

Supplementary Materials: Rethinking the Architecture Design for Efficient Generic Event Boundary Detection

Anonymous Authors

1 ARCHITECTURE DETAILS AND TRAINING SETTINGS

This section detailed introduce the architecture of the different components of the proposed BasicGEBD and EfficientGEBD. We also provide how to transform the boundary detection task to a binary classification one, based on which we further show how to train the model.

1.1 Architecture details

The overview architecture of the proposed BasicGEBD and EfficientGEBD are shown in Figure 1 (a) and (b), respectively. We first describe the overall of EfficientGEBD based on inference procedure Figure 1 (c), then more details about the architecture towards four different components will be provided in the following.

Figure 1 (c) illustrates the detailed overall architecture. Given a video sequence with T frames, we aim to detect the event boundaries. The backbone model will be first used to extract the features from the input video and generate the corresponding features at different layers. For example, the features from the layer-1 will be denoted as $\mathbf{V}_1 \in \mathbb{R}^{T \times C_1 \times H_1 \times W_1}$, where the C_1 , H_1 and W_1 are the number of channels, and spatial resolutions, respectively. Then an AvgPooling layer will squeeze the \mathbf{V}_1 to the dimension of $T \times C_1 \times 1$, and then a fully connected linear projection (FC) will be used to transform the C_1 to C , resulting in the features of \mathbf{R}_1 . We then concatenate all \mathbf{R}_i and unfold the features to generate T different small clips, where the length of each small clip is t_w . In our experiments, $t_w = 17$ is used. Then the boundary detection problem can be transformed to the binary classification problem identifying if the median frame of the small clip is a boundary. After we have all predictions for all timestamps, we can collect them to generate the final predictions, where the boundary can be explored using a given threshold ϵ .

During training, all these final predictions will be considered as binary classification results, which can be trained by the cross-entropy loss effectively.

All the backbone models used in our research follow the design in the original paper [6, 19]. As we described before, the output features from different layers will be processed by an AvgPool Layer and an FC projection. In our experiments, the C is set as 512.

Besides the encoder we mentioned in the main paper, we also test its variants, which are shown in Figure 2 (a,b,c,d). These results can be found in Table 2. Moreover, Figure 2 (e,f) provide the detailed architecture of the fusion model. We use $r = 4$ in for the Cross Att. in our experiments.

We also provide the details about how the similarity maps are generated in our methods. Specifically, we calculate the frame-level pairwise similarity as below:

$$S_t(i, j) = \text{Sim}(\mathbf{D}_i^t, \mathbf{D}_j^t), \quad i, j \in [1, \dots, t_w], \quad t \in [1, \dots, T], \quad (1)$$

where $S_t \in \mathbb{R}^{n_l \times t_w \times t_w}$ is the similarity map of the features from encoder, \mathbf{D}_i^t and \mathbf{D}_j^t are features inside the given snippet and Sim is the cosine similarity function.

As for the decoder, a ResNet10 has 4 residual layers, where each layer contains 1 residual block, consisting of two convolution layers with residual connection is used. We omit the final FC of ResNet10 and the dimensionality of the output features from the decoder is $C \times t_w$.

1.2 Training details.

1.2.1 Gaussian Smoothing. As mentioned before, we follow the continuous paradigm that each frame-wise prediction is merged to obtain the final predictions with length T . Since the annotations for GEBD are subjective and ambiguous, directly using these hard labels to optimize the network could lead to poor generalization ability. Therefore, we smoothed the one-hot labels with a Gaussian kernel to generate soft labels $\tilde{\mathbf{y}}$. The window size of the Gaussian kernel remains the same as the length of the video snippet and $\sigma = 1$ in all experiments.

1.3 Loss Function.

After obtaining the predictions from detectors, we calculate the binary cross entropy loss for each one. The total training loss can be written as:

$$\mathcal{L} = \sum_n^N [-\tilde{\mathbf{y}}_n \log(\mathbf{p}_n) - (1 - \tilde{\mathbf{y}}_n) \log(1 - \mathbf{p}_n)], \quad (2)$$

where \mathbf{p}_n is the collection of all predictions for T , N is the number of all training samples.

2 BASELINE MODEL ANALYSIS

Table 1: The architectures of three representative GEBD methods and the propose models in this paper.

Mehtods	Backbone	Encoder	Sim. Map	Decoder	Fusion
SboCo [10]	ResNet50	1d-Conv	CosSim. or L2-Sim.	ResNet + Transformer	-
DDM-Net [18]	ResNet50	Differences	L2-Sim.	FCN	Pro. Att.
SC-Trans. [12]	ResNet50	Transformer	CosSim.	FCN	-
LightGEBD [5]	X3D-XS	1d-Conv	CosSim.	Transformer	-
BasicGEBD	ResNet50	1d-Conv	CosSim.	FCN	-

SBoCo [10], DDM-Net [18], and SC-Transformer [12] (SC-Trans.), aside with the LightGEBD [5] are selected as the representative supervised GEBD networks. We here detailed state how their architectures share strong similarities.

For backbone networks, expect the recently proposed LightGEBD using a video backbone, X3D [4], all the rest use the ResNet50 [6] as the backbone for feature extraction.

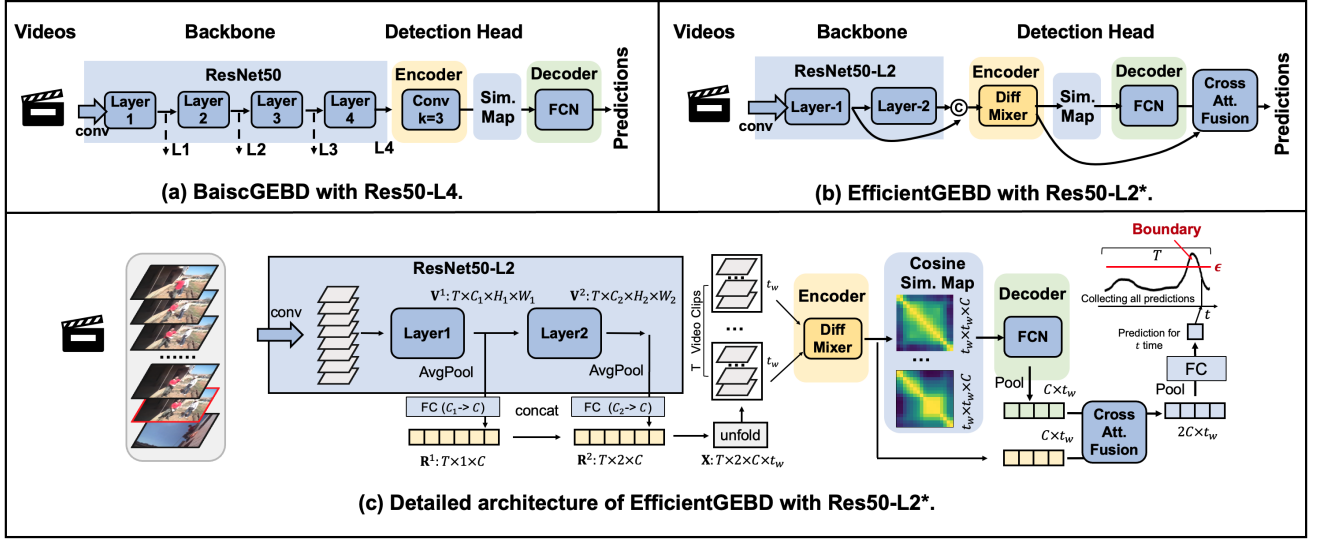


Figure 1: Overview of the proposed BasicGEBD (a) and EfficientGEBD (b). We further proposed the detailed architectures of EfficientGEBD with ResNet50-L2* in (c).

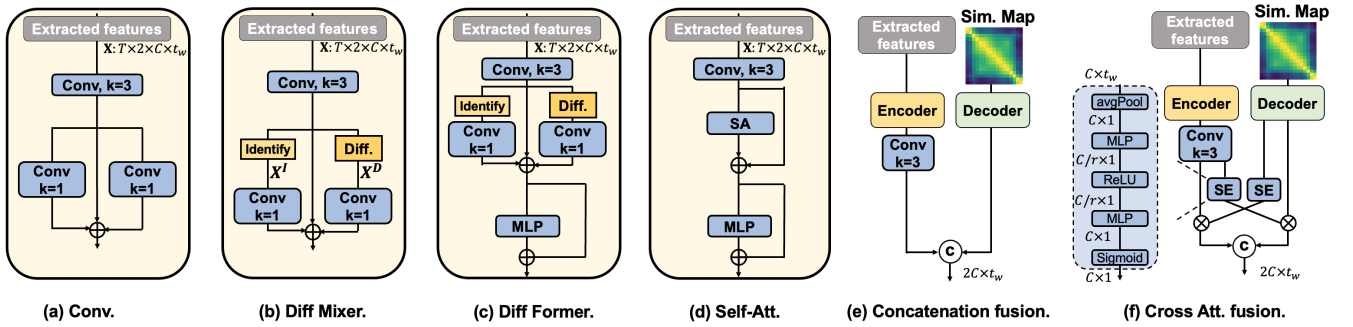


Figure 2: The illustrations of the encoder (a) Conv encoder, (b) Diff Mixer encoder, (c) Diff former encoder, (d) self-Attention encoder. The fusion module with concatenation fusion (e) and cross attention fusion (f).

We also find that all these methods contain an encoder module to process the extracted features from backbones. The LightGEBD and SBoCo directly use a 1-d Conv layer for encoding. While SC-Transformer selects a more complex encoder built based on the Transformer architecture. DDM-Net uses the dense difference map for encoding the extracted features. Although with different formations, an encoder seems a necessary component for all these GEBD models.

The implementation of the similarity maps as well as the 2-d FCN decoder can be seen as a sign of recent outperforming supervised GEBD methods. DDM-Net selects the L2-norm of the difference maps to represent the similarity maps, which show an inverse ratio to that using Cosine similarity. We see that the rest of them all use Cosine similarity. The recent research [12] further investigates the effectiveness of different similarity metrics, including Chebyshev, Manhattan, L2-norm, and Cosine, where the Cosine similarity shows the best performance in GEBD tasks.

The fusion module is the unique one proposed in DDM-Net which fuses the features from the decoder and encoder to enhance the motion cues for GEBD models. In our research, we find that using a small fusion module can bring significant performance improvement for GEBD.

From the about analysis, we can abstract the general architecture of a GEBD model, which contains: (1) The backbone for feature extraction; (2) The encoder for temporal modeling; (3) The similarity map (Sim. Map); (4) The decoder processing the similarity map. As only DDM-Net applies the fusion module, we do not use it in our baseline model.

3 FULLY ROADMAP RESULTS

In this section, we provide a full trajectory going from a basic GEBD model to an efficient GEBD model. The results shown in the main paper only select some of our milestone results due to the limitation of pages. The complete results are shown in Table 2. In the Table,

different study sections are marked by different colors. Moreover, we denote the EfficientGEBD with low efficiency by red, which is not considered as the outcome of our exploration. We only use these results for architecture studies.

As we have stated in the main paper, using a large backbone network does not significantly improve the performance of BasicGEBD in GEBD tasks. We also show that such a scenario still happens but not as severe as that in BasicGEBD when we use EfficientGEBD. For example, the BasicGEBD with ResNet18-L4* can already achieve a high detection performance of 78.2%, while the use of ResNet152-L4* only increasing the performance by 0.7% while largely affecting the efficiency of the model. However, 0.7% is still larger than the performance gain when using ResNet152 and ResNet18 as the backbones for BasicGEBD, demonstrating the effectiveness of our design and further showing the importance of introducing additional temporal modeling ability for GEBD models. This also confirms the correctness of using video backbone to conduct the spatiotemporal feature learning throughout the whole architecture.

4 VISUALIZATION OF SIMILARITY MAPS

This section shows the effectiveness of using the difference of EfficientGEBD when detecting different kinds of boundaries. Actually, using a difference map design meets our intuition in boundary detection tasks: To detect these motion-related boundaries, motion information plays a principal role in perceiving temporal variations and can be effectively modeled by using feature differences at different timestamps. The results are shown in Figure 3.

We show the changes of the feature norms at different timestamps, and further present examples with t_w frames and visualize their pairwise similarity maps of $t_w \times t_w$ captured by detectors. The red lines present the ground-truth boundaries. The pairwise similarity maps are then sent to the contrast module to amplify the discriminatives. Therefore, the diagonal pattern (similarity within the same side of frame groups and dissimilarity between groups) is essential to maximizing the boundary information as discussed in [10]. From the results, we can see that samples like shot changes have distinctive background changes rather than the main objects. These low appearance-level features can be preserved in the original extracted features, X^I . Moreover, as these characteristics also have fewer temporal dependencies, the discriminative information is not clear in the similarity maps of X^D . However, samples like event-level changes can contain complex temporal change patterns, resulting in ambiguous boundaries in the map of X^I . The crucial discriminative information is found mainly in the map of X^D . Overall, the visualizations demonstrate the demands using the differences map of the proposed EfficientGEBD. We also found that using difference maps might not always obtain the discriminative information in similarity map.

5 VISUALIZATION OF DISTRACTION ISSUE

We hypothesize that conducting the spatiotemporal representation learning in such a greedy way can lead to several inefficiency issues in GEBD tasks. As the image domain backbones are usually designed to identify the main objects in an image, learning the spatial features without the guide from temporal information can result in the attention of the backbone distracting from the objects

Table 2: The complete results on Kinetic-GEBD during our architecture ablation.

Method	F1@0.05	GFLOPs	FPS
SC-Transformer	77.7	10.36	971
DDM-Net	76.4	46.52	39
BasicGEBD (Res50-L4)	77.1	4.36	1562
BasicGEBD (Res50-L3)	77.0	3.57	1783
BasicGEBD (Res50-L2)	76.8	2.08	2325
BasicGEBD (Res50-L1)	75.3	1.05	2699
BasicGEBD (Res34-L2)	76.6	1.94	2495
BasicGEBD (Res34-L4)	77.0	3.92	2386
BasicGEBD (Res18-L2)	76.1	1.24	2480
BasicGEBD (Res18-L4)	77.2	2.07	2380
BasicGEBD (Res152-L2)	76.3	2.94	1783
BasicGEBD (Res152-L4)	77.2	11.77	847
Conv	76.4	2.09	2257
Diff Mixer x1	77.0	2.09	2257
Diff former x1	76.3	2.09	2257
Diff Mixer x2	76.9	2.09	2257
Diff former x2	76.2	2.09	2257
Cat. fuse	77.6	2.09	2208
Cross Att.	77.7	2.09	2208
no 2d-FCN	64.4	1.85	2426
FCN-Res10	77.7	2.09	2208
FCN-Res18	77.8	2.42	2181
EfficientGEBD (Res18-L2*)	77.3	1.28	2348
EfficientGEBD (Res18-L4*)	78.2	2.11	2384
EfficientGEBD (Res34-L2*)	77.8	1.97	2352
EfficientGEBD (Res34-L4*)	78.5	3.96	2376
EfficientGEBD (Res50-L2*)	78.3	2.10	2097
EfficientGEBD (Res50-L4*)	78.7	4.39	1541
EfficientGEBD (Res152-L2*)	77.9	2.97	1692
EfficientGEBD (Res152-L4*)	78.9	11.80	816
EfficientGEBD (CSNR50-L2*)	79.7	1.57	2281
EfficientGEBD (CSNR50-L4*)	81.1	2.83	1748
EfficientGEBD (CSNR152-L2)	80.4	1.98	2029
EfficientGEBD (CSNR152-L4)	82.0	6.37	1050
EfficientGEBD (CSNR152-L2*)	80.6	2.00	1215
EfficientGEBD (CSNR152-L4*)	82.9	6.40	1025
BasicGEBD (CSNR50-L4)	81.6	2.79	1889
BasicGEBD (CSNR152-L4)	82.5	6.36	1054

most related to the boundaries, and getting stuck in some areas containing the other objects in each frame. We refer to such an issue as *distraction issue*.

This section provides more visualization results using Grad-CAM++ [1] in Figure 4. The event boundary in (a) is defined by the changes of action during arm wrestling. The high activations of the ResNet backbone get stuck in the spatial areas that contain the head of the person in the center of the frame. With the arm wrestling-related spatial areas missing features extracted from the backbone, the subsequent modules will have difficulties conducting the following temporal modeling, resulting in the failure detection of this boundary.

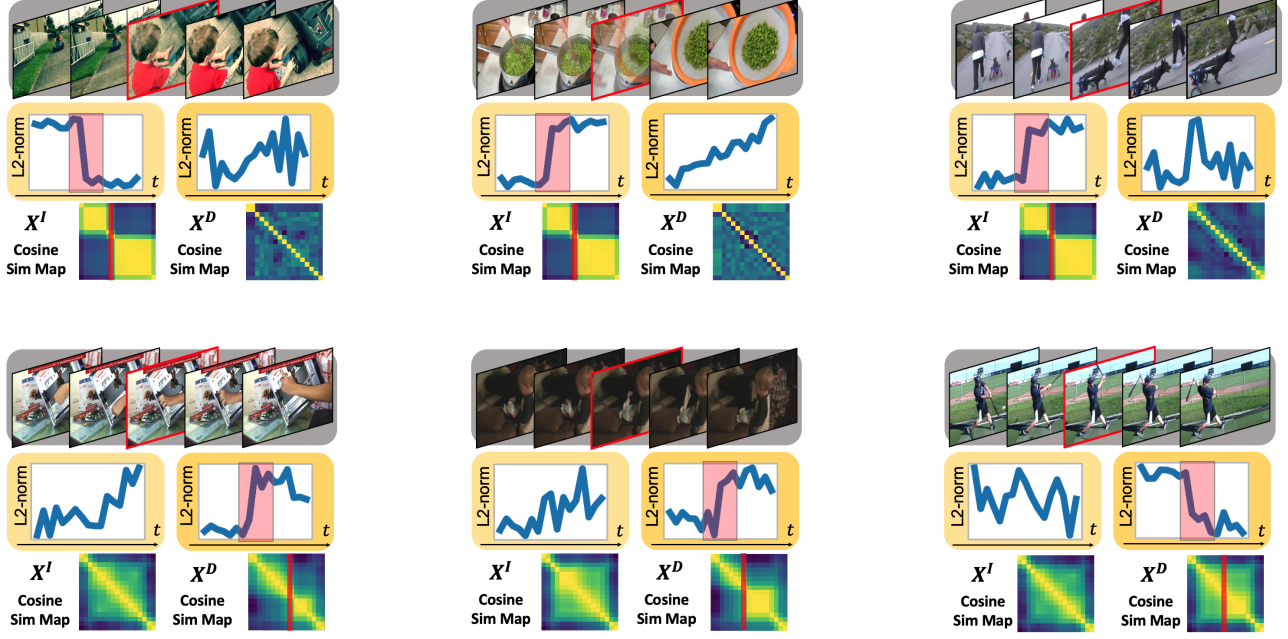


Figure 3: In the figure, we calculate the L2-norm and the cosine similarity map of the features at different timestamps to see whether the discriminative boundary features can be captured.

Table 3: Comparisons in terms of F1 score (%) on Kinetics-GEBD with Rel.Dis. threshold from 0.05 to 0.5.

Method	Backbone	F1 @ Rel. Dis.										
		0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	avg
BMN [14]	ResNet50	18.6	20.4	21.3	22.0	22.6	23.0	23.3	23.7	23.9	24.1	22.3
BMN-StartEnd [14]	ResNet50	49.1	58.9	62.7	64.8	66.0	66.8	67.4	67.8	68.1	68.3	64.0
TCN-TAPOS [11]	ResNet50	46.4	56.0	60.2	62.8	64.5	65.9	66.9	67.6	68.2	68.7	62.7
TCN [11]	ResNet50	58.8	65.7	67.9	69.1	69.8	70.3	70.6	70.8	71.0	71.2	68.5
PC [16]	ResNet50	62.5	75.8	80.4	82.9	84.4	85.3	85.9	86.4	86.7	87.0	81.7
PC+OF [16]	ResNet50	62.5	75.8	80.4	82.9	84.4	85.3	85.9	86.4	86.7	87.0	81.7
SBoCo [10]	ResNet50	73.2	-	-	-	-	-	-	-	-	-	86.6
Temporal Perceiver [17]	ResNet50	74.8	82.8	85.2	86.6	87.4	87.9	88.3	88.7	89.0	89.2	86.0
CVRL [13]	ResNet50	74.3	83.0	85.7	87.2	88.0	88.6	89.0	89.3	89.6	89.8	86.5
CVRL+ [20]	ResNet50	76.8	84.8	87.2	88.5	89.2	89.6	89.9	90.1	90.3	90.6	87.7
DDM-Net [18]	ResNet50	76.4	84.3	86.6	88.0	88.7	89.2	89.5	89.8	90.0	90.2	87.3
SC-Transformer [12]	ResNet50	77.7	84.9	87.3	88.6	89.5	90.0	90.4	90.7	90.9	91.1	88.1
BasicGEBD	ResNet50	76.8	83.4	85.7	87.1	87.9	88.5	88.8	89.1	89.4	89.6	86.6
EfficientGEBD	ResNet50	78.3	85.1	87.4	88.7	89.6	90.1	90.5	90.8	91.1	91.3	88.3
SBoCo [10]	TSN	78.7	-	-	-	-	-	-	-	-	-	89.2
CLA [9]	TSN	79.1	-	-	-	-	-	-	-	-	-	-
CASTANet [7]	CSN	78.1	-	-	-	-	-	-	-	-	-	-
CVRL [13]	CSN	78.6	-	-	-	-	-	-	-	-	-	-
CVRL+ [20]	CSN	81.2	-	-	-	-	-	-	-	-	-	-
BasicGEBD	CSN	82.5	87.7	89.6	90.7	91.4	91.9	92.2	92.4	92.6	92.8	90.4
EfficientGEBD	CSN	82.9	87.9	89.7	90.9	91.5	92.0	92.3	92.6	92.8	93.0	90.5

Table 4: Comparison with others in terms of F1 score (%) on TAPOS with Rel.Dis. threshold from 0.05 to 0.5 with 0.05 interval.

Method	F1 @ Rel. Dis.										
	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	avg
ISBA [3]	10.6	17.0	22.7	26.5	29.8	32.6	34.8	36.9	38.2	39.6	30.2
TCN [11]	23.7	31.2	33.1	33.9	34.2	34.4	34.7	34.8	34.8	34.8	64.0
CTM [8]	24.4	31.2	33.6	35.1	36.1	36.9	37.4	38.1	38.3	38.5	35.0
TransParser [15]	28.9	38.1	43.5	47.5	50.0	51.4	52.7	53.4	54.0	54.5	47.4
PC [16]	52.2	59.5	62.8	64.6	65.9	66.5	67.1	67.6	67.9	68.3	64.2
Temporal Perceiver [17]	55.2	66.3	71.3	73.8	75.7	76.5	77.4	77.9	78.4	78.8	73.2
DDM-Net [18]	60.4	68.1	71.5	73.5	74.7	75.3	75.7	76.0	76.3	76.7	72.8
SC-Transformer [12]	61.8	69.4	72.8	74.9	76.1	76.7	77.1	77.4	77.7	78.0	74.2
EfficientGEBD (Res50-L3*)	62.6	70.1	73.4	75.6	76.7	77.2	77.5	77.9	78.1	78.4	74.7
EfficientGEBD (Res50-L4*)	63.1	70.5	73.7	75.9	76.9	77.4	77.6	78.0	78.2	78.6	74.8

For the child in (b), the issue is similar to that in (a), where the high activations of the ResNet backbone get stuck in the spatial areas that contain the head of the child, rather than the boundary-related spatial areas. These high activations around the head will harm the detection procedure of the swing action, where the swinging object and the arm should be what needs to be focused.

As the image domain backbone is usually pre-trained on the ImageNet dataset [2], where the objects at the center of the image can be viewed as the main objects of the image, using ResNet50 pre-trained on ImageNet can result in too many attention on the area contain the most identified features of the central object. Therefore, these image backbones focus on the head of the person in each frame. The results in (c) further confirm our findings. In such a case, the GEBD model will be distracted by the features most identified features of the central objects, resulting in the distraction issue.

6 ADDITIONAL EXPERIMENTAL RESULTS

6.1 Main results

We provide the full results with Rel.Dis. threshold from 0.05 to 0.5 with 0.05 interval on both Kinetics-GEBD and TAPOS. The results are shown in Table 3 and Table 4. Overall, we see that our methods achieve promising performance compared to previous methods in different Rel. Dis.

6.2 Hyper-parameter studies

We further provide the hyper-parameter studies of the proposed EfficientGEBD. All the experiments are conducted on the Kinetic-GEBD dataset using the EfficientGEBD with ResNet50-L2*.

Table 5: Study on window size.

#win t_w	0.05	avg.
11	77.7	88.2
17	78.3	88.3
23	77.9	88.2

Table 6: Study on T .

#inp T	0.05	avg.
80	76.4	87.8
100	78.3	88.3
120	78.3	88.0

Table 7: Performance with different thresholds.

ϵ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
F1 score	51.2	70.5	78.3	75.5	65.2	50.4	34.7	20.2	7.9

6.2.1 Window Size. We conduct ablations on different lengths of the local window size t_w for generating local small video clips processed by the encoder, as it is in Figure 1. The results are shown in Table 5. The larger window size t_w means a larger temporal respective field for boundary detection. However, as most boundaries are only relative to the local frames rather than long-term information, we see that increasing the window size does not necessarily bring performance growth with additional computations. Also, using small window sizes can slightly affect the performance of boundary detection.

6.2.2 Study on input length T . We also conduct ablations on input length T which represents the sampling frequency of a video. A larger T means a high sampling rate of the video. The local window size t_w is fixed to 17. From the results in 6, we can conclude that the performance tends to be saturated when T is larger than 100 (around 10fps in Kinetics-GEBD).

6.2.3 Study on the threshold ϵ . We also provide the study for our method with different detecting thresholds, ϵ . The results are provided in Table 7 with the local window size $t_w = 17$ and temporal length $T = 100$. From the results we see that the choice of thresholds can affect the final results, therefore we should select it during the inference.

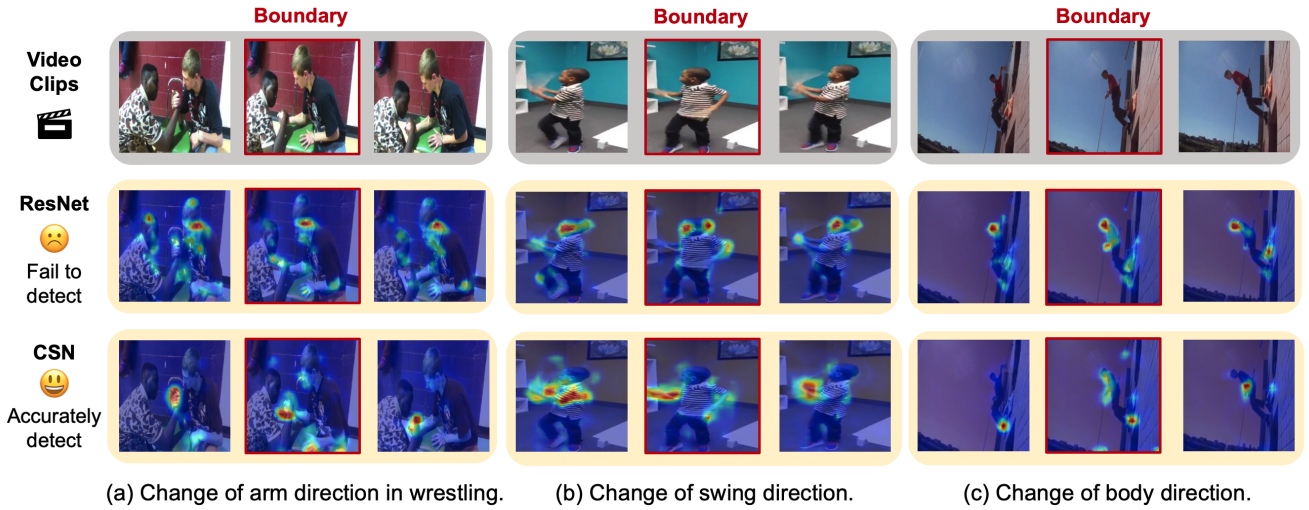


Figure 4: The activations captured by GradCAM++ [1] using ResNet [6] and CSN [19] as the backbones for GEBD models. The median frame is the boundary frame.

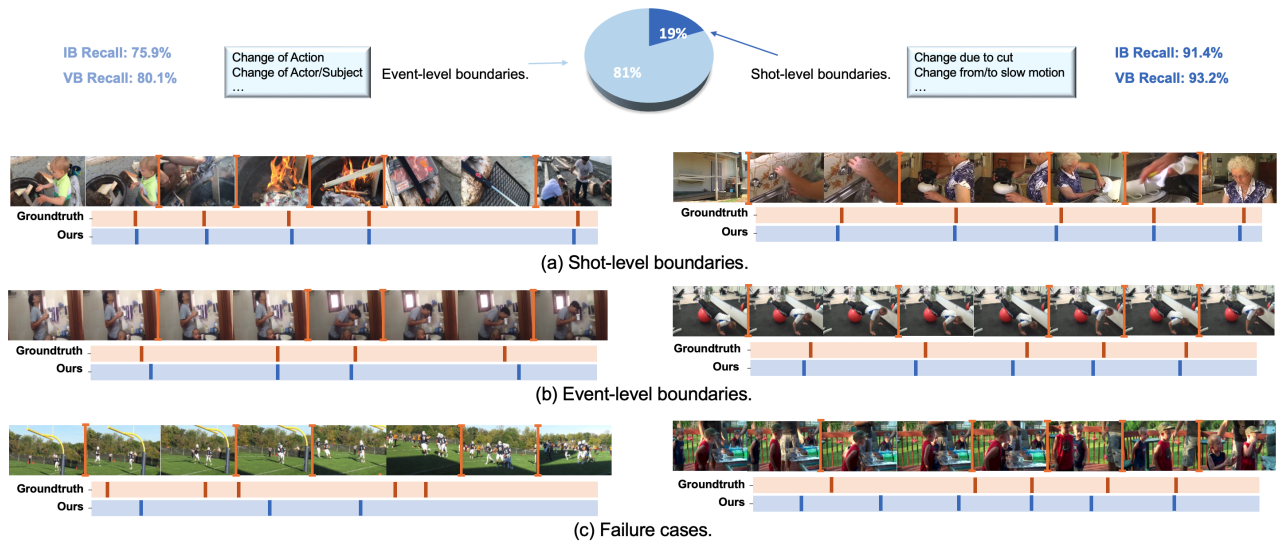


Figure 5: Detection results on Kinetics-GEBD for (a) shot-level changes, (b) event-level changes and (c) failure cases.

6.3 More visualizations

In this subsection, we provide more visualization results for the proposed EfficientGEBD in GEBD tasks.

The provided qualitative results of shot- and event-level boundary detection on Kinetics-GEBD in Figure 5 show the effectiveness of our method. We see that most predictions of our method are accurate.

Here we further provide more additional failure cases of our method. From the illustrations, we see that most of the failure cases share a similar property: Containing multiple objects with a complex sense. This meets our intuition since the changes of each object can be viewed as the boundary, which might distract the

GEBD model and increase the complexity of detecting the event boundaries.

REFERENCES

- [1] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE Winter Conf. App. Comput. Vis. IEEE*, 839–847.
- [2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog. Ieee*, 248–255.
- [3] Li Ding and Chenliang Xu. 2018. Weakly-supervised action segmentation with iterative soft boundary assignment. In *IEEE Conf. Comput. Vis. Pattern Recog.* 6508–6516.

- [4] Christoph Feichtenhofer. 2020. X3d: Expanding architectures for efficient video recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.* 203–213.
- [5] Sourabh Vasant Gothe, Vibhav Agarwal, Sourav Ghosh, Jayesh Rajkumar Vachhani, Pranay Kashyap, and Barath Raj Kandur Raja. 2024. What’s in the Flow? Exploiting Temporal Motion Cues for Unsupervised Generic Event Boundary Detection. In *IEEE Winter Conf. App. Comput. Vis.* 6941–6950.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.* 770–778.
- [7] Dexiang Hong, Congcong Li, Longyin Wen, Xinyao Wang, and Libo Zhang. 2021. Generic event boundary detection challenge at CVPR 2021 technical report: Cascaded temporal attention network (CASTANET). *arXiv preprint arXiv:2107.00239* (2021).
- [8] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. 2016. Connectionist temporal modeling for weakly supervised action labeling. In *Eur. Conf. Comput. Vis.* Springer, 137–153.
- [9] Hyolim Kang, Jinwoo Kim, Kyungmin Kim, Taehyun Kim, and Seon Joo Kim. 2021. Winning the CVPR’2021 Kinetics-GEBD Challenge: Contrastive Learning Approach. *arXiv preprint arXiv:2106.11549* (2021).
- [10] Hyolim Kang, Jinwoo Kim, Taehyun Kim, and Seon Joo Kim. 2022. Uboco: Unsupervised boundary contrastive learning for generic event boundary detection. In *IEEE Conf. Comput. Vis. Pattern Recog.* 20073–20082.
- [11] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. 2016. Segmental spatiotemporal cnns for fine-grained action segmentation. In *Eur. Conf. Comput. Vis.* Springer, 36–52.
- [12] Congcong Li, Xinyao Wang, Dexiang Hong, Yufei Wang, Libo Zhang, Tiejian Luo, and Longyin Wen. 2022. Structured context transformer for generic event boundary detection. *arXiv preprint arXiv:2206.02985* (2022).
- [13] Congcong Li, Xinyao Wang, Longyin Wen, Dexiang Hong, Tiejian Luo, and Libo Zhang. 2022. End-to-end compressed video representation learning for generic event boundary detection. In *IEEE Conf. Comput. Vis. Pattern Recog.* 13967–13976.
- [14] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. 2019. Bmn: Boundary-matching network for temporal action proposal generation. In *Int. Conf. Comput. Vis.* 3889–3898.
- [15] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. 2020. Intra-and inter-action understanding via temporal action parsing. In *IEEE Conf. Comput. Vis. Pattern Recog.* 730–739.
- [16] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. 2021. Generic event boundary detection: A benchmark for event segmentation. In *Int. Conf. Comput. Vis.* 8075–8084.
- [17] Jing Tan, Yuhong Wang, Gangshan Wu, and Limin Wang. 2023. Temporal Perceiver: A General Architecture for Arbitrary Boundary Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* (2023).
- [18] Jiaqi Tang, Zhaoyang Liu, Chen Qian, Wayne Wu, and Limin Wang. 2022. Progressive attention on multi-level dense difference maps for generic event boundary detection. In *IEEE Conf. Comput. Vis. Pattern Recog.* 3355–3364.
- [19] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. 2019. Video classification with channel-separated convolutional networks. In *Int. Conf. Comput. Vis.* 5552–5561.
- [20] Libo Zhang, Xin Gu, Congcong Li, Tiejian Luo, and Heng Fan. 2023. Local Compressed Video Stream Learning for Generic Event Boundary Detection. *Int. J. Comput. Vis.* (2023), 1–18.

755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812