MomentDiff: Generative Video Moment Retrieval from Random to Real (Supplementary Material)

Anonymous Author(s)
Affiliation
Address
`email`

This supplementary material provides more details of our MomentDiff framework:

1. Implementation details.
2. Inference efficiency of MomentDiff.
3. More experiment results.
4. Broader impacts.

1 Implementation details

1.1 Datasets

Public datasets. Charades-STA [1] serves as a benchmark dataset for the video moment retrieval task and is built upon the Charades dataset, originally collected for video action recognition and video captioning. The Charades-STA dataset comprises 6,672 videos and 16,128 video-query pairs, allocated for training (12,408 pairs) and testing (3,720 pairs). On average, the videos in this dataset have a duration of 29.76 seconds. Each video is annotated with an average of 2.4 moments, with each moment lasting approximately 8.2 seconds. **QVHighlights** [2] contains 10,148 videos, each 150 seconds long and annotated with at least one text query describing its relevant content. These videos are from three main categories, daily vlogs, travel vlogs, and news events. On average, there are approximately 1.8 non-overlapping moments per query, annotated on 2s non-overlapping clips. The dataset contains a total of 10,310 queries with 18,367 annotated moments. The training set, validation set and test set include 7,218, 1,550 and 1,542 video-text pairs, respectively. **TACoS** [3] is compiled specifically for video moment retrieval and dense video captioning tasks. It is comprised of 127 videos that depict cooking activities, with an average duration of 4.79 minutes. TACoS contains a total of 18,818 video-query pairs. In comparison to the Charades-STA dataset, TACoS has more video segments that are temporally annotated with queries per video. On average, each video contains 148 queries. Additionally, the TACoS dataset is known for its difficulty, as the queries it contains are limited to only a few seconds or even just a few frames. To ensure impartial comparisons, we use the same dataset split [4], which consists of 10,146, 4,589, and 4,083 video-query pairs for the training, validation, and testing sets, respectively.

Table 1: Training and test sets on two anti-bias datasets.

Dataset	Charades-STA-Len			Charades-STA-Mom		
	$\boldsymbol{w}_0 \leq 10s$	$\boldsymbol{w}_0 > 10s$	Total	$\boldsymbol{c}_0 + \boldsymbol{w}_0/2 \leq 15s$	$\boldsymbol{c}_0 - \boldsymbol{w}_0/2 > 15s$	Total
Training	9307	2326	11633	5330	1332	6662
Test	197	788	985	259	1038	1297

Anti-bias datasets. To explore the location bias problem, we construct two anti-bias datasets with location distribution shifts based on the Charades-STA dataset. In video moment retrieval, length and position are important parameters of spans.

Therefore, we first investigate the effect of span length $\boldsymbol{w}_0$. As shown in Tab. 1, in the training set of Charades-STA-Len, we collect 9,307 video-text pairs with span length $\boldsymbol{w}_0 \leq 10s$ and 2,326 video-text pairs with $\boldsymbol{w}_0 > 10s$, accounting for 80% and 20% of the total training set. In contrast, in the test set, we select 197 video-text pairs with $\boldsymbol{w}_0 \leq 10s$ and 788 video-text pairs with $\boldsymbol{w}_0 > 10s$, accounting for 20% and 80% of the total test set.

Then, we design the dataset Charades-STA-Mom based on the span's end time $\boldsymbol{c}_0 + \boldsymbol{w}_0/2$ and start time $\boldsymbol{c}_0 - \boldsymbol{w}_0/2$. In the training set of Charades-STA-Mom, we collect 5,330 video-text pairs with $\boldsymbol{c}_0 + \boldsymbol{w}_0/2 \leq 15s$ and 1,332 video-text pairs with $\boldsymbol{c}_0 - \boldsymbol{w}_0/2 > 15s$, accounting for 80% and 20% of the total training set. In contrast, in the test set, we select 259 video-text pairs with $\boldsymbol{c}_0 + \boldsymbol{w}_0/2 \leq 15s$ and 788 video-text pairs with $\boldsymbol{c}_0 - \boldsymbol{w}_0/2 > 15s$, accounting for 20% and 80% of the total test set.

Algorithm 1: MomentDiff Training in a PyTorch-like style.

```
# Video features:  v_feats $\in \mathbb{R}^{N_v \times D}$
# Text features:  t_feats $\in \mathbb{R}^{N_t \times D}$
# Ground truth spans:  gt_spans : $[*, 2]$
# alpha_cumprod(m):  cumulative product of $\alpha_i$
# Fully-connected layer:  FC()
# Video Moment Denoiser:  VMD()
def train(v_feats, t_feats, gt_spans):
   # Similarity-aware Condition Generator
   f_feats = SCG(v_feats, t_feats)  #  $N_v \times D$

   # Span normalization
   ps = pad_spans(gt_spans)  # Pad gt_spans to $[N_r, 2]$
   ps = (ps * 2 -1) * $\lambda$  # Signal scaling
   m = randint(0, M)  # Noise intensity
   noi = normal(mean=0, std=1) # Noise
   ps_m = sqrt( alpha_cumprod(m)) * ps +
           sqrt(1 - alpha_cumprod(m)) * noi  # Noisy span
   ps_m = (ps_m/$\lambda$ +1)/2  # Normalization

   # Span embedding
   ps_emb_m = FC(ps_m)

   # Intensity-aware attention
   output_m = VMD(ps_emb_m, m, f_feats)  # Output embedding

   # Denoising training
   hat_ps_m = Span_pred(output_m)  # Predicted span
   hat_cs_m = Score_pred(output_m)  # Confidence score
   # Computing loss
   loss = loss_sim(f_feats) + loss_vmr(hat_ps_m, hat_cs_m, gt_spans)

   return loss
```

1.2 Pseudo Code of MomentDiff

Algorithm 1 provides the pseudo-code of MomentDiff Training in a PyTorch-like style.

The inference procedure of MomentDiff is a denoising sampling process from noise to temporal spans. Starting from spans sampled in Gaussian distribution, the model progressively refines its predictions, as shown in Algorithm 2.

Algorithm 2: MomentDiff inference in a PyTorch-like style.

```
# Video features:  v_feats $\in \mathbb{R}^{N_v \times D}$
# Text features:  t_feats $\in \mathbb{R}^{N_t \times D}$
# Video Moment Denoiser:  VMD()
def test(v_feats, t_feats, sampling_num):
   # Similarity-aware Condition Generator
   f_feats = SCG(v_feats, t_feats)  #  $N_v \times D$

   # Noisy span:$[N_r, 2]$
   ps_m = normal(mean=0, std=1)

   # uniform sample
   intensity = reversed(linespace(-1, M, sampling_num))
   # [(M-1, M-2), (M-2, M-3), ..., (1, 0), (0, -1)]
   intensity_pairs = list(zip(intensity[:-1], intensity[1:])

   for intensity_now, intensity_next in zip(intensity_pairs):
      # predict ps_0 from ps_m
      output_m = VMD(ps_m, f_feats, intensity_now)
      # Predicted span
      hat_ps_m = Span_pred(output_m)
      # Update ps_m
      ps_m = ddim_update(hat_ps_m, ps_m, intensity_now, intensity_next)

   # Confidence score
   hat_cs_m = Score_pred(output_m)

   return hat_ps_m, hat_cs_m
```

Table 2: The inference time of 2DTAN [5], MMN [6], MomentDETR [2] and MomentDiff on Charades-STA with VGG video features and Glove text features. We report R1@0.5, R1@0.7 and MAP$_{avg}$. Default settings are marked in blue.

Method	Charades-STA			Inference time
	R1@0.5	R1@0.7	MAP$_{avg}$	(second)
2DTAN [5]	41.34	23.91	29.26	42.18
MMN [6]	46.93	27.07	31.58	53.42
MomentDETR [2]	50.54	28.01	29.87	12.42
MomentDiff (Step=1)	49.17	26.39	29.12	7.56
MomentDiff (Step=2)	50.81	27.84	31.27	8.23
MomentDiff (Step=10)	**52.36**	28.08	**31.75**	11.01
MomentDiff (Step=50)	51.94	28.25	31.66	20.74
MomentDiff (Step=100)	52.21	**28.84**	31.01	34.35

2 Inference Efficiency of MomentDiff

Inference time. Inference efficiency is critical for machine learning models. We test 2DTAN [5], MMN [6], MomentDETR [2] and MomentDiff on the Pytorch framework [7] in Tab. 2. We test all models with one NVIDIA Tesla A100 GPU.

Compared with 2DTAN [5] and MMN [6], MomentDiff (Step=1) not only achieves the best results on recall, but also improves the inference speed by 5-7 times. This is because 2DTAN and MMN predefine a large number of proposals, which may be redundant and increase computational overhead. Compared to MomentDETR [2], MomentDiff (Step=10) achieves better results with similar inference time. The possible reasons are that we adopt fewer random spans, very simple network structures, and avoid post-processing.

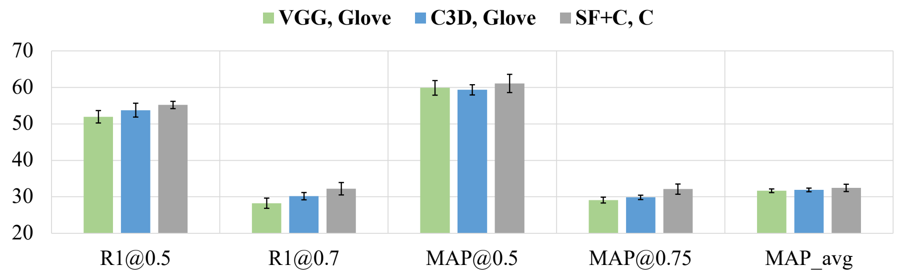

Figure 1: Performance fluctuations (%) corresponding to different features and multiple random seeds on the Charades-STA dataset.

Table 3: Ablation study (%) on the Charades-STA dataset with SF+C video features and CLIP text features. We report R1@0.5, R1@0.7 and MAP$_{avg}$. Default settings are marked in blue.

(a) Different sampling strategies.

Sampling	R1@0.5	R1@0.7	MAP$_{avg}$
DDPM [8]	**55.62**	32.37	32.48
DDIM [9]	55.57	**32.42**	**32.85**

(b) Effect of the schedule of β.

Schedule	R1@0.5	R1@0.7	MAP$_{avg}$
Linear	54.76	31.43	31.59
Cosine	**55.57**	**32.42**	**32.85**

(c) Effect of the Box Renewal [10].

Schedule	R1@0.5	R1@0.7	MAP$_{avg}$
w/o Box Renewal	55.57	32.42	32.85
w/ Box Renewal	**56.03**	**32.64**	**33.18**

(d) Effect of the batch size.

batch size	R1@0.5	R1@0.7	MAP$_{avg}$
16	52.42	30.81	30.14
32	**55.57**	**32.42**	**32.85**
64	53.78	32.25	31.93

(e) Effect of the number of spans on R1@0.5.

train \ test	1	3	5	10	20
1	50.83	50.91	50.97	50.87	50.82
3	53.98	54.12	54.19	54.23	54.21
5	55.36	55.41	55.57	**55.69**	55.51
10	53.71	53.84	53.86	53.89	53.93
20	53.16	53.38	53.40	53.44	53.42

3 More Experiment Results

3.1 Error bars

Fig. 1 shows the performance fluctuation of the model on the Charades-STA dataset. We use different random seeds (seed= 2023, 2022, 2021, 2020, 2019) and different features (VGG, Glove; C3D, Glove; SF+C, C;) to organize experiments. This shows that the model always converges and achieves stable results for different initializations. This phenomenon demonstrates the ability of the model to learn to generate real spans from arbitrary random spans.

3.2 Ablation study

Different sampling strategies. Denoising Diffusion Probabilistic Models (DDPM) [8] and Denoising Diffusion Implicit Models (DDIM) [9] are popular and classic diffusion models. We show the results of both strategies in Tab. 3(a). We find that DDIM and DDPM perform similarly, but DDIM samples faster. Therefore we adopt DDIM as the default technology.

Effect of the schedule of β. The schedule of β determines the weighting ratio of different intensity noises. In Tab. 3(b), we find that the cosine schedule works better in our experiment. The cosine schedule makes the noisy spans change slowly at the beginning and end during the diffusion process, and the generation effect is more stable. So we set the cosine schedule as the default.

Effect of the Box Renewal. Box Renewal is a post-processing technique in DiffusionDet [10]. Tab. 3(c) shows that Box Renewal can indeed slightly improve the results. To keep the inference process as simple as possible, we do not use Box Renewal by default.

Effect of the batch size. As shown in Tab. 3(d), we set the batch size to 32 to achieve the best results.

Effect of the number of spans on R1@0.5. Our training and inference are decoupled. Our simple framework allows us to input any number of random noises. Tab. 3(e) shows that there is a slight improvement when testing with more noise boxes.

4 Broader Impacts

First, our work does not involve private data. Second, we believe that AI is a double-edged sword, and our model is no exception. For example, when users or websites use our model, only natural language is needed to locate video moments and collect desired video material, which improves the productivity of society. However, this may have a negative impact if the natural language entered by the user contains words related to violence, pornography, etc. We will consider these scenarios and implement a more secure VMR model.

References

[1] Gao, J., C. Sun, Z. Yang, et al. Tall: Temporal activity localization via language query. In *ICCV*, pages 5267–5275. 2017.

[2] Lei, J., T. L. Berg, M. Bansal. Qvhighlights: Detecting moments and highlights in videos via natural language queries. In *NeurIPS*. 2021.

[3] Regneri, M., M. Rohrbach, D. Wetzel, et al. Grounding action descriptions in videos. *TACL*, 1:25–36, 2013.

[4] Zeng, R., H. Xu, W. Huang, et al. Dense regression network for video grounding. In *CVPR*, pages 10287–10296. 2020.

[5] Zhang, S., H. Peng, J. Fu, et al. Learning 2d temporal adjacent networks for moment localization with natural language. In *AAAI*, pages 12870–12877. 2020.

[6] Wang, Z., L. Wang, T. Wu, et al. Negative sample matters: A renaissance of metric learning for temporal grounding. In *AAAI*, pages 2613–2623. 2022.

[7] Paszke, A., S. Gross, F. Massa, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[8] Ho, J., A. Jain, P. Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*. 2020.

[9] Song, J., C. Meng, S. Ermon. Denoising diffusion implicit models. In *ICLR*. 2021.

[10] Chen, S., P. Sun, Y. Song, et al. Diffusiondet: Diffusion model for object detection. abs/2211.09788, 2022.