1. Implementation Details

We implement the proposed method in PyTorch platform with SGD optimizer. We use a batch size of 1024. The initial learning rate is set 0.01 and scaled by a factor of 0.1 for every 50 epochs. The $L_2$ weight decay coefficient is set to $5 \times 10^{-4}$. We empirically set $T = 2$ for both YouTube and TVSum dataset, and the training procedure is terminated after 200 epochs. All experiments were conducted on a machine with a single NVIDIA TITAN RTX GPU.

For a fair comparison, we reproduced and reported the results of the MINI-Net [5], which also utilizes the auditory and visual information, with their officially released codes and trained on our self-collected dataset. Following the protocol widely used in [6, 5], we trained the model on self-collected dataset, then evaluate on the benchmark datasets (i.e., YouTube Highlights and TVSum). In particular, Xiong et al. [6] collected approximate 10 million videos from Instagram for training, and Hong et al. [5] trained MINI-Net with their self-collected approximate 200k videos with 8k videos per topic through contacting with the authors since they did not mention these in the paper. However, those datasets are not publicly available in which we doubt with their actual performance. For this reason, following the same protocol [6, 5], we crawled about 35k videos (average of 1.4k videos per topic) based on hashtags from Instagram as training set.

Besides, we sample the topic-specific videos that are shorter than 60 seconds as positive videos, and take the video longer than 60 seconds in videos with different tags as negative videos. To preprocess each video, we break a video up uniformly into one-second clips and randomly sample the consecutive clips ($\tau = 60$ clips) during training.

For audio and visual feature extraction, we use 3DResNet-34 network [3] pretrained on Kinetics-600 dataset [1] to extract the visual feature $f_v \in \mathbb{R}^{512}$, and the audio feature $f_a \in \mathbb{R}^{128}$ is extracted by VGGish model [4] pretrained on AudioSet dataset [2].

Besides, we follow the standard evaluation protocol as [6, 5], i.e., the mean average precision is utilized to measure the model performance on YouTube Highlight dataset, and Top-5 mean average precision for TVSum dataset.

2. Computation Issue

In this part, we investigate the computation efficiency for audio-visual tensor fusion module. Assume the inputted video segment embedded features are denoted as $f_v \in \mathbb{R}^{d_v}$ and $f_a \in \mathbb{R}^{d_a}$. Then the time complexity of computing fused features $f_h \in \mathbb{R}^{d_h}$ with core tensor $T_c$ is $O(d_v d_a d_h)$ since $f_h = (T_c \times_1 f_v) \times_2 f_a$. For simplicity, we set $d_v = d_a = d_h = d$, so the time complexity of above equation is $O(d^3)$.

However, the low-rank audio-visual tensor fusion module of our approach only requires $O(Rd^2)$, which is faster than the original version since $R \ll d$. Therefore, our method is more efficient compared with the method without low-rank constraint.

3. More Quantitative Results

We vary the rank constraints $R$ to the audio-visual feature fusion. As explained earlier, we use a series of rank one kernels to decompose the multi-modal feature representation $T_c$ that we argues by doing his, we project the fused features into an unimodal subspace. We found it is crucial to choose the right amount of constraints to balance the complexity of decomposition while maintaining the useful interactions between video and audio features. As we can see in Figure 1, there is a considerable gain in the detection performance as stronger constraint putting into place until the critical point i.e., $R = 8$ that excessive constraints will no longer help and the performance starts to drop moderately.

To further validate the benefit of combining multiple modalities, we trained the models without audio features or visual features, and the testing results are summarized in Table 1. From the table, we can observe that even only trained with visual features, our method is still able to achieve com-
that can be easily detected, the improvement would not be
ever, since CoSum dataset is dominated by single scenes
outperformed other state-of-the-art methods. How-

4. We can observe that our method achieves 0.9304 mAP,

topics. The experimental results are summarized in Table

The CoSum dataset contains 51 videos with 10 different
in addition to the YouTube Highlights and TVSum datasets.

tends to produce better performance.

Moreover, we conduct experiment on the CoSum dataset

Figure 1. Variations in performance by constrain the rank R of
core tensor $T$.

Figure 1. Variations in performance by constrain the rank R of
core tensor $T$.

Table 1. Performance comparison of different models with single
modality and dual-modalities on two datasets. * indicates our
implementation trained on self-collected dataset.

Table 1. Performance comparison of different models with single
modality and dual-modalities on two datasets. * indicates our
implementation trained on self-collected dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>YouTube</th>
<th>TVSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINI-Net* w/o vision</td>
<td>0.4853</td>
<td>0.5474</td>
</tr>
<tr>
<td>MINI-Net* w/o audio</td>
<td>0.5539</td>
<td>0.6675</td>
</tr>
<tr>
<td>MINI-Net* [5]</td>
<td>0.5837</td>
<td>0.7020</td>
</tr>
<tr>
<td>Ours w/o vision</td>
<td>0.5316</td>
<td>0.6041</td>
</tr>
<tr>
<td>Ours w/o audio</td>
<td>0.5892</td>
<td>0.7364</td>
</tr>
<tr>
<td>Ours</td>
<td>0.6297</td>
<td>0.7682</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Clips</th>
<th>YouTube</th>
<th>TVSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 20$</td>
<td>0.6173</td>
<td>0.7427</td>
</tr>
<tr>
<td>$\tau = 40$</td>
<td>0.6208</td>
<td>0.7415</td>
</tr>
<tr>
<td>$\tau = 60$</td>
<td>0.6297</td>
<td>0.7682</td>
</tr>
<tr>
<td>$\tau = 80$</td>
<td>0.6223</td>
<td>0.7440</td>
</tr>
</tbody>
</table>

Table 2. Performance comparison of different number of clips in
training process on two datasets.

tables, and the results are shown in Table 3. It

Table 3. Average mAP comparison under different scale of sparsity
regularization $\beta$ on two datasets.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>YouTube</th>
<th>TVSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001</td>
<td>0.6297</td>
<td>0.7682</td>
</tr>
<tr>
<td>0.001</td>
<td>0.6272</td>
<td>0.7603</td>
</tr>
<tr>
<td>0.01</td>
<td>0.6033</td>
<td>0.7195</td>
</tr>
<tr>
<td>0.1</td>
<td>0.6035</td>
<td>0.7161</td>
</tr>
</tbody>
</table>

Table 3. Average mAP comparison under different scale of sparsity
regularization $\beta$ on two datasets.

Table 3. Average mAP comparison under different scale of sparsity
regularization $\beta$ on two datasets.

significant.

4. Supplementary Proof

In this section, we provide mathematical proof of our
video score fusion scheme in attention-gated instance aggre-
gation module in order to alleviate gradient vanishing
problem encountered during positive video optimization.

We revisit the conventional Noise-OR video score aggre-
gation method as:

$$\hat{p}^{(i)}_V = 1 - \prod_{s=1}^{m} (1 - p_s^{(i)}),$$ (1)

where $p_s^{(i)}$ is the confidence score for segment $v_s^{(i)}$ of
video $V^{(i)}$. Then, we consider the optimization for pos-
itive videos, and take the partial derivative of binary cross
entropy loss $\mathcal{L}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial p_j^{(i)}} = \frac{\partial \mathcal{L}}{\partial p_V^{(i)}} \left( \prod_{k=1,k\neq j}^{m} (1 - p_k^{(i)}) \right) = \frac{\hat{p}^{(i)}_V - 1}{\hat{p}^{(i)}_V (1 - p_j^{(i)})}. $$ (2)

Then, provided that we pick two arbitrary segments
$u$, $v \in [1, m]$ in the positive video, and set $p_u^{(i)} = c, p_v^{(i)} = 1 - c$, where $\epsilon < 0$. In addition, the rest $p_j^{(i)}$ are set to
$\delta \in (0, 1)$. The video score in Eq.(1) is computed as:

$$\hat{p}^{(i)}_V = 1 - \prod_{j=1}^{m} (1 - p_j^{(i)}) = 1 - \epsilon(1 - \epsilon)^{m-2},$$ (3)

which indicates that $\hat{p}^{(i)}_V \rightarrow 1$. Therefore, we can conclude that $\partial \mathcal{L}/\partial p_j^{(i)} \rightarrow 0$ from Eq.(2), which results in the grad-
itent vanishing issue during the optimization.

By contrast, in the attention-gated instance aggregation
module, we define video score as:

$$p_s = \sigma(W_p c_s^{(T)} + b_p),$$ (4)

$$\hat{p}^{(i)}_V = \sigma \left( W_p \sum_{j=1}^{m} \alpha_j c_j^{(T)} + b_p \right),$$ (5)

where $\sigma(\cdot)$ is the normalized function represented as
$\sigma(x) = 1/(1 + \exp(-x))$. For simplicity, we set $\alpha_i =
Supervised Methods

Weakly Supervised Methods

<table>
<thead>
<tr>
<th>Topic</th>
<th>Supervised Methods</th>
<th>Weakly Supervised Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KVS</td>
<td>PBS</td>
</tr>
<tr>
<td>Base Jump</td>
<td>0.662</td>
<td>0.672</td>
</tr>
<tr>
<td>bike Polo</td>
<td>0.674</td>
<td>0.682</td>
</tr>
<tr>
<td>Eiffel Tower</td>
<td>0.731</td>
<td>0.744</td>
</tr>
<tr>
<td>Excavators River Cross</td>
<td>0.685</td>
<td>0.694</td>
</tr>
<tr>
<td>Kids Play in Leaves</td>
<td>0.701</td>
<td>0.705</td>
</tr>
<tr>
<td>MLB</td>
<td>0.668</td>
<td>0.677</td>
</tr>
<tr>
<td>Notre Dame Cathedral</td>
<td>0.671</td>
<td>0.681</td>
</tr>
<tr>
<td>Statue of Liberty</td>
<td>0.713</td>
<td>0.722</td>
</tr>
<tr>
<td>Surf</td>
<td>0.642</td>
<td>0.648</td>
</tr>
<tr>
<td>Average</td>
<td>0.684</td>
<td>0.692</td>
</tr>
</tbody>
</table>

Table 4. Performance comparison (Top-5 mAP score) on CoSum dataset. Our method outperforms against all of the compared state-of-the-art methods. * indicates our implementation trained on self-collected dataset.

1(i = 1, · · · , m) and b = 0. Therefore, we can reform the video score as follows:

\[ p_V^{(i)} = \frac{1}{1 + \exp(-W_p \sum_{j=1}^{m} e_j^{(T)})} = \frac{1}{1 + \prod_{j=1}^{m} \exp(-W_p e_j^{(T)})} = \frac{1}{1 + \prod_{j=1}^{m} \left(1 + \frac{1}{\sigma(W_p e_j^{(T)}) - 1}\right)}. \] (6)

Similarly, the gradient of binary cross entropy loss \( \mathcal{L} \) can be computed as:

\[ \frac{\partial \mathcal{L}}{\partial p_j^{(i)}} = \frac{p_j^{(i)} - 1}{p_j^{(i)}(1 - p_j^{(i)})}. \] (7)

Without loss of generality, taking the same settings mentioned above, we can observe:

\[ p_V^{(i)} = \frac{1}{1 + \prod_{j=1}^{m} \left(\frac{1}{p_j^{(i)}} - 1\right)} = \frac{1}{1 + (\frac{1}{\delta} - 1)^{m-2}}, \] (8)

where \( \delta \in (0, 1) \) is a constant, which shows that \( p_V^{(i)} \to 1 \). As a consequence, \( \partial \mathcal{L}/\partial p_j^{(i)} = \frac{1}{1 + \left(\frac{\delta}{1-\delta}\right)^{m-2}} \to 0 \).

Besides, we also represent the visualization of binary case between original Noise-OR method and our proposed method in Figure 2. The area that gradient vanishes for Noise-OR method is much larger than that of our proposed method, which verifies the fact that our method can ease the gradient vanishing problem.

5. Visual Examples

In addition, we also illustrate the highlight detection results in Figure 3.

Reference


[2] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal,
Figure 3. Qualitative results of our method on highlight detection. (Best viewed in color.)