Detecting Content Segments from Online Sports Streaming Events: Challenges and Solutions

Zongyi Liu*, Yarong Feng, Shunyan Luo, Yuan Ling, Shujing Dong, Shuyi Wang
Customer Experience and Business Trends, Amazon.com
2121 7th Ave, Seattle, WA, 98121
joeliu*, yarongf, shunyl, yualing, shujdong, wanshuyi@amazon.com

Abstract

Developing a client-side segmentation algorithm for online sports streaming holds significant importance. For instance, in order to assess the video quality from an end-user perspective such as artifact detection, it is important to initially segment the content within the streaming playback. The challenge lies in localizing the content due to the intricate scene changes between content and non-content sections in popular sports like football, tennis, baseball, and more. Client-side content detection can be implemented in two ways: intrusively, involving the interception of network traffic and parsing service provider data and logs, or non-intrusively, which entails capturing streamed videos from content providers and subjecting them to analysis using computer vision technologies. In this paper, we introduce a non-intrusive framework that leverages a combination of traditional machine learning algorithms and deep neural networks (DNN) to distinguish content sections from non-content sections across various online sports streaming services. Our algorithm has demonstrated a remarkable level of accuracy and effectiveness in sports broadcasting events, effectively overcoming the complexities introduced by intricate non-content insertion methods during the games.

1. Introduction

The rapid growth of streaming services has made it important to quantify the video quality from end-user perspective. Some of the key metrics include device specific artifacts detection such as large scaling or interlacing during the game broadcasting time. To compute these metrics, it is important to initially segment the content within the streaming playback. Today, several computer vision-based approaches for client-side content detection have been proposed. For instance, in 2005, Hua *et al.* [29] proposed an algorithm that first extracted context-based features from a video and then applied a Support Vector Machine (SVM). In 2006, M. Covel *et al.* [9] presented an approach that used both acoustic and visual cues to detect repeated signals and then segmented out non-content sections. More recently, Xu and Du [7] introduced a method that searches for merchant logos in a video stream using template matching. However, this method requires a collection of merchant logos of interest. Liu *et al.* [16] presented an algorithm that combines hand-crafted features from the visual, textural, and audio modalities and employs the Tri-Adaboost classifier to separate non-content segments from content segments. Despite the rapid development of deep learning in the computer vision area, most of researches are either focused on the $2D$ spatial segmentation, such as the popular Faster-RCNN [23] and the recent prompt-based SAM algorithm [14], or are in the activity detection or scene change domain temporal segmentation, such as [6, 30, 32]. only a couple of DNN algorithms have been built for the content detection. One of them is the Ad-Net proposed by Minaee *et al.* [21], which first uses an open-source video shot algorithm [1] and then applies a pre-trained DNN to classify segmented video clips. The other work is proposed by Z. Liu [17], which uses audio data to perform the segmentation first and then fuses the video and audio data and applies a DNN to classify each segment.

In client-side based content detection, there are two general approaches: intrusive and non-intrusive. The intrusive approach intercepts network traffic and parses it to extract relevant information. However, this method can be challenging due to the frequent changes in data format and encryption by service providers. The non-intrusive approach captures the video and analyzes it using computer vision technologies. This approach can be divided into two groups: reference based and non-reference based. In the reference based approach, a non-content gallery is used to search for matches in the playback. However, obtaining and updating the gallery can be difficult. The non-reference based approach is more applicable to Video Quality Analysis (VQA) since it relies on analyzing audio and video features without the need for a gallery. However, this ap-

proach is also challenging, particularly for popular sports live streaming videos such as football, baseball and basketball. For example, the league promotion segments during game breaks can be similar to the game contents, as shown in Fig.1 (a) and (b). Additionally, there are also many embedded non-content, as shown in Fig.1(c)-(f).

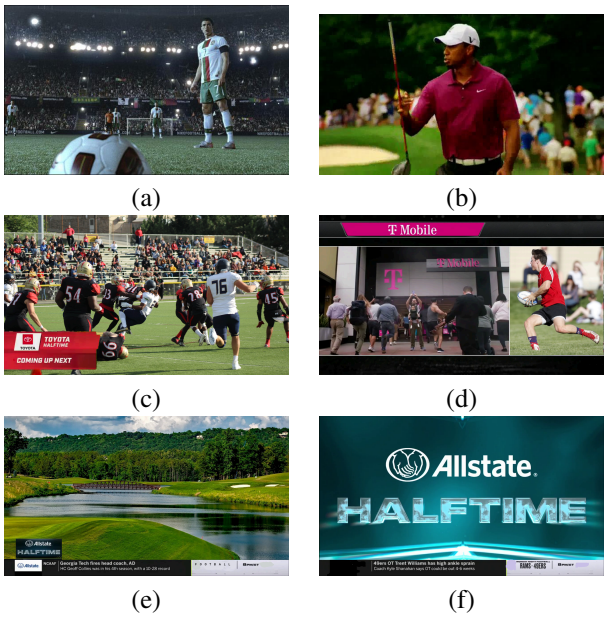

Figure 1. Examples of challenging non-content images include samples (a) and (b), which are from league promotion segments aired during game breaks, and samples (c)-(f), which are from in-game commercial that are seamlessly integrated into the broadcast of the game. The sample images listed here are publicly available online.

In this paper, we present a novel non-reference based client-side computer vision algorithm for robustly detecting content sections in online sports streaming. The algorithm is composed of two parts: a short-term classifier that takes in a 10-second video clip and outputs frame-level estimations, and a long-term predictor that takes in the full video-playback and generates more accurate segmentation output. Our main contributions include: (i) presenting a robust client-side non-reference framework that can accurately localize challenging content sections from sport streaming events on various service providers, (ii) presenting a standalone short-term frame-level classifier that takes 10 seconds long clip input with processing time only 55 milliseconds, making it suitable for real-time content detection applications, and (iii) presenting a traditional machine learning based classifier to detect league promotion segments (Fig. 1 (a)) that is very challenging because they often exhibit similar visual and audio patterns with the main games.

The non-content sections can be grouped into *intrusive* and *non-intrusive*, as shown in Fig. 6 in the Appendix. Our algorithm is specifically designed to detect the *intrusive* segments because we need to exclude them in order to measure the game streaming quality, such as the artifact detection: large scaling, interlacing, etc. as described in [19]. In our algorithm, the intrusive non-contents include all video sections during game breaks and some embedded commercials that visually affect end-user game watching experience, such as the example shown in Fig. 1 (d).

The paper is organized as follows: in Sec. 2, we provide a detailed description of the short-term classifier. In Sec. 3, we describe the structure and workflow of the long term predictor. In Sec. 4, we present the results of our evaluation using a large video dataset. Finally, in Sec. 5, we summarize our findings and suggest future research directions.

2. Short-term Frame-level Classifier

The short-term frame-level classifier is designed for real-time application. Its input is a 10-second clip, which is processed through two pathways: audio and video. Here we take both video and audio signals as the input because the previous researches have indicated that adding audio signal can help improve the temporal event segmentation results [17, 30]. In the audio pathway, we first re-sample the signal to 44.1 kHz and then apply the Log-Mel Spectrogram transformation [10, 20] to convert it into a 2D image. We then use the PANNs Cnn14 network [15] to extract audio features. In the video pathway, we first re-sample the images to 224×224 spatially and 4 fps temporally, and then extract features for each frame using the 2D Resnet50 network [12]. The video and audio features are fused at the frame level by concatenation. As the audio feature space has a different temporal dimension from the video feature space, we run a 1D convolution on the audio features to match their temporal dimension. The fused features are then processed by a transformer encoder that performs self-attention operations [26] and a customized FFN based on temporal 1D convolutions to output the final frame-level binary classification result (non-content versus content). Fig. 7 in the Appendix illustrates the workflow.

In our work, we disentangled the spatial and temporal feature computation for the video pathway data. We achieved this by first running the 2D Resnet-50 on each input frame, followed by running the temporal encoder and FFN across frames. This approach is preferred over using 3D CNNs, as 2D CNNs have significantly lower computational cost. Specifically, the 3D Resnet-50 has 48 million learnable parameters, while the 2D Resnet-50 is less than half that size with only 23 million learnable parameters. Additionally, a 3D convolution of size $T \times H \times W$ is T times slower than its 2D counterpart of size $1 \times H \times W$. Furthermore, Xie *et. al* [31] found that replacing expensive 3D CNNs with low-cost 2D convolutions can result in better

performance in terms of both speed and accuracy.

We designed a customized FFN based on 1D convolution, which differs from most state-of-the-art algorithms such as the Vision Transformer [11] and the Swin Transformer [18]. Our choice of a 1D convolutional network is because, in order to classify whether a frame is an non-content or content, we often need to consider its temporally neighboring frames to make a more reliable estimation, and 1D convolution is better suited for extracting local patterns and features.

3. Long Term Section Predictor

The long term section predictor takes the output of the short term classifier, aggregates them, and produces more reliable results. Figure 2 illustrates its workflow that comprises three components: (i) a section creator that aggregates the frame-level classification results into non-content or content sections, (ii) a league promotion classifier that identifies non-content sections that closely resemble game content, as shown in Figures 1 (a) and (b), and (iii) a post-processing step that utilizes audio data and predefined keywords to update the segmented sections. We will provide a detailed description of each component in the following subsections.

3.1. Section Creator

To process an entire input video playback, we first divide it into a list of temporally adjacent clips, each 10 seconds in length. For each clip, we apply the short-term frame-level classifier as described in Section 2, resulting in T probability outputs. Here, T is set to 40 as shown in Fig. 7. We concatenate all frame-level outputs for the input video to form a long prediction sequence, $PS_{fps=4}$=($P_{11}, P_{12}, ..., P_{1T}, P_{21}, P_{22}, ..., P_{2T}$, $..., P_{K1}, P_{K2}, ..., P_{KT}$), where K is the total number of clips. We then perform a 1D temporal average pool using a window size of five seconds and a stride length of one second, resulting in a second-level (1 fps) prediction sequence $PS_{fps=1} = (P_1, P_2, ..., P_N)$. Here, N is the duration of the input video playback in seconds.

We have observed that text can be used to distinguish between non-content segments and content segments. For instance, certain keywords such as "highlights", "gameday", and channel logos (e.g., "Foo Channel") typically appear only in a specific area of the screen (e.g., top left or top right) during game segments. To leverage this observation, we developed a regional keyword detector as part of our algorithm. The detector first decodes the input video into a list of frames at a sampling rate of 1 fps. For each frame, we apply the AWS OCR algorithm [2] to extract its text. We also use a logo detector based on the faster R-CNN architecture [23] to search for TV station logos and stopwatch regions. The output is another sequence $WS_{fps=1}$=(W_1, W_2, ..., W_N), where W_i contains the text and logo regions for frame i. Note that WS is temporally aligned with the prediction sequence (PS) described above because they share the same sampling rate. For each W_t, we use an empirically-built rule engine to search for keywords in specific regions. If a match is found, we update the corresponding value (P_t) in the prediction sequence. For example, if the word "highlights" appears at the bottom right corner of the screen in W_t, we update P_t to 0 (indicating content). Note that this rule engine is generic and can be applied to videos captured from any station.

To create non-content/content sections from the updated prediction sequence $PS_{fps=1}$, we first apply a pre-defined threshold to binarize each P_i. Next, we create sections from the connected components of the positive class: *non-content* (NC), and the negative class: *content*. Additionally, we perform a $1D$ temporal morphological closing operation on the non-content sections to eliminate small fragments. The output of the section creator is a list of sections ($Sec_{NC}(1)$, $Sec_{Content}(1)$, ..., $Sec_{NC}(M)$, $Sec_{Content}(M)$), where each section is represented by its starting second ($t1$) and ending second ($t2$), denoted as $Sec(i) = (t1, t2)$.

3.2. League Promotion Classifier

Separating league promotion segments from sports content segments is challenging because they often share similar visual and audio patterns, as illustrated in Fig. 1 (a) and (b). In our algorithm, we use a novel video encoder based on camera shot change features, implemented via a classifier. Our approach relies on the observation that *non-content segments typically have more shot changes than content segments*. We will perform an empirical study on this hypothesis in Appendix Sec. A. To encode features, we first split each input section (generated by the Section Creator) into a list of overlapping video clips, each of which is 10 seconds long with a hop size of 2 seconds. We then apply a shot change detector using the Transnet V2 [25] algorithm, which is a DNN-based approach that computes the frame-level probability sequence at 15 fps. Since each clip is 10 seconds long, we obtain 150 probability values, where each value P_{shot} indicates whether the corresponding frame contains a shot change or not. To obtain clip-level features regarding shot change frequency, we first filter out frames with P_{shot} less than a pre-defined threshold T_{shot}, which is set to 0.3 in our algorithm. The remaining frames are then summarized using a 9-bin histogram of frame count based on P_{shot}. Each bin in the histogram corresponds to a specific value range of P_{shot}, such as $[0.3, 0.35)$, and has a height equal to the number of frames with P_{shot} in that range. This way, clips with more shot changes will have a right-skewed distribution represented by the histogram feature, while clips with fewer shot changes will have a left-

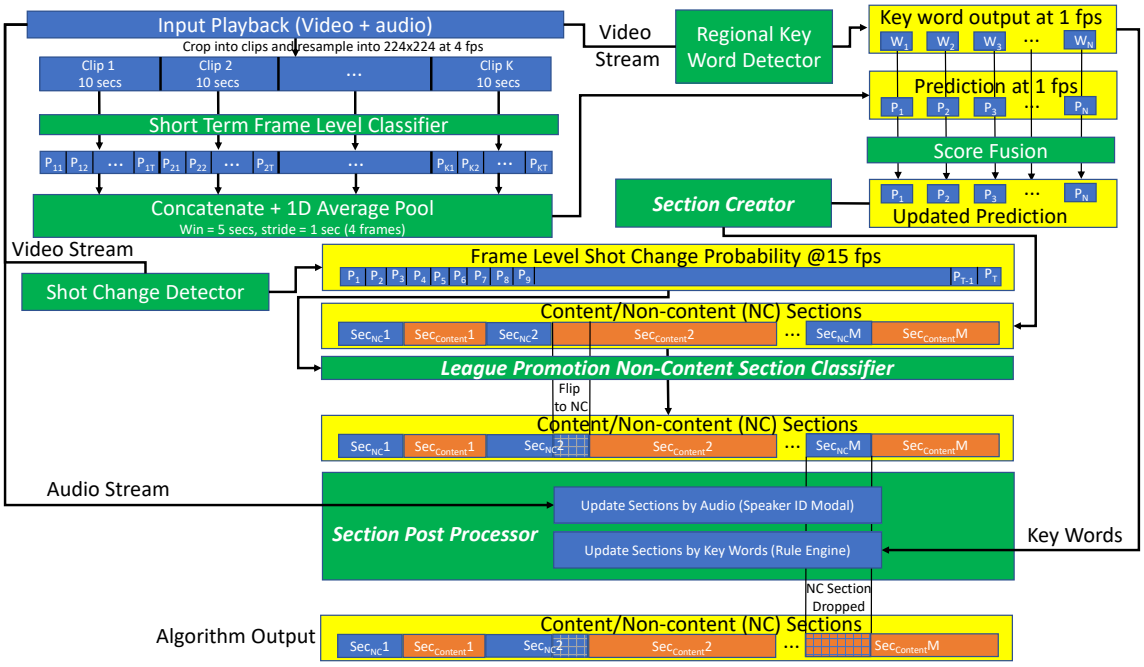

Figure 2. The workflow of the long-term section predictor involves segmenting a lengthy video playback into non-content (NC) and content sections. It consists of a *section creator* that aggregates the frame-level classification, a *league promotion classifier* that identifies non-content sections that closely resemble game content, and a *post-processing step* that utilizes audio and text signals to improve the accuracy.

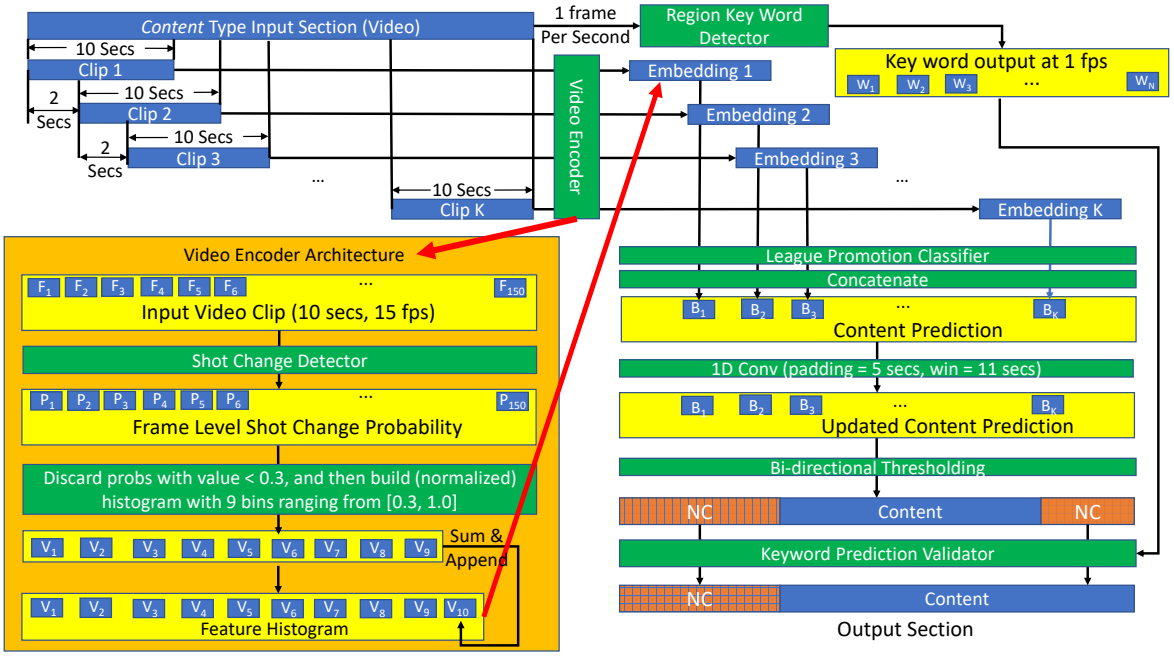

Figure 3. The workflow of the League Promotion Classifier on a sample input section of content type. In the output, the header segment is updated to non-content (NC) section type.

skewed distribution. We also append the count of frames with $P_{shot} > T_{shot}$ so that the output histogram feature per clip has 10 values.

The feature embedding for each clip is input into a binary league promotion classifier. In our algorithm, we chose the traditional Support Vector Machine (SVM) [8] with a lin-

ear kernel over a DNN due to its faster computation and the difficulty of obtaining a diverse set of league promotion segments data required to train a DNN. The predictions from all clips are concatenated into a sequence, and a $1D$ convolution with a kernel of 11 seconds and padding of 5 seconds is applied to smooth out outliers. The resulting prediction sequence is then binarized where segments with frequent shot changes are labeled as league promotion non-content type. However, we must exercise caution, as some non-content/content segments may also have infrequent/frequent shot changes. To address this issue, we reset a segment with frequent shot change to non-content type only when it is temporally adjacent to another non-content section. And we also incorporate a validation step that utilizes a keyword-based rule engine, as described in Section 3.1. For instance, if the classifier re-labels a segment of a *content*-type input video section as *league promotion* type, but many frames in this segment contain the "Foo channel" logo in the top left region, then the segment will not be updated but will be kept as *content* type.

The figure shown in Figure 3 shows a sample workflow of the league promotion classifier. In this example, we begin with a video input of type *content*. After performing video cropping and shot change feature extraction, the classifier identifies a head segment and a tail segment as league promotion type. The keyword validator then removes the tail segment part from re-labeling, resulting in only the head segment being updated as an *non-content* type in the output.

3.3. Post Processing

The post-processing step is designed to further improve the quality of the non-content/content type sections created earlier. It consists of two parts. The first part is section filtering, where we perform both audio and video checks. In the audio check, we select every section with a duration of less than 25 seconds, compute its audio features using the SpeakerID based algorithm presented in [17], and compare it with the features of its neighboring sections. If they are similar, we update this section to have the same type as its neighbors. In the video check, we select every section of *content* type with a duration of less than 30 seconds, compute the ratio of frames that contain keywords as described in Sec. 3.1. If the ratio is too low, we update the type of this section from *content* to *non-content*. Here both 25 seconds and 30 seconds are empirically set using the data as described in Sec. 4.

The second part of the post-processing step is section boundary fine-tuning. For each section Sec_t, we compare its audio features with its temporally adjacent neighbor sections Sec_{t-1} and Sec_{t+1} using the same algorithm as described above. If Sec_t has similar audio features with Sec_{t-1} as measured in Euclidean distance, it indicates that the section boundary frame (f_b) is inaccurate. In such cases, we use the probability sequence produced by the shot change detector as described in Sec. 3.2 to find the frame f_b' that has the local maximum shot change likelihood near f_b, and reset the boundary frame to f_b'. We also repeat this step if Sec_t has similar audio features with Sec_{t+1}.

Table 1. The number of video clips we used in the short-term frame level classifier training process. The content clips are broadcasting sport games. Each clip has a duration between 10 to 60 seconds.

Class	Train	Test	Validation
Non-content	5484	290	290
Content	5040	264	264

4. Experiments and Performance

4.1. Training Data and Process

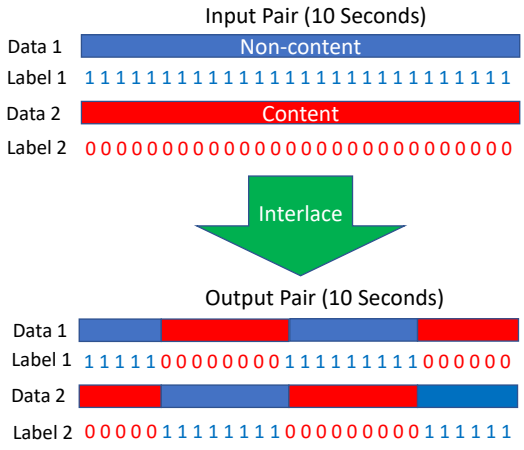

Figure 4. An example of interlacing a pair of clip data (10 seconds each) and their labels.

Our algorithm consists of several models, with the training process mainly focused on two in-house built classifiers: the short-term frame-level classifier, as described in Sec. 2, and the league promotion classifier, as described in Sec. 3.2. For the remaining parts, such as the OCR model and the TransNet v2 model, we use them off-the-shelf.

To train the short-term frame-level classifier, we collected non-content video playbacks from various sources. And all the content videos are the broadcasting sport game playbacks. Each video playback was divided into small video clips with duration ranging from 10 to 60 seconds, to increase the diversity of the data. In order to balance the dataset, we performed downsampling so that the ratio between the number of *non-content* type clips and the number of *content* type clips fell within the range of $[0.9, 1.1]$. Subsequently, we split the dataset into training, testing, and

Table 2. The number of video clips we used in the league promotion classifier training process. The video clips are created from the same source used in the short-term classifier as listed in Table 1

Class	Train	Test	Validation
Non-content	6292	700	58
Content	4687	521	112

validation sets with a ratio of 90%, 5%, and 5%, respectively, as listed in Table 1. During the training process, we created batches with an equal number of *non-content* type clips and *content* type clips, and randomly sampled 10 seconds of data from each of them. To help the classifier learn the state transitions ($Non-content \Longleftrightarrow Content$) in real-time, we employed an interlacing data augmentation step as shown in Figure 4. This ensured that each clip contained one or more state transitions.

To initialize the neurons of the short-term classifier as shown in Figure 7, we used pre-trained weights from [27] and [15] for the 2D Resnet-50 model and the PANNs Cnn14 model, respectively. For the remaining neurons, we applied the *Kaiming* initialization method provided by PyTorch [22]. With regard to the loss function, we computed the *cross-entropy loss* for each prediction output and then averaged it across all (40) samples of a (10 seconds long) input clip, as defined in Equation 1.

$$L(X,Y) = \frac{1}{40}\sum_{i=1}^{40}(-y_i log(x_i) - (1-y_i)log(1-x_i)) \quad (1)$$

where $X = (x_1, x_2, ..., x_{40})$ and $Y = (y_1, y_2, ..., y_{40})$ are the model predicted probability and the binary ground truth label for one input clip, respectively.

To train the league promotion section classifier, we utilized the same set of cropped clips used to train the short-term frame-level classifier, splitting the training and testing data in a 9:1 ratio. While the training dataset include all non-content types, the validation dataset only consists of league promotion segments in order to better evaluate the accuracy. Table 2 lists the size of each dataset. It is worth noting that this classifier does not require weight initialization since we opted for the traditional machine learning approach of using SVM.

4.2. Algorithm Performance Evaluation

The evaluation dataset comprises 63 video playbacks of online broadcasting sport games streamed on different stations. The duration of each clip ranges from 30 to 120 minutes. Human annotators manually labeled the starting and ending times (in seconds) for each *non-content* and *content* type segment. Since our algorithm is designed to detect intrusive insertion segments, non-intrusive insertion segments like the image shown in Fig. 1 (e) are labeled as *content* type. In this dataset, there are around 850 mins of *non-content* segments and 2350 mins of *content* segments. Note that some embedded commercials are difficult to label as intrusive or non-intrusive, such as the image shown in Fig. 1 (c). As only a tiny portion of the evaluation dataset ($< 1\%$) are in this case, we exclude them from the performance computation to reduce human error.

Since our algorithm's output has a temporal resolution of second, we report the performance at the second level. This means that for each second of playback, we compare the algorithm's output with the manually labeled type. We used popular metrics: precision, recall and the $f1$ score, to quantify the accuracy performance. The detailed definition of these metrics is described in Appendix Sec. C. Table 3 summarizes our algorithm's performance on the evaluation dataset. We can see that it achieves very promising results, with a precision rate above 98% and a recall rate above 95%.

Table 3. The performance of our algorithm in terms of precision rate and recall rate, both of which are measured in seconds. The evaluation dataset consists of 63 online sports video playbacks such as football with duration ranging from 30 minutes to 120 minutes.

Class	Predicted Count	Ground Truth Count	Precision Rate	Recall Rate
Non-content	50795	50741	96.7%	95.6%
Content	140708	140762	98.4%	98.8%

4.3. Generalization to Sport Types Excluded from the Training Dataset.

To evaluate the generalization of our algorithm, we intentionally exclude some types of sport events from our training dataset, such as the baseball games and the basketball games. These broadcasting events also feature numerous non-content segments that are challenging to detect, including league promotion and embedded commercial segments, as depicted in Fig. 5. Specifically, we collected four basketball playbacks and four baseball playbacks, each ranging from 30 to 70 minutes in length. Subsequently, we applied our algorithm to these playbacks and reported the performance in Table 4.

Since our algorithm hasn't been trained on these types of sports, its accuracy dropped as expected. For instance, it yielded an non-content precision rate of 87.5% for basketball games. However, this outcome is still highly encouraging in terms of the generalization. And we are confident that further enhancements can be achieved by adding video playbacks from these sports into the training dataset.

(a) (b) (c) (d) (e) (f)

Figure 5. Samples of challenging images from sports that are not included in the training set. The images listed here are publicly available online.

Table 4. The generalization of our algorithm on sports ***excluded*** from the training dataset. The accuracy is reported in terms of precision rate and recall rate, both of which are measured in seconds. The evaluation dataset four basketball playbacks and four baseball playbacks with each ranging from 30 minutes to 70 minutes long.

Sport Type	Non-content Precision	Non-content Recall	Content Precision	Content Recall
Basketball	87.5%	98.2%	99.4%	95.2%
Baseball	94.3%	94.6%	98.0%	97.9%

4.4. Short-term Classifier Window Size Study

Table 5. The performance of the short-term classifier as described in Sec. 2 in terms of different sampling window size. We used the same evaluation dataset as Table 3.

Window Secs	Non-content f1 Score	Content f1 Score
8	83.6%	93.4%
9	83.6%	94.6%
10	88.7%	96.1%

The short-term classifier's sampling window size in Sec. 2 is set to 10 seconds. This choice aligns with the prevailing approach found in leading video activity detection databases, such as the Kinetic series [5, 13, 24]. However, given the real-time nature of the short-term classifier's design, it becomes interesting to investigate its performance under the influence of smaller sampling windows. In Table 5, we have compared the accuracy achieved using sampling windows of 8, 9, and 10 seconds, respectively. Evidently, as the sampling window size decreases, there is a corresponding reduction in accuracy. This outcome is consistent with our expectations, as a shorter sampling window inherently provides a lesser amount of information.

4.5. Short-term Classifier Video/Audio Encoder Selection Study

In the short-term frame-level classifier, we utilized the ResNet50 and PANNs CNN14 models to encode video and audio features, respectively. Both of these models are based on convolutional neural networks (CNNs). As transformer architectures have shown significant improvement on solving computer vision problems, we also evaluated their impact on our application. Specifically, we studied two state-of-the-art image transformers: the Vit model [11] and the Swin model [18], as well as two state-of-the-art audio transformers: the Wav2Vec2 model [4] and the Data2Vec Audio model [15]. To gain a better understanding, we changed one encoder at a time, such as replacing the video encoder with Vit while keeping the audio encoder as PANNs CNN14.

Table 6 compares the performance of the new encoders with our encoder (the last row) in terms of memory space (during inference), running time (during inference), and accuracy. For fair comparison, all encoders are re-trained where we described the process in Appendix B. We observe that although our encoder's accuracy on the regular and baseball sports is slightly lower than (around 0.5%) the best model, it demonstrates significantly higher accuracy for the basketball sport type: surpassing other encoders by 10% or more. This suggests that our encoder can be more effectively generalized across sport types without requiring additional training data. Moreover, it also substantially outperforms other encoders in terms of memory consumption and running time. In particular, when compared to the Swin2D model which achieved the best accuracy for the regular and baseball sports, our algorithm is approximately three times faster and only requires 32% of the memory. These advantages make it a superior option for real-time applications.

4.6. Ablation Study

We studied the impact of each component in our algorithm on the overall performance and presented the results in Table 7. We can observe that the SF classifier already achieves an accuracy of approximately 87% on the *non-content* type and around 95% on the *content* type. Given that SF is a real-time classifier that only requires a 10-second video input, this performance is highly promising. Combining SF with any component of PAC, RF, and PP has been found to improve the performance, particu-

Table 6. The performance comparison between the transformer based encoders and our encoder (the last row) for the short-term classifier *only* (SF) as described in Sec. 2. The regular evaluation dataset (sport types included in the training dataset) is described in Table 3 and the basketball and baseball evaluation dataset (sport types ***excluded*** from the training dataset) are described in Table 4. The accuracy is measured in terms of the $f1$ score in percentage for the non-content (NC) and content (CT) sections. All encoders are re-trained with the process explained in Appendix B.

Video Encoder	Audio Encoder	Memory Space (Gb)	Running Time (ms)	Regular $f1$		Basketball $f1$		Baseball $f1$	
				NC	CT	NC	CT	NC	CT
Vit [11]	PANNs Cnn14 [15]	5.45	150	87.8	95.4	44.5	37.9	86.5	95.2
Swin2D [18]	PANNs Cnn14 [15]	8.73	170	**89.0**	**96.2**	52.2	64.4	**91.1**	**97.0**
Resnet50 [12]	Wav2Vec2 [4]	4.03	80	88.4	95.6	62.2	76.3	88.0	96.0
Resnet50 [12]	Data2Vec Audio [3]	4.09	88	86.6	94.7	50.5	55.4	88.1	96.0
Resnet50 [12]	PANNs Cnn14 [15]	**3.43**	**55**	88.8	96.0	**70.8**	**85.1**	90.2	96.7

Table 7. The Ablation study on each component of our algorithm in terms of the short-term classifier (SF) as described in Sec. 2, the league promotion classifier (PAC) as described in Sec. 3.2, the region key word detector (RK) as described in Sec. 3.1, and the post processing (PP) step as described in Sec. 3.3. We used the same evaluation dataset as Table 3.

Algo.	Non-Content Precision	Content Precision	Non-Content Recall	Content Recall
SF Only	90.5%	95.4%	87.1%	96.7%
SF + PAC	94.5%	97.8%	93.8%	98.0%
SF + RK	96.8%	97.3%	92.5%	98.9%
SF + PP	94.0%	96.0%	88.8%	98.0%
Full	96.7%	98.4%	95.6%	98.8%

larly for the *non-content* types. Specifically, we can see that the PAC component helps to boost the *non-content* recall rate to 93.8%, and the RK component helps to raise both the *non-content* precision and recall to 92.5% and 98.9%, respectively.

4.7. Model Complexity Study

Table 8. The number of parameters of individual models built in our algorithm. Here SF is the shot term classifier as described in Sec. 2, PAC is the league promotion classifier as described in Sec. 3.2, and SID is the SpeakerID model built in the audio check algorithm as described in Sec. 3.3.

Model	SF	PAC	SID
Parameter Count	$334M$	$206M$	$1.4M$
Running time	$55ms$	$7.3ms$	$12.8ms$

We investigated the complexity of the individual models used in our algorithm. The space complexity was measured by the number of parameters in each model, and the computational complexity was measured by running the algorithm on an AWS P3 instance with one GPU of Tesla V100 $16G$ and CPUs of Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz. The results are summarized in Table 8. We can observe that the short-term classifier has only 334 million parameters and is able to process a 10 seconds of video in 55 milliseconds. This suggests that the model can be scaled to run in a real-time application using a machine with only moderate hardware configurations.

5. Conclusions and Future Study

In this paper, we propose a client-side algorithm for effectively segmenting contents from live broadcasting sports events. Our algorithm consists of two components: a short-term frame-level classifier designed for real-time applications, processing 10-second video clips, and a long section predictor that generates more accurate predictions for hours-long videos.

To evaluate the algorithm's performance, we conducted tests on a dataset containing 63 video playbacks ranging from 30 to 120 minutes. The results demonstrate a precision rate of approximately 98% and a recall rate of approximately 95%, indicating high accuracy in segmenting content sections from non-content sections. Furthermore, we assessed the individual models employed in our algorithm in terms of their impact on accuracy, space complexity, and computational complexity. The findings suggest that these models can be integrated into real-time applications using moderate hardware configurations.

We conducted additional experiments to verify its effectiveness with sports that are excluded from the training dataset. The results demonstrated consistent performance across these sports, showcasing the algorithm's versatility and applicability to generalize to new types of sports.

When considering future research, one direction is to explore the prospect of generalizing the algorithm to segment out content sections in various live streaming events, including music concerts, social gatherings, and travel and exploration experiences.

References

[1] https://mklab.iti.gr/results/video-shot-and-scene-segmentation/. 1

[2] https://aws.amazon.com/rekognition/. 3

[3] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language, 2022. 8

[4] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc., 2020. 7, 8

[5] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600, 2018. 7

[6] Shixing Chen, Chun-Hao Liu, Xiang Hao, Xiaohan Nie, Maxim Arap, and Raffay Hamid. Movies2scenes: Using movie metadata to learn scene representation, 2023. 1

[7] Cong Xu and Xiuhua Du. A real-time adaptive algorithm for detecting advertisement logo in videos. In *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, pages 1467–1471, 2013. 1

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. 4

[9] M. Covell, S. Baluja, and M. Fink. Detecting ads in video streams using acoustic and visual cues. *Computer*, 39(12):135–137, 2006. 1

[10] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 3, 7, 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2, 8

[13] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. 7

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 1

[15] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020. 2, 6, 7, 8, 12

[16] N. Liu, Y. Zhao, Z. Zhu, and H. Lu. Exploiting visual-audio-textual characteristics for automatic tv commercial block detection and segmentation. *IEEE Transactions on Multimedia*, 13(5):961–973, 2011. 1

[17] Zongyi Liu. A deep neural framework to detect individual advertisement (ad) from videos. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3567–3576, 2023. 1, 2, 5

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. 3, 7, 8

[19] Zongyi Joe Liu, Bruce Ferry, and Simon Lacasse. A scalable deep neural network to detect low quality images without a reference. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 324–330, 2021. 2

[20] P. MERMELSTEIN. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, pages 374–388, 1976. 2

[21] Shervin Minaee, Imed Bouazizi, Prakash Kolan, and Hossein Najafzadeh. Ad-net: Audio-visual convolutional neural network for advertisement detection in videos. *CoRR*, abs/1806.08612, 2018. 1

[22] PyTorch. torch.nn.init.kaiming_normal_. https://pytorch.org/docs/stable/generated/torch.nn.init.kaiming_normal_.html, n.d. [Computer software]. 6

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 91–99, 2015. 1, 3

[24] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset, 2020. 7

[25] Tomáš Souček and Jakub Lokoč. Transnet v2: An effective deep network architecture for fast shot transition detection, 2020. 3

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2

[27] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 6, 12

[28] Thomas Wolf, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Yacine Ma, Yacine Jernite, Jean-Baptiste Plu, Loic Michou, Ian Keras, Shivan Xu, Philippe Patry, Sylvain Gugger, Moussa Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing for pytorch and tensorflow. https://huggingface.co, 2020. 12

[29] Xian-Sheng Hua, Lie Lu, and Hong-Jiang Zhang. Robust learning-based tv commercial detection. In *2005 IEEE International Conference on Multimedia and Expo*, pages 4 pp.–, 2005. 1

[30] Fanyi Xiao, Yong Jae Lee, Kristen Grauman, Jitendra Malik, and Christoph Feichtenhofer. Audiovisual slowfast networks for video recognition, 2020. 1, 2

[31] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2

[32] Xitong Yang, Fu-Jen Chu, Matt Feiszli, Raghav Goyal, Lorenzo Torresani, and Du Tran. Relational space-time query in long-form videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6398–6408, June 2023. 1

Appendix

(a) (b) (c)

Figure 6. (a) An intrusive insertion example where a commercial is played during game break, (b) an non-intrusive insertion example where the "Hyundai" logo displayed on the shirt of the player, and (c) another non-intrusive insertion example where the "Ford" and "Canon" brand names displayed on the billboard. All the images shown here are publicly available.

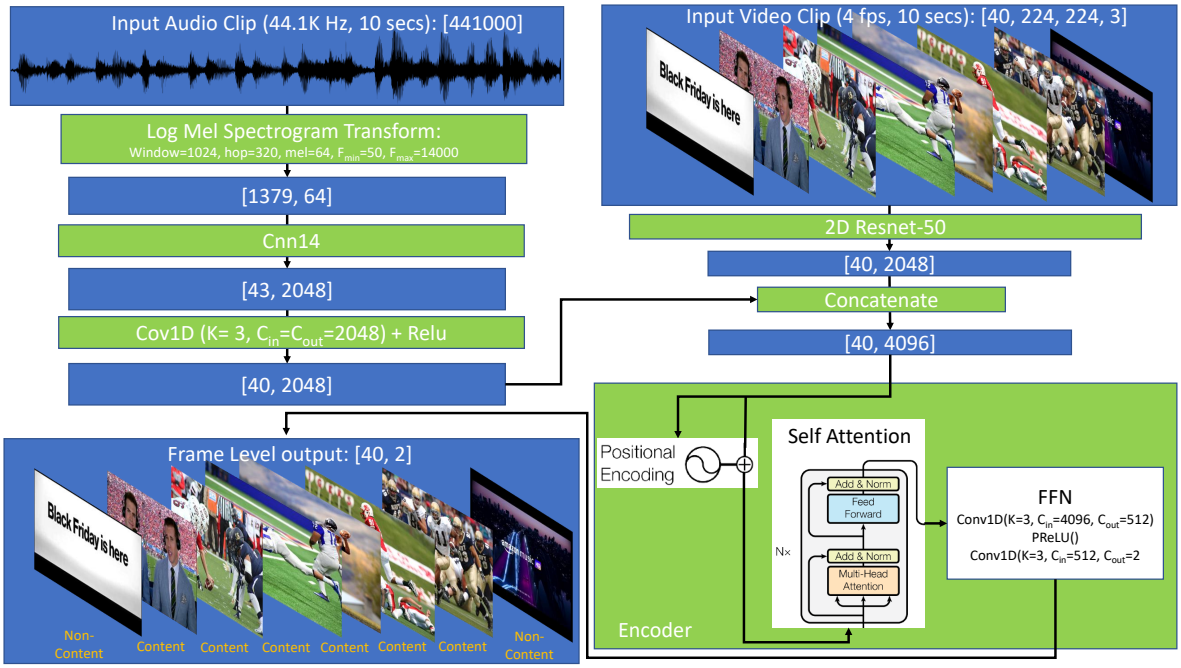

Figure 7. The workflow of the short-term frame level classifier where the input is an 10 seconds long video+audio clip and the output is the classification result of every frame. The images listed here are publicly available online.

A. Feature Histogram Study on League Promotion Classifier

In Sec. 3.2 we proposed a hypothesis that *league promotion segments typically have more shot changes than content segments*. To empirically validate this assumption, we sampled 94 sport games and 86 league promotion segments, compute the feature histogram, and plot the average value in Fig. 8. Note that here we dropped the last bin which is the sum of the first nine bins, in order to better visualize. We can see that the non-content (green) class has higher values than the content (red) class in all bins, which supports our hypothesis.

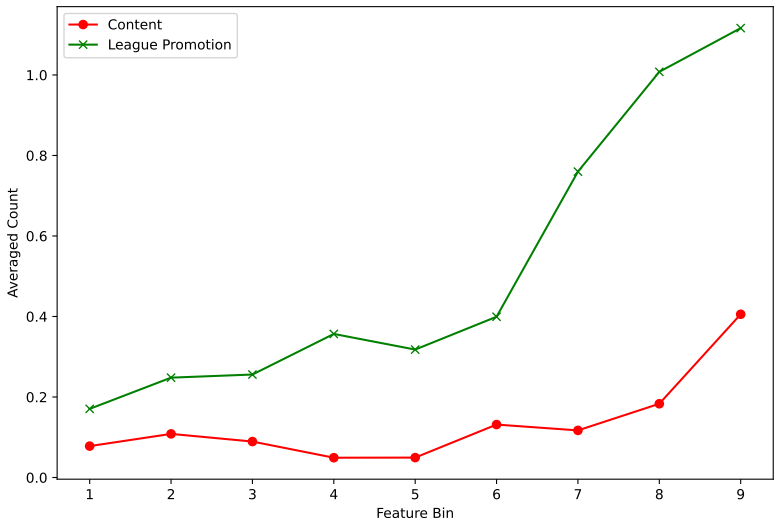

Figure 8. A sample feature histogram used as the input of the League Promotion Classifier as defined in Sec. 3.2 (Fig. 3).

B. Short-term Classifier Video/Audio Encoders Training Process

In Section 4.5, we studied the performance of different types of encoders in the short-term classifier. The results are presented in Table 6. To ensure a fair comparison, we retrained all the encoders using the same dataset as defined in Section 4.1. We employed transfer learning and initialized the weights with pretrained models obtained from [15,27,28]. Specifically, we used the following models for initialization: vit_base_patch16_224_in21k for the Vit model, swin_base_patch4_window7_224 for the Swin2D model, facebook/wav2vec2-base-960h for the Wav2Vec2 model, and facebook/data2vec-audio-base-960h for the Data2Vec Audio model.

The model complexity metrics in Table 6, namely *Memory Space* and *Running Time*, were measured by running the models in inference mode on an AWS P3 instance with a single Tesla V100 GPU.

C. Accuracy Metric Definition

The accuracy of a classifier is commonly measured by three metrics: precision, recall and $f1$ score. The *precision* metric, defined in Eq. 2, measures the accuracy of positive predictions made by a classifier. It calculates the proportion of true positive predictions among all the positive predictions made by the model.

$$Precision = \frac{\text{\# of correct positive prediction}}{\text{\# of total positive predictions}} \tag{2}$$

The *recall* metric, defined in Eq. 3, measures the ability of a classifier to identify all positive instances correctly. It calculates the proportion of true positive predictions among all the actual positive instances.

$$Recall = \frac{\text{\# of correct positive predictions}}{\text{\# of ground truth positive instances}} \tag{3}$$

The $f1$ score as defined in Eq. 4 combines precision and recall into a single metric that balances their contributions. It is the harmonic mean of precision and recall, providing a comprehensive evaluation of a classifier's performance. It is particularly useful when there is an uneven distribution between positive and negative instances in the dataset.

$$f1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$