CSTA: CNN-based Spatiotemporal Attention for Video Summarization

Supplementary Material

A. Experiment details

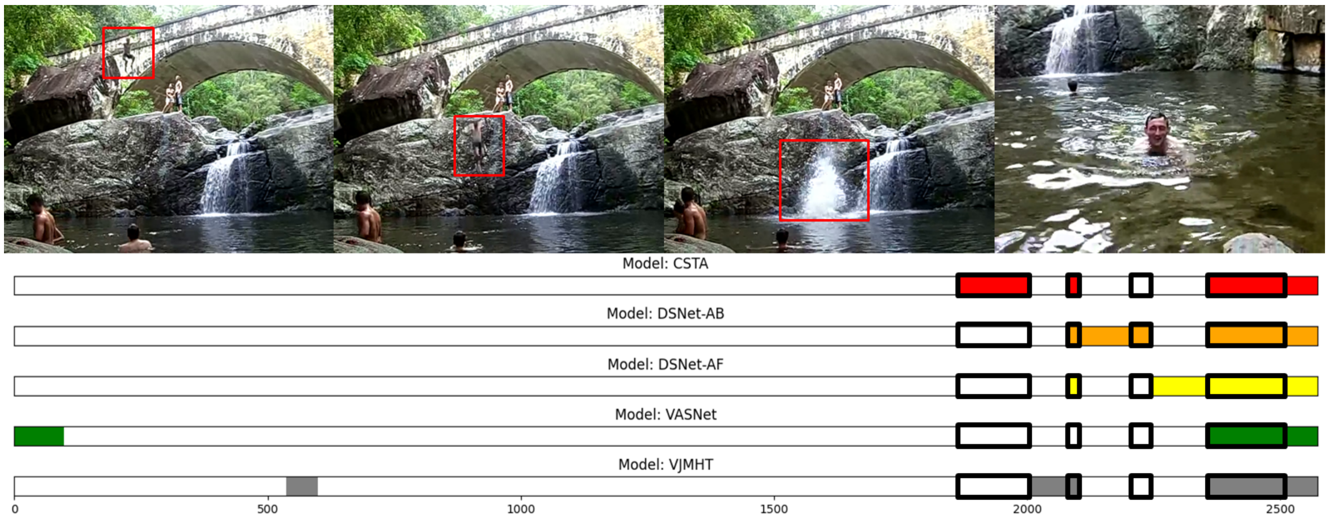
A.1. Measure correlation

Based on [37], we test everything 10 times in each experiment for strict evaluation since video summarization models are sensitive to randomness due to the lack of datasets. Additionally, we follow [37] to perform the experiments rigorously by using non-overlapping five-fold cross-validation for reflection of all videos as test data. For each fold, we use 80% of the videos in the dataset for training and 20% for testing. We then average the results of all folds to export the final score. Owing to non-overlapping videos in the training data in each split, different training epochs are required; therefore, we pick the model that shows the best performance on test data during the training epochs of each split. During training, the predicted score for each input video is compared to the average score of all ground truth scores of summary videos for that input video. During inference, the performance for each video is calculated by comparing each ground truth score with the predicted score and then averaging them.

A.2. Implementation details

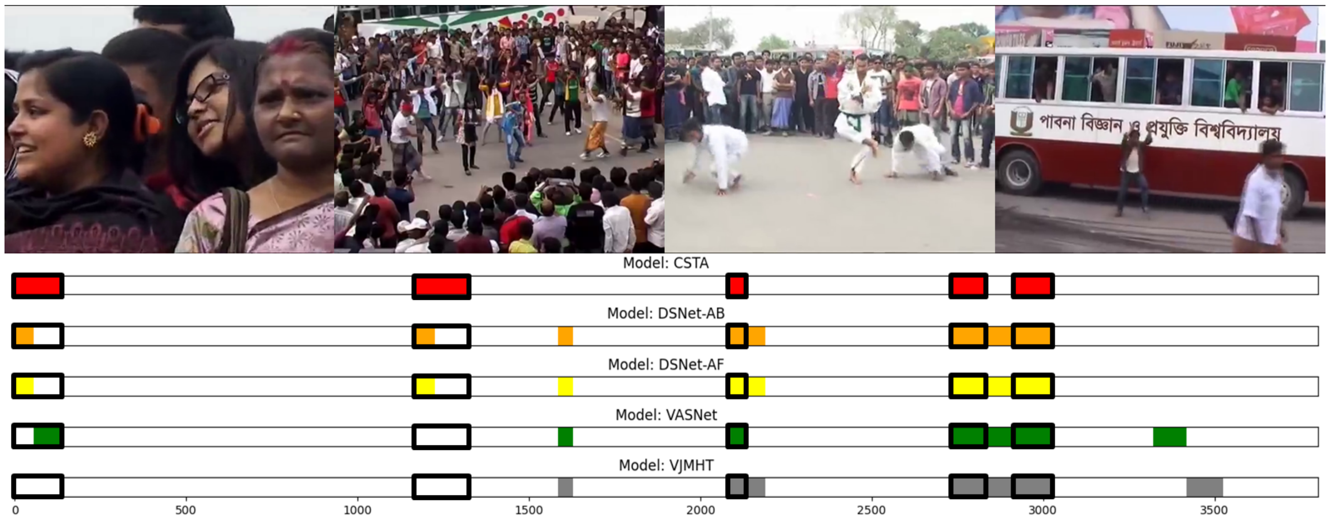
For a fair comparison, we follow the standard procedure [9, 15, 24, 25, 28, 42] by uniformly subsampling the videos to 2 fps and acquiring the image representation of every frame from GoogleNet [35]. GoogleNet is also used as a trainable CNN to match the dimension of all features to 1,024, and all CNN models are pre-trained on ImageNet [5]. The initial weights of the linear layers in the classifier are initialized by Xavier initialization [10], while key and value embeddings are initialized randomly. The output channels of linear layers and key and value embedding dimensions are 1,024. The reduction ratio r in CNN is 32, an inherent trait of GoogleNet, and all adaptive pooling layers are adaptive average pooling operations. The shape of the CLS token is $3\times1{,}024$, the epsilon value for layer normalization is 1e-6, and the dropout rate is 0.6. We train CSTA on a single NVIDIA GeForce RTX 4090 for 100 epochs with a batch size of 1 and use an Adam optimizer [23] with 1e-3 as the learning rate and 1e-7 as weight decay.

B. Summary video visualization

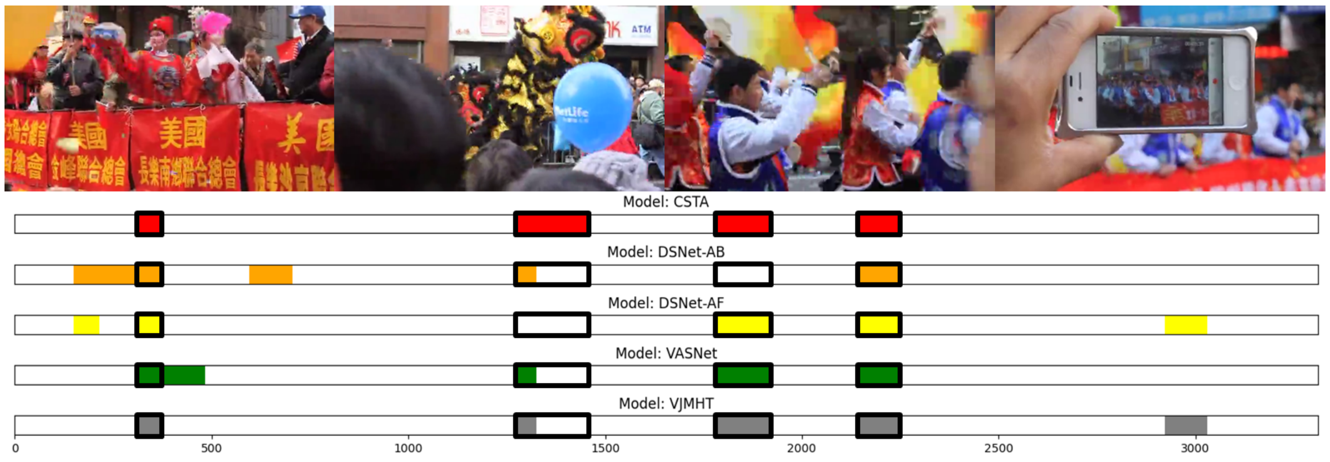


(a) The images from the summary video titled "paluma jump" about people diving into the water.

As shown in Figure 5, we visualize and compare the generated summary videos from different models. We compared CSTA with DSNet-AB, DSNet-AF [47], VASNet [7], and VJMHT [24]. The videos were selected from SumMe (Figure 5a) and TVSum (Figure 5b and Figure 5c). Since each model used different videos for the test, we chose videos used for training by all models.



(b) The images from the summary video titled "ICC World Twenty20 Bangladesh 2014 Flash Mob - Pabna University of Science & Technology (PUST)" about people performing flash mobs on the street and crowds watching them.



(c) The images from the summary video titled "Chinese New Year Parade 2012 New York City Chinatown" about the parade celebrating the Chinese New Year on the streets of New York City.

Figure 5. Visualization and comparison of summary videos generated by different models. The images above are the frames selected by CSTA as parts of the summary video. The graphs below show which frames models pick as keyframes. From the graphs, each row is the result of each model. The x-axis is the order of the frames, and the black boxes are the ground truth frames. The color parts are the frames each model selects, and the white parts are the frames unselected by each model.

The summary video in Figure 5a was taken during the "paluma jump," representing people diving into the water at Paluma. The first three frames show the exact moment people dive into the water. Based on the selected frames in the graphs, CSTA selects keyframes that represent the main content of the video more accurately than the other models. Although other models chose key moments in the later part of the video, they did not search for diving moments as precisely as CSTA.

The summary video in Figure 5b was taken during the "ICC World Twenty20 Bangladesh 2014 Flash Mob - Pabna University of Science & Technology (PUST)," representing flash mops on the street. The frames selected by CSTA display different flash mop performances on the street and the people watching them. CSTA selects keyframes in videos more often than the other models, which either select non-keyframes or skip keyframes, as shown in the graphs in Figure 5b.

The summary video in Figure 5c was taken during the "Chinese New Year Parade 2012 New York City Chinatown," representing the parade celebrating the Chinese New Year in New York City. Based on the chosen images, CSTA finds representative frames containing the parade or people reacting to it (*e.g.*, images showing tiger-like masks, people marching on the street, or people recording the parade, respectively). Unlike the other models, the graphs exhibit CSTA creating exactly the same summary videos with ground truth. These results suggest the superiority of CSTA over the other models.

C. CNN models

Baseline	SumMe		TVSum		CSTA	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
MobileNet-V2[33]	0.170	0.189	0.122	0.155	CSTA(MobileNet-V2)	0.228	0.255	0.194	0.254
EfficientNet-B0[36]	0.185	0.206	0.119	0.150	CSTA(EfficientNet-B0)	0.222	0.247	0.194	0.255
ResNet-18[14]	0.167	0.187	0.140	0.178	CSTA(ResNet-18)	0.225	0.251	0.195	0.256

Table 4. The results of CSTA with different CNN models as the baseline.

We tested CSTA using different CNN models as the baseline, as shown in Table 4. We unified the dimension size to 1,024 because each CNN model exports different dimensions of features. All CNN models improved their performance with the CSTA architecture. This supports the notion that CSTA does not work only for GoogleNet.

D. Architecture history

Here, we provide a step-by-step explanation of how CSTA is constructed. For all experiments, τ and ρ represent Kendall's and Spearman's coefficients, respectively. The score marked in bold indicates that the model was selected because it yielded the best performance from the experiment. Each experiment was tested 10 times for strict verification, and the average score was recorded as the final one.

As explained in Section 4.1, the form of the ground truth of SumMe is summary videos, so the models aim to generate summary videos correctly. Kendall's and Spearman's coefficients between the predicted and ground truth summary videos are the basis for evaluating the performance of SumMe. Based on the code provided in previous studies [9, 13], we generate summary videos by assigning 1 for selected frames and 0 otherwise. In videos, most frames are not keyframes, so the performance of SumMe is usually higher than that of TVSum. The form of the ground truth of TVSum is the shot-level importance score, so models should aim to predict accurate shot-level importance scores. The scores of entire frames are determined by assigning the identical scores of subsampled frames to nearby frames based on the code provided in previous studies [1, 7, 13, 47]. Therefore, Kendall's and Spearman's coefficients of subsampled frames are the basis for evaluating the performance of TVSum. The difference between SumMe and TVSum can cause bad performance on TVSum, and the performance on SumMe looks much better than that on TVSum.

D.1. Channel of input feature

Channels	SumMe		TVSum	
	τ	ρ	τ	ρ
1	0.169	0.188	0.122	0.154
3	0.176	0.197	0.129	0.163

Table 5. Comparison of different number of input channels on GoogleNet.

In Table 5, we test the number of channels of input frame features. As explained in Section 3.2, we copy the input frame feature two times to create three channels of input to match the number of common channels of images that are usually used to train CNN models. We use GoogleNet as the baseline and check the results when the channels of input frame features are 1 and 3. The model taking 3 channels of features as inputs performs better than taking a single channel of features. This supports the idea that creating the shape of input frame features the same as the RGB images helps to utilize CNN models better.

D.2. Verify attention

Softmax. We verify the effects of attention structure and perform softmax along different axes, as shown in Table 6. Attention structure, composed of CNN generating attention maps and the classifier predicting scores, increases the baseline performance regardless of softmax (Baseline+Att). Compared to the baseline, the attention structure increased by 0.038

Setting	SumMe		TVSum	
	τ	ρ	τ	ρ
Baseline(GoogleNet)	0.176	0.197	0.129	0.163
Baseline+Att	**0.214**	**0.239**	**0.167**	**0.219**
Baseline+$Att + Soft_T$	0.184	0.205	0.176	0.231
Baseline+$Att + Soft_D$	0.186	0.207	0.170	0.224
Baseline+Att+Soft$_{\mathbf{T\&D}}$	**0.189**	**0.211**	**0.182**	**0.240**

Table 6. The ablation study for the softmax. The baseline is the plain GoogleNet summarizing videos, which is the same as in Figure 4. *Att* is an attention-based CNN structure without softmax. *Soft* applies the softmax operation to the model along the frame axis (T), dimension axis (D), or both axes ($T\&D$).

and 0.042 for Kendall's and Spearman's coefficients, respectively, on SumMe, where it increased by 0.038 and 0.056 for Kendall's and Spearman's coefficients, respectively, on TVSum. This demonstrates CNN's ability as an attention algorithm.

Reflecting weighted values between frames ($Att + Soft_T$) or dimensions ($Att + Soft_D$) improved the baseline model by at least 0.008 on SumMe and 0.041 on TVSum. This supports the importance of spatial attention, and it is better to consider weighted values along both the frame and dimension axes ($Att + Soft_{T\&D}$) than focusing on only one of the axes.

On SumMe, the model without softmax is better than the model with softmax; however, the reverse is the case on TVSum. Since there is no best model for all datasets, we choose both models as the baseline and find the best one when extending the structure.

Setting	SumMe		TVSum	
	τ	ρ	τ	ρ
Baseline(Att$_{\mathbf{T\&D}}$)	**0.189**	**0.211**	**0.182**	**0.240**
Baseline+$Balance_T$	0.186	0.207	0.178	0.233
Baseline+$Balance_D$	0.186	0.207	0.182	0.239
Baseline+$Balance_{BD}$	0.186	0.207	0.175	0.230
Baseline+$Balance_{BU}$	0.187	0.208	0.183	0.240

Table 7. The ablation study for balancing ratio. The baseline applies the softmax operation to the attention map along the frame and dimension axes (Table 6). *Balance* is the balancing ratio between frames and dimensions. T adjusts the weighted values along the frame axis to the dimension axis. D adjusts the scale of the weighted values along the dimension axis to the frame axis. BD decreases the scale of larger ones into smaller ones. BU upscales the scale of smaller ones into larger ones.

Balance Ratio. We hypothesize that the imbalance ratio between frames and dimensions deteriorates the performance of the model with softmax. For example, suppose the number of frames is 100, and the dimension size is 1,000. In this case, the weighted values between frames are usually larger than those between dimensions (on average, 0.01 and 0.001 between frames and dimensions, respectively). This situation can lead to overfitting the frame importance, so we tested the performance of the model under a balanced ratio between the number of frames and dimensions, as shown in Table 7. However, all results were worse than the baseline, so we used the default setting.

D.3. Self-attention extension

Given that our model operates the attention structure differently from existing ones, we must test which existing method works for CSTA. First, we verify the key, value embeddings, and scaling factors used in self-attention [38]. The key and value embeddings project input data into another space by exploiting linear layers. At the same time, the scaling factor divides all values of attention maps with the size of the dimension. Unlike self-attention handling 1-dimensional data, we should consider the frame and dimension axes for the scaling factor because of 2-dimensional data. We test the scaling factor using the size of dimensions ($Scale_D$), frames ($Scale_T$), and both ($Scale_{T\&D}$).

The best performance for the model without softmax is achieved by utilizing the key, value embedding, and scaling factors with the size of frames ($EMB + Scale_T$), as shown in Table 8a. Although utilizing the scaling factors with the size of dimensions ($Scale_D$) yields better performance than $EMB + Scale_T$ on SumMe, it yields much worse performance on

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline(Att)	0.214	0.239	0.167	0.219	Baseline+EMB	0.158	0.177	0.033	0.042
Baseline+$Scale_D$	0.220	0.246	0.173	0.227	Baseline+$EMB + Scale_D$	0.209	0.238	0.187	0.246
Baseline+$Scale_T$	0.213	0.238	0.173	0.227	**Baseline+EMB+Scale$_\mathbf{T}$**	**0.214**	**0.239**	**0.187**	**0.245**
Baseline+$Scale_{T\&D}$	0.196	0.218	0.154	0.203	Baseline+$EMB + Scale_{T\&D}$	0.192	0.215	0.191	0.250

(a) The result of the model without softmax as the baseline.

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline($Att_{T\&D}$)	0.189	0.211	0.182	0.240	**Baseline+EMB**	**0.207**	**0.231**	**0.193**	**0.253**
Baseline+$Scale_D$	0.151	0.168	0.192	0.251	Baseline+$EMB + Scale_D$	0.162	0.181	0.190	0.249
Baseline+$Scale_T$	0.160	0.178	0.192	0.252	Baseline+$EMB + Scale_T$	0.163	0.182	0.191	0.251
Baseline+$Scale_{T\&D}$	0.149	0.166	0.187	0.146	Baseline+$EMB + Scale_{T\&D}$	0.163	0.181	0.187	0.244

(b) The result of the model with softmax as the baseline.

Table 8. The ablation study for methods in self-attention. Att is the model without softmax, and $Att_{T\&D}$ is the model with softmax along the frame and dimension axes (Table 6). EMB employs key and value embedding into the baseline model. $Scale$ divides the values of attention maps by the number of frames (T) or dimensions (D) or both of them ($T\&D$).

TVSum, even considering the performance gaps on SumMe. Also, $EMB + Scale_D$ and $EMB + Scale_{T\&D}$ show slightly better performance than $EMB + Scale_T$ on TVSum, but much worse on SumMe. We select $EMB + Scale_T$ as the best model based on overall performance.

For the model with softmax, we select the model using key and value embedding (Baseline+EMB) because it reveals the best performance for all datasets, as shown in Table 8b.

D.4. Transformer extension

We verify the methods used in transformers [38], which are positional encodings and dropouts. Positional encoding strengthens position awareness, whereas dropout enhances generalization. We expect the same effects when we apply the positional encodings and dropouts to the input frame features. We use fixed positional encoding (FPE) [38], relative positional encoding (RPE) [1], learnable positional encoding (LPE) [38], and conditional positional encoding (CPE) [4]. We must test both 2-dimensional (TD) and 1-dimensional (T) positional encoding matrices to represent temporal position explicitly because the data structure differs from the original positional encoding that handles only 1-dimensional data. For CPE, T operates a depth-wise 1D CNN operation for each channel, whereas TD operates entire channels. We use 0.1 as the dropout ratio, the same as [38].

The results of both models with and without softmax reveal that employing positional encodings or dropout into the input frame features deteriorates the performance of Kendall's and Spearman's coefficients for all datasets, as shown in Table 9. We suppose that adding different values to each frame feature leads to distortion of data, making it difficult for the model to learn patterns from frame features because CSTA considers the frame features as images. If more data are available, the model can learn location information from these positional encodings because they are similar to bias. Thus, all results yielded by the model with and without softmax worsen when using positional encoding or dropout on SumMe. However, some results are similar to or even better than the baseline on TVSum because TVSum has more data than SumMe. Due to the lack of data, we chose the baseline models based on performance.

D.5. PGL-SUM extension

Unlike existing transformers, PGL-SUM [1] proves effects when applying positional encodings and dropouts to the multiplication outputs between key and query vectors. We adopted the same methods to further improve the model's positional recognition effectiveness by adding the positional encodings and applying dropouts to CNN's outputs. We use 0.5 for the dropout ratio, the same as [1].

The best performance for the model without and with softmax is achieved by Baseline+$Drop$ and Baseline+$FPE(TD) + Drop$, respectively, as shown in Table 10. In Table 10b, some models perform slightly better than Baseline+$FPE(TD) + Drop$ on TVSum, but their performance is considerably worse than the selected one on SumMe. Comparing the best performance in both tables, we observe that the performance of the model with softmax

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline(Att)	**0.214**	**0.239**	**0.187**	**0.245**	Baseline+*Drop*	0.181	0.202	0.190	0.250
Baseline+$FPE(TD)$	0.145	0.161	0.031	0.041	Baseline+$FPE(TD)$ + *Drop*	0.148	0.165	-0.007	-0.009
Baseline+$FPE(T)$	0.192	0.214	0.189	0.248	Baseline+$FPE(T)$ + *Drop*	0.179	0.200	0.191	0.251
Baseline+$RPE(TD)$	0.193	0.216	0.190	0.249	Baseline+$RPE(TD)$ + *Drop*	0.172	0.192	0.191	0.250
Baseline+$RPE(T)$	0.195	0.217	0.191	0.251	Baseline+$RPE(T)$ + *Drop*	0.190	0.211	0.188	0.247
Baseline+$LPE(TD)$	0.140	0.156	-0.032	-0.043	Baseline+$LPE(TD)$ + *Drop*	0.136	0.151	-0.004	-0.005
Baseline+$LPE(T)$	0.152	0.169	0.062	0.082	Baseline+$LPE(T)$ + *Drop*	0.138	0.153	0.036	0.048
Baseline+$CPE(TD)$	0.143	0.160	0.168	0.221	Baseline+$CPE(TD)$ + *Drop*	0.139	0.155	0.175	0.229
Baseline+$CPE(T)$	0.142	0.158	0.140	0.183	Baseline+$CPE(T)$ + *Drop*	0.140	0.156	0.145	0.190

(a) The results of the model without softmax as the baseline. Att applies key and value embedding with scaling factor for the size of frames without softmax (Table 8a).

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline($\mathbf{Att_{T\&D}}$)	**0.207**	**0.231**	**0.193**	**0.253**	Baseline+*Drop*	0.134	0.149	0.192	0.252
Baseline+$FPE(TD)$	0.144	0.160	0.152	0.199	Baseline+$FPE(TD)$ + *Drop*	0.148	0.165	0.134	0.176
Baseline+$FPE(T)$	0.138	0.154	0.191	0.250	Baseline+$FPE(T)$ + *Drop*	0.152	0.170	0.193	0.253
Baseline+$RPE(TD)$	0.149	0.166	0.193	0.253	Baseline+$RPE(TD)$ + *Drop*	0.151	0.168	0.193	0.253
Baseline+$RPE(T)$	0.142	0.159	0.192	0.252	Baseline+$RPE(T)$ + *Drop*	0.158	0.176	0.192	0.252
Baseline+$LPE(TD)$	0.152	0.169	0.078	0.102	Baseline+$LPE(TD)$ + *Drop*	0.148	0.164	0.052	0.068
Baseline+$LPE(T)$	0.134	0.149	0.079	0.103	Baseline+$LPE(T)$ + *Drop*	0.145	0.162	0.089	0.117
Baseline+$CPE(TD)$	0.135	0.150	0.119	0.156	Baseline+$CPE(TD)$ + *Drop*	0.146	0.162	0.114	0.149
Baseline+$CPE(T)$	0.143	0.159	0.180	0.236	Baseline+$CPE(T)$ + *Drop*	0.149	0.166	0.184	0.241

(b) The results of the model with softmax as the baseline. $Att_{T\&D}$ applies key and value embedding with softmax (Table 8b).

Table 9. The ablation study for methods in transformers. FPE is fixed positional encoding, RPE is relative positional encoding, LPE is learnable positional encoding, and CPE is conditional positional encoding. T represents the frame axis, and TD represents the frame and dimension axis for positional encoding. *Drop* exploits dropout to input frame features.

(Baseline+$FPE(TD)$ + *Drop*) is better than that without softmax (Baseline+*Drop*) for all datasets. Although their performance is similar on TVSum, the baseline model without softmax shows 0.219 and 0.244 for Kendall's and Spearman's coefficients, respectively, on SumMe. The baseline model with softmax shows 0.225 and 0.251 for Kendall's and Spearman's coefficients, respectively, on SumMe. Based on the performance, we chose the model with softmax using $FPE(TD)$ and *Drop* as the final model.

D.6. CLS token

We further test the CLS token [6] at different combining places. CSTA fuses the CLS token with input frame features right after employing CNN or softmax or creating final features. The final features are created by applying attention maps to input features.

Combining the CLS token after creating the final features yields the best performance, as shown in Table 11. We hypothesize that the reason is that the classifier is generalized. The CLS token is trained jointly with the model to reflect the overall information of the dataset. The global cues of the dataset generalize the classifier because fully connected layers are found in the classifier. Thus, all results using the CLS token improved the baseline. Moreover, adding the CLS token after creating the final features means incorporating the CLS token just before the classifier. For this reason, the best performance is achieved by delivering the CLS token without changes and generalizing the classifier the most.

D.7. Skip connection

We finally verify the skip connection [14] for stable optimization of CSTA, as shown in Table 12. Without layer normalization, adopting the skip connection by adding outputs from the key embedding and CNN (SC_{KC}) yields the best performance among all settings. However, it yields slightly worse performance than the baseline model for all datasets. Using layer normalization with SC_{KC} ($SC_{KC} + LN$) shows slightly less performance than the baseline model on TVSum, whereas it yields much better performance than the baseline on SumMe. For better overall performance, we selected the combination of skip

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline(Att)	0.214	0.239	0.187	0.245	Baseline+$Drop$	0.219	0.244	0.191	0.250
Baseline+$FPE(TD)$	0.118	0.131	0.064	0.084	Baseline+$FPE(TD)+Drop$	0.164	0.183	0.149	0.197
Baseline+$FPE(T1)$	0.184	0.205	0.169	0.222	Baseline+$FPE(T1)+Drop$	0.183	0.204	0.171	0.225
Baseline+$RPE(TD)$	0.207	0.231	0.143	0.188	Baseline+$RPE(TD)+Drop$	0.203	0.227	0.146	0.192
Baseline+$RPE(T1)$	0.187	0.209	0.164	0.216	Baseline+$RPE(T1)+Drop$	0.176	0.196	0.166	0.218
Baseline+$LPE(TD)$	0.190	0.212	0.136	0.179	Baseline+$LPE(TD)+Drop$	0.202	0.226	0.151	0.199
Baseline+$LPE(T1)$	0.184	0.205	0.173	0.227	Baseline+$LPE(T1)+Drop$	0.178	0.199	0.177	0.233
Baseline+$CPE(TD)$	0.141	0.157	0.157	0.206	Baseline+$CPE(TD)+Drop$	0.123	0.136	0.145	0.191
Baseline+$CPE(T1)$	0.111	0.123	0.069	0.090	Baseline+$CPE(T1)+Drop$	0.129	0.144	0.090	0.118

(a) The results of the model without softmax as the baseline. Att applies key and value embedding with scaling factor for the size of frames without softmax (Table 8a).

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline($Att_{T\&D}$)	0.207	0.231	0.193	0.253	Baseline+$Drop$	0.204	0.228	0.193	0.253
Baseline+$FPE(TD)$	0.222	0.248	0.188	0.247	**Baseline+FPE(TD)+Drop**	**0.225**	**0.251**	**0.191**	**0.251**
Baseline+$FPE(T1)$	0.207	0.231	0.183	0.241	Baseline+$FPE(T1)+Drop$	0.199	0.222	0.184	0.243
Baseline+$RPE(TD)$	0.200	0.223	0.189	0.248	Baseline+$RPE(TD)+Drop$	0.205	0.229	0.189	0.248
Baseline+$RPE(T1)$	0.214	0.239	0.184	0.242	Baseline+$RPE(T1)+Drop$	0.209	0.233	0.184	0.243
Baseline+$LPE(TD)$	0.216	0.241	0.185	0.243	Baseline+$LPE(TD)+Drop$	0.217	0.242	0.186	0.245
Baseline+$LPE(T1)$	0.197	0.220	0.181	0.239	Baseline+$LPE(T1)+Drop$	0.207	0.231	0.181	0.238
Baseline+$CPE(TD)$	0.204	0.228	0.187	0.246	Baseline+$CPE(TD)+Drop$	0.202	0.226	0.190	0.249
Baseline+$CPE(T1)$	0.209	0.233	0.192	0.253	Baseline+$CPE(T1)+Drop$	0.205	0.228	0.190	0.250

(b) The results of the model with softmax as the baseline. $Att_{T\&D}$ applies key and value embedding with softmax (Table 8b).

Table 10. The ablation study for methods in PGL-SUM. FPE is fixed positional encoding, RPE is relative positional encoding, LPE is learnable positional encoding, and CPE is conditional positional encoding. T is the frame axis, and TD is the frame and dimension axes for positional encoding. $Drop$ exploits dropout to features after positional encoding.

Setting	SumMe		TVSum	
	τ	ρ	τ	ρ
Baseline($Att_{T\&D}$)	0.225	0.251	0.191	0.251
Baseline+CLS_{CNN}	0.232	0.259	0.194	0.254
Baseline+CLS_{SM}	0.233	0.260	0.193	0.254
Baseline+CLS$_{\textbf{Final}}$	**0.236**	**0.263**	**0.194**	**0.254**

Table 11. The ablation study for the CLS token. $Att_{T\&D}$ is the baseline model applying FPE and dropout (Table 10). CLS is the model that uses the CLS token and combines it with frame features after CNN (CNN) or softmax (SM) or creating final features (Final).

Setting	SumMe		TVSum		Setting	SumMe		TVSum	
	τ	ρ	τ	ρ		τ	ρ	τ	ρ
Baseline($Att_{T\&D}$)	0.236	0.263	0.194	0.254					
Baseline+SC_{KC}	0.233	0.261	0.193	0.253	**Baseline+SC$_{\textbf{KC}}$+LN**	**0.243**	**0.271**	**0.192**	**0.252**
Baseline+SC_{CF}	0.115	0.128	0.043	0.056	Baseline+$SC_{CF}+LN$	0.162	0.181	0.052	0.068
Baseline+SC_{IF}	0.126	0.141	-0.018	-0.024	Baseline+$SC_{IF}+LN$	0.163	0.181	0.186	0.244

Table 12. The ablation study for the skip connection. $Att_{T\&D}$ is the baseline model incorporating the CLS token after creating the final features (Table 11). SC means skip connection. KC is the key embedding output fused with CNN output. CF is CNN output combined with final features. IF is the combined input and final features. LN is layer normalization exploited immediately after the skip connection.

connections, which is the sum of both key embedding and CNN outputs, and layer normalization as the final model.

E. Hyperparameter setting

Dataset	Correlation	**Batch size=1**	Batch size=25%	Batch size=50%	Batch size=75%	Batch size=100%
SumMe	τ	**0.243**	0.214	0.210	0.211	0.202
	ρ	**0.271**	0.239	0.235	0.235	0.225
TVSum	τ	**0.192**	0.173	0.161	0.163	0.152
	ρ	**0.252**	0.228	0.212	0.214	0.200

(a) The ablation study for different batch sizes. Each percentage is the batch size ratio of the entire training dataset.

Dataset	Correlation	Dropout=0.9	Dropout=0.8	Dropout=0.7	**Dropout=0.6**	Dropout=0.5
SumMe	τ	0.227	0.237	0.240	**0.246**	0.243
	ρ	0.253	0.264	0.268	**0.275**	0.271
TVSum	τ	0.198	0.196	0.194	**0.192**	0.192
	ρ	0.260	0.258	0.255	**0.252**	0.252
Dataset	Correlation	Dropout=0.4	Dropout=0.3	Dropout=0.2	Dropout=0.1	
SumMe	τ	0.243	0.239	0.241	0.237	
	ρ	0.271	0.267	0.269	0.264	
TVSum	τ	0.192	0.192	0.190	0.192	
	ρ	0.252	0.253	0.250	0.252	

(b) The ablation study for different dropout ratios used for CNN outputs with a single batch size (Table 13a).

Dataset	Correlation	WD=1e-1	WD=1e-2	WD=1e-3	WD=1e-4	WD=1e-5
SumMe	τ	0.174	0.178	0.176	0.203	0.227
	ρ	0.194	0.199	0.197	0.226	0.253
TVSum	τ	0.055	0.082	0.195	0.195	0.199
	ρ	0.070	0.107	0.256	0.255	0.261
Dataset	Correlation	WD=1e-6	**WD=1e-7**	WD=1e-8	WD=0	
SumMe	τ	0.237	**0.246**	0.242	0.246	
	ρ	0.264	**0.274**	0.270	0.275	
TVSum	τ	0.194	**0.194**	0.193	0.192	
	ρ	0.255	**0.255**	0.253	0.252	

(c) The ablation study for different weight decays with a single batch size and dropout ratio of 0.6 (Table 13b). WD means weight decay.

Dataset	Correlation	LR=1e-1	LR=1e-2	**LR=1e-3**	LR=1e-4
SumMe	τ	-0.292	0.164	**0.246**	0.211
	ρ	-0.284	0.182	**0.274**	0.236
TVSum	τ	-0.089	0.060	**0.194**	0.162
	ρ	-0.080	0.080	**0.255**	0.213
Dataset	Correlation	LR=1e-5	LR=1e-6	LR=1e-7	LR=1e-8
SumMe	τ	0.162	0.163	0.168	0.153
	ρ	0.181	0.182	0.187	0.171
TVSum	τ	0.101	0.030	0.035	0.037
	ρ	0.133	0.039	0.046	0.049

(d) The ablation study for different learning rates with a single batch size, a dropout ratio of 0.6, and weight decay of 1e-7 (Table 13c). LR means learning rate.

Table 13. The ablation study for different hyperparameter settings.

Here, we test the model with different hyperparameter values. The best performance of the model is achieved with a single batch size, and it keeps decreasing with larger batch sizes, as shown in Table 13a. Thus, we use the single batch size.

The bigger the dropout ratio, the better the model's performance on TVSum, as shown in Table 13b. However, the performance of the model is bad if the dropout ratio is too large, so we chose 0.6 as the dropout ratio, considering both performance on SumMe and TVSum.

We fixed the dropout ratio at 0.6 and tested with various values of weight decay, as shown in Table 13c. The performance increased as the value of weight decay decreased. When weight decay is 1e-7, the performance on SumMe is similar to that without weight decay. However, it shows slightly better performance on TVSum than the model with 0 weight decay. Thus, considering the overall performance on SumMe and TVSum, we select 1e-7 as the final value for weight decay.

In Table 13d, we finally test different learning rates with 1e-7 as weight decay. When the learning rate is too large, the performance is terrible for all datasets. When the learning rate is too small, the performance is also poor. When the learning rate is 1e-3, it shows the best performance, so we decided on this value as the final learning rate.