	I3D	53.82	32.34	89.74	60.23	—	23.84	—	33.91	1.82	108.34
GTR-B (Ours)	**49.67**	**28.45**	**79.83**	**64.34**	—	**62.45**	**39.23**	**91.40**	**61.76**	**39.34**	**28.34**	**60.85**	**46.67**	**13.4**	**25.98**
GTR-L (Ours)	**50.43**	**28.91**	**80.22**	**64.95**	—	**62.51**	**39.56**	**91.62**	**61.97**	**39.93**	**29.21**	**61.22**	**47.10**	**12.7**	**40.56**
GTR-H (Ours)	**50.57**	**29.11**	**80.43**	**65.14**	—	**62.58**	**39.68**	**91.62**	**62.03**	**40.39**	**30.22**	**61.94**	**47.73**	**11.8**	**61.35**

Table 4: Performance comparisons on three benchmarks(%). All the reported results on ANet and TACoS datasets are based on C3D (Tran et al., 2015) extracted feature. "*" denotes finetuning on corresponding backbones. For pair comparisons, Parma (M) includes the parameter of feature extractor (C3D). **GTR-B** is more efficient while **GTR-H** achieves the highest recall.

perform the early fusion and conditional fusion ones. 2) Among late fusion models, the stepwise fusion model achieves the best performance by using the largest number of parameters. 3) The performance is not sensitive to the crop size, but is positively correlated with sample rate and converges gradually(cf. Figure 4). 4) We find that the FLOPs difference does not increase significantly regarding the crop size and sample rate (cf. Figure 4).

Guides. *Stepwise fusion is the optimal fusion mechanism, even for long and high-resolution videos.*

4.2.3 Decoder Design Principles (Q3)

Settings. To explore the key design principles of a stronger multi-modal Transformer decoder, we considered two aspects: 1) *Deeper vs. Wider*, *i.e.*, whether the decoder should go deeper or wider? Based on the basic GTR-B model[2], we designed two GTR-B variants with nearly equivalent parameters: GTR-B-Wide ($d = 352$) and GTR-B-Deep ($N_d = 8$)[1]. The results are reported in Table 5 (a). 2) *Columnar vs. Shrink vs. Expand.*, *i.e.*, how to design the attention head number in different layers. We tried three different choices: i) the same number of heads in all layers (columnar); ii) gradually decrease the number of heads (shrink); iii) gradually increase the number of heads (expand). Results are reported in Table 5 (b).

Observations. 1) Under the constraint of similar parameters, the deep model (GTR-B-Deep) outperforms the wide counterpart (GTR-B-Wide). 2) The model with the expanding strategy (GTR-B-Expand) achieves the best performance while the shrinking one (GTR-B-Shrink) is the worst.

Guides. *Going deeper is more effective than going wider and progressively expanding the attention heads leads to better performance.*

Models	Param(M)	ActivityNet		TACoS	
		$R_1^{0.5}$	$R_1^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$
GTR-B-Wide	34.05	49.94	28.62	39.52	28.70
GTR-B-Deep	33.89	**50.15**	**28.74**	**39.60**	**29.04**

(a) Wide *vs.* Deep.

Models	Heads	$R_1^{0.3}$	$R_1^{0.5}$
GTR-B-Columnar	[4, 4, 4, 4, 4, 4]	49.42	28.34
GTR-B-Shrink	[8, 8, 4, 4, 2, 1]	49.01	27.95
GTR-B-Expand	[1, 2, 4, 4, 8, 8]	**49.67**	**28.45**

(b) Columnar *vs.* Shrink *vs.* Expand on ANet.

Table 5: **(a)** Performance (%) comparisons between the *wide* variant and *deep* variant. **(b)** Performance (%) comparisons between different attention head number choices.

4.2.4 Training Recipes (Q4)

Settings. Due to the lack of inductive biases, visual Transformers always over-rely on large-scale datasets for training (Dosovitskiy et al., 2021; Touvron et al., 2020). To make multi-modal Transformers work on relatively small multi-modal datasets, we discussed several common training tricks: 1) *Pretrained weights.* Following the idea of pretrain-then-finetune paradigm in CNN-based models[3], we initialize our video cubic embedding layer from the pretrained ViT. Specifically, we initialized our 3D linear projection filters by replicating the 2D filters along the temporal dimension. 2) *Data augmentations.* To study the impact of different data augmentation strategies, we apply three typical choices sequentially, *i.e.*, random flip, random crop, and color jitter. Results are reported in Table 6.

Observations. 1) Using the pretrained cubic embedding weights can help to improve model performance significantly. 2) Color jitter brings the most performance gains among all visual data augmentation strategies. We conjecture that this may be due to the fact that color jitter can change the visual appearance without changing the structured infor-

[3]Modern CNN models always rely on pretrained weights from large-scale datasets as initialization (*e.g.*, ImageNet), and then finetune the weights on specific downstream tasks.

Models	$R_1^{0.5}$	$R_1^{0.7}$
Baseline	41.50	20.13
+ Pretrained Weight	**47.12**$_{+5.62}$	**26.03**$_{+5.90}$
+ Random flip	47.59$_{+0.47}$	26.69$_{+0.66}$
+ Random crop	47.61$_{+0.02}$	26.81$_{+0.12}$
+ Color Jitter	**49.67**$_{+2.06}$	**28.45**$_{+1.64}$

Table 6: Results of different training tricks of the baseline GTR-B model[2] on the ANet dataset.

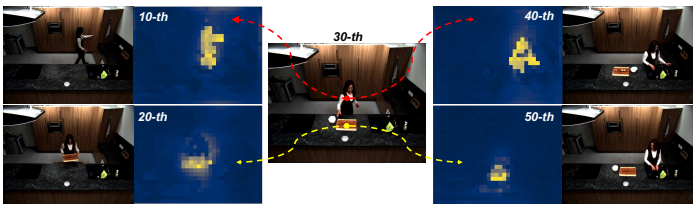

Figure 5: A visualization example of the self-attention in the video encoder.

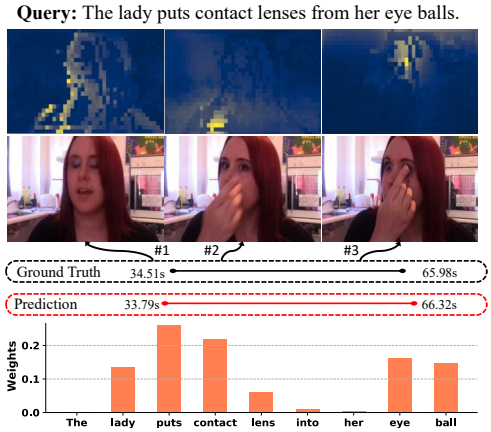

Figure 6: Top: A visualization example of the cross-attention in the decoder. **Bottom:** Visualization of the sentence attention weights.

mation, which is critical for multi-modal tasks.

Guides. *Using pretrained embedding weights and color jitter video augmentation are two important training tricks for multi-modal Transformers.*

4.3 Comparisons with State-of-the-Arts

Settings. We compared three variants of GTR (*i.e.*, GTR-B, GTR-L, and GTR-H) to state-of-the-art video grounding models: **2D-TAN** (Zhang et al., 2020b), **DRN** (Zeng et al., 2020), **SCDM** (Yuan et al., 2019a), **QSPN** (Xu et al., 2019), **CBP** (Wang et al., 2020). These video grounding methods were based on pre-extracted video features (*e.g.*, C3D or I3D) while our GTRs were trained in an end-to-end manner. For more fair comparisons, we selected two best performers (2D-TAN and DRN), and re-trained them by finetuning the feature extraction network. Results are reported in Table 4[1].

Results. From Table 4, we have the following observations: 1) All three GTR variants outperform existing state-of-the-art methods on all benchmarks and evaluation metrics. Particularly, the GTR-H achieves significant performance gains on more strict metrics (*e.g.*, 6.25% and 4.23% absolute performance differences on Charades-STA with metric $R_1^{0.7}$ and TACoS with metric $R_1^{0.5}$, respectively.) 2) The inference speed of all GTR variants are much faster than existing methods (*e.g.*, 13.4 QPS in GTR-B vs. 2.36 QPS in 2D-TAN). Meanwhile, our GTR has fewer parameters than existing methods (*e.g.*, 25.98M in GTR-B vs. 93.37M in 2D-TAN).

4.4 Visualizations

Self-Attention in Video Encoder. We showed an example from TACoS dataset in Figure 5. For a given video snippet (*e.g.*, the 30-*th* frame[4]), we selected two reference patches[4], and visualized the attention weight heatmaps of the last self-attention layer on other four frames (*i.e.*, the 10-*th* frame to 50-*th* frame). From Figure 5, we can observe that the self-attention in the video encoder can effectively focus more on the semantically corresponding areas (*e.g.*, the person and chopping board) even across long temporal ranges, which is beneficial for encoding global context.

Cross-attention in Decoder. An example from ANet dataset was presented in Figure 6. We took the stepwise fusion strategy for MCMA in the decoder. For the output, we selected the segment query slot which outputs the highest confidence scores and visualized its attention weights on two modal features. For the background video frames[4] (#1), the decoder mainly focuses on figure outline. For ground-truth video frames[4] (#2, #3), it captures the most informative parts (*e.g.,* moving hands and touching eye action), which plays an important role in location reasoning. As for the language attention weights, it focuses on essential words, *e.g.*, the salient objects (lady, eye) and actions (put).

5 Conclusions and Future Works

In this paper, we propose the first end-to-end multi-modal Transformer-family model GTR for video grounding. By carefully designing several novel components (*e.g.*, cubic embedding layer, multi-head cross-modal attention module, and many-to-one matching loss), our GTR achieves new state-of-the-art performance on three challenging benchmarks. As a pioneering multi-modal Transformer

[4]We slightly abuse "frame" and "patch" here, and we refer to them as a video snippet and a video cube, respectively.

model, we also conducted comprehensive explorations to summarize several empirical guidelines for model designs, which can help to open doors for the future research on multi-modal Transformers. Moving forward, we aim to extend GTR to more general settings, such as weakly-supervised video grounding or spatial-temporal video grounding.

6 Acknowledgement

This paper was partially supported by National Engineering Laboratory for Video Technology-Shenzhen Division, and Shenzhen Municipal Development and Reform Commission (Disciplinary Development Program for Data Science and Intelligent Computing). It is also supported by National Natural Science Foundation of China (NSFC 6217021843). Special acknowledgements are given to AOTO-PKUSZ Joint Research Center of Artificial Intelligence on Scene Cognition technology Innovation for its support. Mike Shou does not receive any funding for this work.

References

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *ICCV*, pages 5803–5812.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *ECCV*, pages 213–229.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *EMNLP*, pages 162–171.

Jingyuan Chen, Lin Ma, Xinpeng Chen, Zequn Jie, and Jiebo Luo. 2019a. Localizing natural language in videos. In *AAAI*, pages 8175–8182.

Long Chen, Chujie Lu, Siliang Tang, Jun Xiao, Dong Zhang, Chilie Tan, and Xiaolin Li. 2020. Rethinking the bottom-up framework for query-based video localization. In *AAAI*, pages 10551–10558.

Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. 2021. Transformer tracking. In *CVPR*.

Zhenfang Chen, Lin Ma, Wenhan Luo, and Kwan-Yee K Wong. 2019b. Weakly-supervised spatio-temporally grounding natural sentence in video. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *ICCV*, pages 5267–5275.

Kaifeng Gao, Long Chen, Yifeng Huang, and Jun Xiao. 2021. Video relation detection via tracklet based visual transformer. In *ACM MM*.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *WACV*, pages 245–253.

Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. 2019. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *AAAI*, pages 8393–8400.

Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. 2021. Transreid: Transformer-based object re-identification. *arXiv*.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. 2021. Transgan: Two transformers can make one strong gan. *arXiv*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *ICCV*, pages 706–715.

Daizong Liu, Xiaoye Qu, Jianfeng Dong, Pan Zhou, Yu Cheng, Wei Wei, Zichuan Xu, and Yulai Xie. 2021. Context-aware biaffine localizing network for temporal sentence grounding. *arXiv*.

Daizong Liu, Xiaoye Qu, Xiao-Yang Liu, Jianfeng Dong, Pan Zhou, and Zichuan Xu. 2020. Jointly cross-and self-modal graph attention network for query-based moment localization. In *ACM MM*, pages 4070–4078.

Meng Liu, Xiang Wang, Liqiang Nie, Xiangnan He, Baoquan Chen, and Tat-Seng Chua. 2018a. Attentive moment retrieval in videos. In *SIGIR*, pages 15–24.

Meng Liu, Xiang Wang, Liqiang Nie, Qi Tian, Baoquan Chen, and Tat-Seng Chua. 2018b. Cross-modal moment localization in videos. In *ACM MM*, pages 843–851.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv*.

Chujie Lu, Long Chen, Chilie Tan, Xiaolin Li, and Jun Xiao. 2019. Debug: A dense bottom-up grounding approach for natural language video localization. In *EMNLP*, pages 5147–5156.

Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. 2021. Trackformer: Multi-object tracking with transformers. *arXiv*.

Jonghwan Mun, Minsu Cho, and Bohyung Han. 2020. Local-global video-text interactions for temporal grounding. In *CVPR*, pages 10810–10819.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*, pages 25–36.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, pages 658–666.

Cristian Rodriguez, Edison Marrese-Taylor, Fatemeh Sadat Saleh, Hongdong Li, and Stephen Gould. 2020. Proposal-free temporal moment localization of a natural-language query in video using guided attention. In *WACV*, pages 2464–2473.

Zheng Shou, Dongang Wang, and Shih-Fu Chang. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, pages 1049–1058.

Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. 2016. End-to-end people detection in crowded scenes. In *CVPR*, pages 2325–2333.

Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. 2020. Transtrack: Multiple-object tracking with transformer. *arXiv*.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2020. Training data-efficient image transformers & distillation through attention. *arXiv*.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Jingwen Wang, Lin Ma, and Wenhao Jiang. 2020. Temporally grounding language queries in videos by contextual boundary-aware prediction. In *AAAI*, pages 12168–12175.

Weining Wang, Yan Huang, and Liang Wang. 2019. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *CVPR*, pages 334–343.

Wenxiao Wang, Lu Yao, Long Chen, Deng Cai, Xiaofei He, and Wei Liu. 2021. Crossformer: A versatile vision transformer based on cross-scale attention. *arXiv*.

Shaoning Xiao, Long Chen, Jian Shao, Zhuang Yueting, and Jun Xiao. 2021a. Natural language video localization with learnable moment proposals. In *EMNLP*.

Shaoning Xiao, Long Chen, Songyang Zhang, Wei Ji, Jian Shao, Lu Ye, and Jun Xiao. 2021b. Boundary proposal network for two-stage natural language video localization. In *AAAI*.

Huijuan Xu, Kun He, Bryan A Plummer, Leonid Sigal, Stan Sclaroff, and Kate Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*, pages 9062–9069.

Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. 2021. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv*.

Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. 2020. Learning texture transformer network for image super-resolution. In *CVPR*, pages 5791–5800.

Yitian Yuan, Xiaohan Lan, Xin Wang, Long Chen, Zhi Wang, and Wenwu Zhu. 2021. A closer look at temporal sentence grounding in videos: Datasets and metrics. *arXiv*.

Yitian Yuan, Lin Ma, Jingwen Wang, Wei Liu, and Wenwu Zhu. 2019a. Semantic conditioned dynamic modulation for temporal sentence grounding in videos. In *NeurIPS*.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2019b. To find where you talk: Temporal sentence localization in video with attention based location regression. In *AAAI*, pages 9159–9166.

Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. 2020. Dense regression network for video grounding. In *CVPR*, pages 10287–10296.

Can Zhang, Meng Cao, Dongming Yang, Jie Chen, and Yuexian Zou. 2021. Cola: Weakly-supervised temporal action localization with snippet contrastive learning. In *CVPR*, pages 16010–16019.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2019a. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *CVPR*, pages 1247–1257.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. 2020a. Span-based localizing network for natural language video localization. In *ACL*.

Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. 2020b. Learning 2d temporal adjacent networks for moment localization with natural language. In *AAAI*, pages 12870–12877.

Zhu Zhang, Zhijie Lin, Zhou Zhao, and Zhenxin Xiao. 2019b. Cross-modal interaction networks for query-based moment retrieval in videos. In *SIGIR*, pages 655–664.

Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. 2017. Temporal action detection with structured segment networks. In *ICCV*, pages 2914–2923.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*.

7 Appendix

7.1 GTR Variant Settings

We list typical GTR variant parameter settings in Table 7.

Models	N_v	N_s	N_d	d	Params(M)
GTR-B	4	4	6	320	25.98
GTR-L	6	6	8	320	40.56
GTR-H	8	8	8	512	61.35
GTR-B-Wide	4	4	6	352	34.05
GTR-B-Deep	4	4	8	320	33.89
GTR-L-Wide	6	6	8	352	47.18
GTR-L-Deep	6	6	10	320	47.23

Table 7: Typical GTR variant parameter settings.

7.2 Empirical Studies and Observations

7.2.1 Visual Token Acquisition

Through ablation studies, we have found that temporal overlapping sampling plays a critical role in Cubic Embedding. Here we present more experimental results to confirm our findings. Specifically, we choose three kernel sizes (8, 8, 3), (16, 16, 3), and (8, 8, 6). In each case, we set five sets of stride shape corresponding to temporal overlapping, spatial overlapping, and non-overlapping cases, respectively. Experimental results in Table. 8 show that temporal overlapping brings about significant performance improvement under various kernel shape settings.

We also compare our Cubic Embedding layer with the framewise partition in Vision Transformer. Part of the results have been listed in the main paper and the full results are shown in Table. 9. It demonstrates the superiority of our cubic embedding layer.

7.2.2 Decoder Design Principles

Columnar vs. Shrink vs. Expand. We provide more experiments to determine how to design the attention head number in different layers. We set up three different distributions of attention heads: i) the same number of heads in all layers (columnar); ii) gradually decrease the number of heads (shrink); iii) gradually increase the number of heads (expand). The results are listed in Table. 10. It is consistent with our conclusion that **progressively expanding the attention head leads to better performance**.

Deep vs. Wide. We have developed two variants (GTR-B-Wide, GTR-B-Deep) and reported the results on ActivityNet Caption and TACoS datasets, which demonstrates that going deeper is more effective than going wider. Here we provide two additional variants (GTR-L-Wide, GTR-L-Deep) and report results on all three datasets in Table 12. Similarly, the deep model also outperforms the wide model, which further confirms our conclusion.

Models	Stride	$R_1^{0.5}$	$R_1^{0.7}$	GFLOPs
kernel (8, 8, 3)				
Temporal	(8, 8, 1)	49.73	28.51	41.05
	(8, 8, 2)	49.67	28.45	20.55
Spatial	(4, 4, 3)	44.54	23.12	54.51
	(6, 6, 3)	44.23	22.81	24.26
None	(8, 8, 3)	43.73	22.45	14.69
kernel (16, 16, 3)				
Temporal	(16, 16, 1)	43.23	24.13	10.83
	(16, 16, 2)	43.14	23.93	5.44
Spatial	(8, 8, 3)	41.26	21.58	13.33
	(12, 12, 3)	41.03	21.35	6.42
None	(16, 16, 3)	40.83	21.26	3.90
kernel (8, 8, 6)				
Temporal	(8, 8, 2)	46.71	26.84	17.93
	(8, 8, 4)	46.10	26.71	8.99
Spatial	(4, 4, 6)	42.96	21.93	22.21
	(6, 6, 6)	42.95	21.90	9.90
None	(8, 8, 6)	42.84	21.61	6.01

Table 8: Input ablations for GTR-B on ActivityNet Caption dataset(%).

7.2.3 Multi-modal Fusion Mechanisms

In general, multi-modal fusion methods can be divided into three categories (cf. Figure. 7): 1) *Early fusion:* The multi-modal features are fused before being fed into the decoder. 2) *Conditional fusion*: The language feature acts as conditional signals of segment queries of the decoder. We concatenate it with the learnable segment queries features. 3) *Late fusion:* Multi-modal fusion is conducted with the decoder via the proposed Multi-head Cross-modal Attention (MCMA) Module.

For the specific instantiations of MCMA, except for the four fusion strategies mentioned in the main paper, we additionally present two additional variants in Figure. 8. **1)** *Hybrid Fusion (value split)* concatenates the key features and computes the query and value multiplication separately. Specifically, $\hat{\mathbf{Q}} \in \mathbb{R}^{N\times d}$, $\mathbf{H}^k \in \mathbb{R}^{F\times d}$, $\mathbf{S}^k \in \mathbb{R}^{S\times d}$ generate the attention matrix with shape $(F+S)\times d$, which is divided into two parts and each is with shape $F \times d$ and $S \times d$, respectively. These two divided attention weights are used to computed with $\mathbf{H}^v$ and $\mathbf{S}^v$ separately to generate the final results. **2)** *Stepwise Fusion (language-vision)* fuses the language feature firstly and then the video features.

Models	ActivityNet Captions				Charades-STA				TACoS			
	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.3}$	$R_5^{0.5}$
Framewise	46.10	26.27	76.51	62.04	60.05	36.94	89.40	60.25	35.98	26.34	57.09	45.72
Cubic	49.67	28.45	79.83	64.34	62.45	39.23	91.40	61.76	39.34	28.34	60.85	46.67

Table 9: Framewise: ViT embedding manner; Cubic: ours.

Models	Heads	ActivityNet Captions				Charades-STA				TACoS			
		$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.3}$	$R_5^{0.5}$
GTR-B-Columnar	[4, 4, 4, 4, 4, 4]	49.42	28.34	79.75	64.20	62.34	39.02	91.25	61.41	39.01	27.93	60.42	46.39
	[8, 8, 8, 8, 8, 8]	49.43	28.34	79.76	64.23	62.36	39.03	91.26	61.41	39.03	27.93	60.42	46.39
GTR-B-Shrink	[8, 8, 4, 4, 2, 1]	49.01	27.95	79.26	64.00	62.15	38.81	90.93	61.45	38.95	27.73	59.93	45.92
	[16, 16, 8, 8, 4, 1]	49.03	27.96	79.26	64.02	62.16	38.81	90.93	61.45	38.95	27.74	59.94	45.92
GTR-B-Expand	[1, 2, 4, 4, 8, 8]	49.67	28.45	79.83	64.34	62.45	39.23	91.40	61.76	39.34	28.34	60.85	46.67
	[1, 4, 8, 8, 16, 16]	49.64	28.42	79.81	64.32	62.45	39.22	91.41	61.77	39.33	28.33	60.85	46.67

Table 10: Columnar vs. Shrink vs. Expand. Progressively expanding the attention heads leads to better performance.

The experimental results are presented in Table. 13 and we have the following findings. 1) Stepwise fusion has the best performance on all three datasets by using the largest number of parameters. 2) The performance of hybrid fusion (value split) is almost the same as hybrid fusion. 3) Also, stepwise fusion(L-V) shares the similar performance with stepwise fusion, which demonstrates that the order of fusion of visual or language information is not sensitive.

To investigate the influence of frame crop size and sample rate, we conduct more experiments on stepwise fusion models. The results in Table. 11 show that 1) the performance is not sensitive o the frame crop size; 2) the performance is positively correlated with the sample rate but gradually reach convergence.

7.3 Performance of GTR

The performance of GTR under more IoUs is available in Table. 14.

Sample Rate	Crop size	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$
1/8	112	49.67	28.45	79.83	64.34
	140	49.72	28.52	79.95	64.40
	168	49.73	28.52	79.97	64.40
	224	49.73	28.52	79.97	64.40
1/8	112	49.67	28.45	79.83	64.34
1/6		51.54	29.43	80.11	64.76
1/4		52.34	29.57	80.14	64.79
1/2		52.35	29.57	80.14	64.79

Table 11: Performance (%) on ANet on different frame crop size and sample rate.

Models	Param(M)	ActivityNet Captions				Charades-STA				TACoS			
		$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.3}$	$R_5^{0.5}$
GTR-B-Wide	34.05	49.94	28.62	79.86	64.43	62.49	39.41	91.42	61.85	39.52	28.70	60.91	46.84
GTR-B-Deep	33.89	**50.15**	**28.74**	**80.14**	**64.60**	**62.50**	**39.50**	**91.58**	**61.93**	**39.60**	**29.04**	**61.04**	**47.03**
GTR-L-Wide	47.18	50.48	28.96	80.28	65.00	62.51	39.58	91.62	61.94	39.98	29.35	61.45	47.30
GTR-L-Deep	47.23	**50.52**	**29.05**	**80.37**	**65.03**	**62.54**	**39.66**	**91.62**	**62.00**	**40.12**	**29.51**	**61.75**	**47.52**

Table 12: Performance (%) comparisons between the wide variants and deep variants.

Models	ActivityNet Captions				Charades-STA				TACoS				Param(M)	GFLOPs
	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.3}$	$R_5^{0.5}$		
Early Fusion	39.35	19.64	70.39	56.34	54.13	32.72	84.69	54.53	28.25	19.56	50.46	36.06	12.86	11.31
Conditional Fusion	35.25	14.57	66.35	51.34	50.62	25.65	79.34	49.03	24.62	14.37	47.34	34.74	12.42	10.60
Joint Fusion	42.51	21.53	71.24	58.04	57.66	33.68	87.52	57.34	33.69	23.72	54.27	40.14	16.62	13.91
Divided Fusion	46.91	26.21	76.85	63.13	60.93	38.22	89.84	59.95	37.82	26.31	58.13	44.80	25.16	21.02
Hybrid Fusion	**46.82**	**25.45**	**76.20**	**62.41**	**60.21**	**37.25**	**89.63**	**59.51**	**37.01**	**25.77**	**57.63**	**44.32**	**20.94**	**17.25**
Hybrid Fusion(value split)	46.69	25.32	75.83	61.93	59.94	36.93	89.52	59.32	36.85	25.49	57.42	44.05	20.82	16.75
Stepwise Fusion	**49.67**	**28.45**	**79.83**	**64.34**	**62.45**	**39.23**	**91.40**	**61.76**	**39.34**	**28.34**	**60.85**	**46.67**	**25.98**	**20.55**
Stepwise Fusion(L-V)	49.62	28.37	79.76	64.25	62.37	38.93	91.17	61.53	39.19	28.29	60.62	46.55	25.98	20.55

Table 13: Multi-modal fusion comparisons.

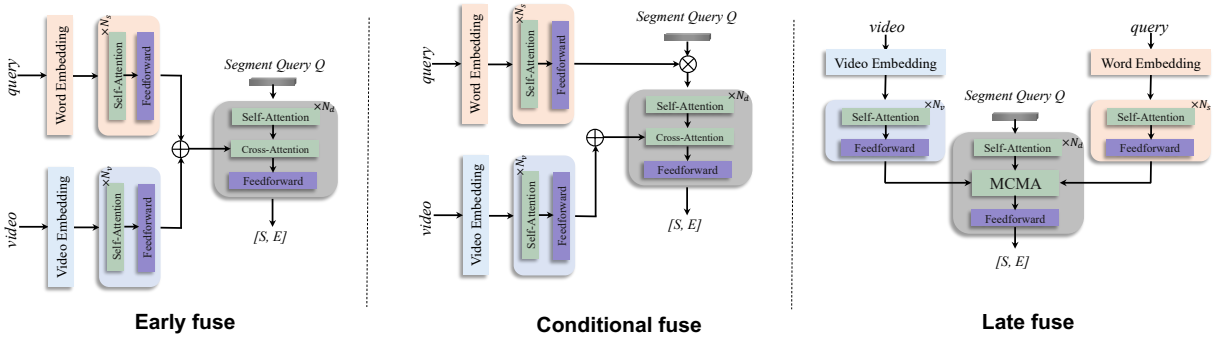

Figure 7: Three methodologies of multi-modal fusion.

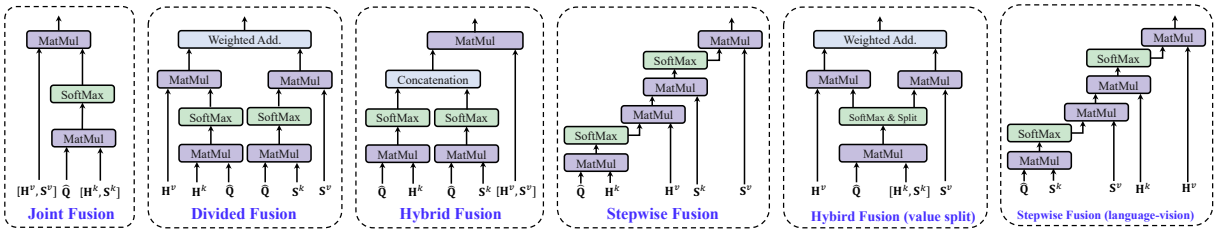

Figure 8: Modal variants of Multi-head Cross-Modal Attention module (MCMA). The first fourth variants are already presented in the main paper and the rightmost two are two variants for Hybrid Fusion and Stepwise Fusion respectively.

Models	ActivityNet Captions						Charades-STA				TACoS					
	$R_1^{0.3}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.3}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.5}$	$R_1^{0.7}$	$R_5^{0.5}$	$R_5^{0.7}$	$R_1^{0.1}$	$R_1^{0.3}$	$R_1^{0.5}$	$R_5^{0.1}$	$R_5^{0.3}$	$R_5^{0.5}$
GTR-B (Ours)	**66.15**	**49.67**	**28.45**	**87.94**	**79.83**	**64.34**	**62.45**	**39.23**	**91.40**	**61.76**	**48.85**	**39.34**	**28.34**	**72.59**	**60.85**	**46.67**
GTR-L (Ours)	**66.72**	**50.43**	**28.91**	**88.21**	**80.22**	**64.95**	**62.51**	**39.56**	**91.62**	**61.97**	**49.25**	**39.93**	**29.21**	**73.14**	**61.22**	**47.10**
GTR-H (Ours)	**66.80**	**50.57**	**29.11**	**88.34**	**80.43**	**65.14**	**62.58**	**39.68**	**91.62**	**62.03**	**49.41**	**40.39**	**30.22**	**73.24**	**61.94**	**47.73**

Table 14: Performance comparisons on three benchmarks(%).