A Deep Neural Framework to Detect Individual Advertisement (Ad) from Videos

Zongyi (Joe) Liu*, Yarong Feng, Yuan Ling, Shunyan Luo, Shujing Dong, Bruce Ferry
FGBS, Amazon.com
2121 7th Ave, Seattle, WA, USA
joeliu*, yarongf, yualing, shunyl, shujdong, bferry@amazon.com

Abstract

Detecting commercial Ads from a video is important. For example, the commercial break frequency and duration are two metrics to measure the user experience for streaming service providers such as Amazon Freevee. The detection can be done intrusively by intercepting the network traffic and then parsing the service providers data and logs, or non-intrusively by capturing the videos streamed by content providers and then analyzing using the computer vision technologies. In this paper, we present a non-intrusive framework that is able to not only detect an Ad section, but also segment out individual Ads. We show that our algorithm is scalable because it uses light weight audio data to do global segmentation, as well as is domain crossing (movies, TVs and live streaming sports) captured from the popular streaming services such as the Freevee and the Prime Video (PV) live sports.

1. Introduction

With the rapid growth of streaming services, quantifying how content providers' Ad inserting algorithms has become important for understanding end-user experience measurement. For example, how often are Ads inserted, is an Ad played at the right spot, or how relevant is an Ad to the current streaming content? To compute these metrics, we need first have an algorithm to reliably segment out individual Ads. The segmentation can be done intrusively or non-intrusively. The intrusive approach intercepts the network traffic and then builds data and logs parsers. The difficulty of this method is building a robust parser because the data and logs are usually encrypted by service providers and their formats also change frequently over time. The non-intrusive approach is to capture the video first and then analyze it using computer vision technologies. In this paper, we will focus on the non-intrusive method. Generally speaking, computer vision based Ad detection algorithms can be categorized into two groups: the first group is reference based where we use an Ad gallery to search if there is any match in a playback. The second group is non-reference based where we don't have a gallery and can only make the decision based on the audio and video features. The non-reference based approach is usually more challenging because some Ads, such as a movie trailer, are very similar to contents. But it also has more applications in the Video Quality Analysis (VQA) domain because for a streaming service provider such as the Freevee, its Ad gallery set is not only difficult to obtain, but also being updated frequently over time.

Some early works of the computer vision based Ads detection include that in 2005 Hua *et. al* [31] proposed an algorithm that first extracted the 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 Ad section. More recently, C. Xu and X. Du [8] introduced a method to search the Ad logo from a video stream using template matching. However, this method requires a collection of Ad logos of interest. N. Liu *et. al* [19] presented an algorithm that combines hand crafted features of the visual modality, textural modality and audio modality, then employed the Tri-Adaboost classifier to separate Ad from contents. Despite the rapid development of deep learning in the computer vision area these days, we are only able to find one literature algorithm that uses DNN: S. Minaee *et. al* [22] proposed an Ad-Net that first used an open source video shot algorithm [1] and then applied a pre-trained DNN to classify the segmented video clips.

In this paper, we propose a novel non-reference Ad detection framework. The main contributions of our work include (i) For the global segmentation, we propose a non-reference DNN model that is based on the audio data. Compared to video data, audio data has much higher temporal resolution: its sampling rate is usually above $16K$ HZ where video signal's sampling rate is generally below 60 frame per second (fps). In addition, audio data is much lighter weight compared to video data. For example, 20 minutes of audio playback with $16K$ hz only has 19.2 mil-

lion data points, but an RGB video with the same amount of time sampled in 60 fps will have around $7,465$ million data points. The other algorithms in this domain, on the other hand, either used hand-craft features from the video channel to perform segmentation, or used the audio input but are reference based([9]). (ii) We presented a temporal attention model to for Ads classification and compared the performances of temporal pooling with RNN algorithms, where the other algorithms are mostly using the hand-craft features. According to our study, the only literature work that used DNN model for classification is the Ad-Net presented in [22]. However, this algorithm only did not study RNN methods.

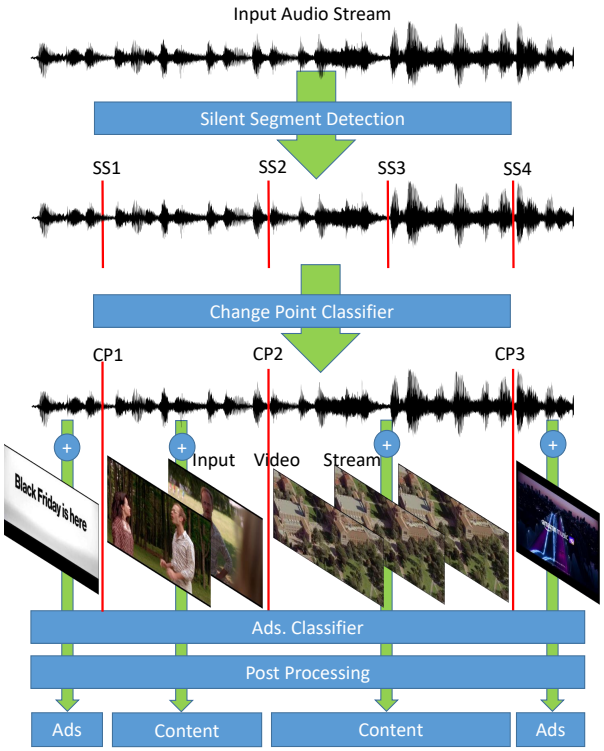

Figure 1. The flow chart of our algorithm. The sample images listed here are publicly available online.

The rest of the paper is organized as follows: in Sec. 2, 3 and 4 we describe the algorithm in detail, in Sec. 5 we evaluate our algorithm using data provided by the PV and Freevee catalogs, and in Sec. 6 we conclude the paper and discuss future directions. To improve readability, we listed the acronyms in Table. 1.

2. Audio-based Non-Reference Playback Segmentation

Our segmentation algorithm is shown in Fig. 2. It starts with searching silent short segments from the audio data,

Table 1. The acronyms used in this paper.

Acronym	Meaning
CP	Change Point: Content to/from Ad
TAP	Temporal Average Pooling Model
$LSTM$	Long Short-Term Memory Model
ViT	Vision Transformer

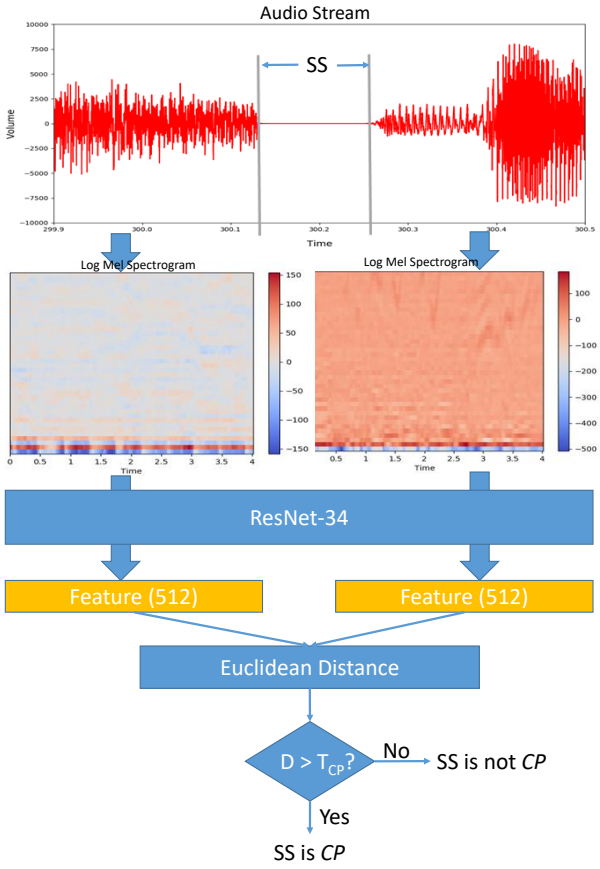

Figure 2. The flow chart of our audio-stream based playback segmentation algorithm that finds a *silent segment* (SS) first and then classifies whether it is a *change point* (CP).

based on our observation that *there is usually a short transition time between a content and an Ad, or between individual Ads*. Here a silent segment is defined to be a time period $[t_1, t_2]$ such that the maximum volume of the audio signals within it is less than a threshold. In our algorithm, the minimum duration is set to 10 milliseconds and the volume threshold is set to 4 after we normalize the wav data into $int16$ type and then take the absolute values. To clean up the noise, we apply a band pass filter of $[300HZ, 6000HZ]$ to the input audio data.

Then, we check whether a detected segment is a CP. Here we first extract the audio clips wav_1 and wav_2 that

are before and after the silent segment. Both wav_1 and wav_2 have a time duration of win. The win value selection is important for the downstream classifier because if it is too small, we don't have enough information; but if it is too large, then it may contain scene changes at other time that will confuse the classifier. We will study win value in Sec. 5. Next, we compute the Log Mel Spectrogram [10, 20] for both wav_1 and wav_2. Next, we extract the a 512 dimensional feature vectors. Then we compute the Euclidean distance between the feature vectors and label the silent segment as a CP if its value is above a pre-defined threshold T_{CP}. Here we picked the modified ResNet-34 presented by University of Oxford Robotic Lab's SpeakerID framework as the backbone. After fine-tuned it, we compute the feature vector from a Log Mel Spectrogram. Briefly speaking, this model starts with a 7×7 convolution layer, followed by 3×3 max pooling layer, which is then followed by four residual net blocks. Then it performed a temporal Self-Attentive Pooling [3] and a fully connected layer with output dimension 512. For details of this network, please refer to A. Nagrani *et. al*'s paper [23, 7, 24].

For every silent segment that is classified as non-CP, we perform an additional check using the video signals to prevent under segmentation. Specifically, we run the open source scene detect algorithm available at [2] for the frames near it. If a scene change frame is detected inside the segment, then we will re-label it as a CP. Since we only run the video detector on the frames near the detected silent segments, the additional computational cost is low.

After the computation, our algorithm outputs a detected CP list: $[CP_1, CP_2, CP_3, ..., CP_n]$. These CPs split the input playback into $n+1$ segments: $\{SG_0 = [0, t_{11}], SG_1 = [t_{12}, t_{21}], SG_2 = [t_{22}, t_{31}], ..., SG_n = [t_{n2}, t_{end}]\}$. Here t_{i1}, t_{i2} are the starting time and ending time of CP_i, and t_{end} is the ending time of the playback.

We also added a short segment clean up step: for each short segment, we check its temporal adjacent neighbors. If one or more segments are also short, we merge them and then repeat this process. Otherwise we will discard the short segment. In our algorithm, we empirically picked eight seconds as the short segment cutoff. This pruning process helps reduce the small segments which in turn improves the down-stream Ad classification step because (i) the Ad classifier in Sec. 3 requires an input clip of 4.3 seconds and (ii) we usually have better prediction for a segment with several input clips and then aggregate the result at the end. On the other hand, it creates under-segmentation problem for short Ads that are less than 8 seconds. We will address this problem in the post process step as described in Sec. 4.

To train the CP classifier, we picked the Siamese network architecture so that we can easily add new training videos without having to change the network structure. Given a training audio clip i, we first find all silent segments inside it, then randomly select one to create a non-CP (positive) pair. If no segment is found, then we randomly select a time stamp to create a positive pair. The CP (negative) pairs can be obtained by sampling audio clips from different playbacks. Since a transition can happen from content to Ad and between individual Ads, we included both Ad and content clips in the training stage. For the loss function, we studied both the triplet loss [28] as defined in Eq. 1 and the contrastive loss [5, 13] as defined in Eq. 2. The result that listed in Sec. 5 showed that triplet loss has better performance. Note that for a training batch of size NB, we have NB non-CP pairs and $(NB-1)^2$ CP pairs. So we also performed hard data mining as described in [23] during training stage to achieve better performance. Transfer learning is employed to initialize the models weights where we picked the model trained on the VoxCeleb2 dataset [7] that contains a million utterances from $6,112$ different people.

$$\begin{aligned} L(A,P,N) &= max(L(P) - L(N) + \alpha, 0) \\ L(P) &= ||f(A) - f(P)||^2 \\ L(N) &= ||f(A) - f(N)||^2 \end{aligned} \tag{1}$$

The triplet loss function where A is an anchor input, P is its non-CP pair (from the same video), N is its CP pair (from a different video), α is a margin between CP and non-CP pairs that is usually set to 1, and f is the feature extractor (ResNet-34 network).

$$\begin{aligned} L(X_1, X_2) &= L(1) + L(2) \\ L(1) &= (1-Y)\tfrac{1}{2}(||f(X_1) - f(X_2)||^2 \\ L(2) &= Y\tfrac{1}{2}max(0, \alpha - ||f(X_1) - f(X_2)||^2) \end{aligned} \tag{2}$$

The contrastive loss function where X_1 and X_2 are the input pairs of CP classifier, $alpha$ is the margin between CP and non-CP pairs that is usually set to 1, and f is the feature extractor (ResNet-34 network). $Y = 0$ if X_1 and X_2 are a non-CP pair (from the same playback), and $Y = 1$ if they are a CP pair (from different playbacks).

3. Temporal Attention Networks for Ad Classification

The segmentation algorithm described above has split an input stream into segments $[SG_0, SG_2, ..., SG_n]$. For each segment, we need to label it as Ad or content. In our algorithm, we employ a non-reference classifier that consists of a feature extraction model and a temporal attention based classification model.

3.1. Feature Extraction Model

We employed F. Xiao *et. al*'s Audiovisual SlowFast network [32] that fuses the video and audio inputs to extract the feature vector, as Fig. 3 shows. Briefly speaking, the

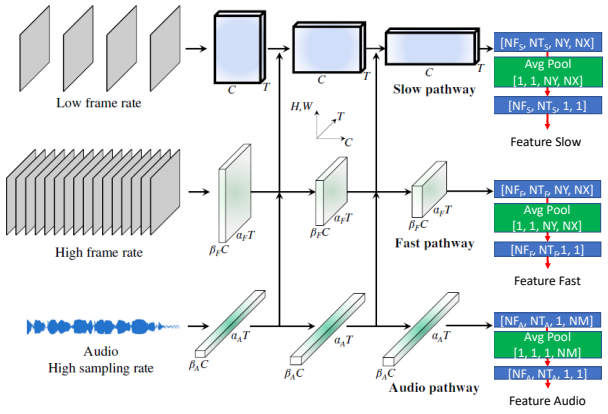

Figure 3. We use the Audiovisual SlowFast Networks [32] to extract the feature vector. Here NF_S, NF_F and NF_A are the number of channels and NT_S, NT_F and NT_A are the number of temporal resolution used in the slow, fast and audio pathway, respectively; NX, NY are the video image dimension after processed by the ResNet blocks; NM is the number of mel-spectrum of the audio frame. Note that NT_A equals to NT_F because the audio pathway modules temporally down-samples the data.

Table 2. The temporal dimensions (NT_S, NT_F and NT_A) and the number channels (NF_S, NF_F and NF_A) used in the slow, fast and audio pathways of the Audiovideo Slowfast Network to extract features.

NT_S	NT_F	NT_A	NF_S	NF_F	NF_A
4	32	256	2048	256	1024

network uses two channels to process a video stream: (i) a *slow* pathway that has a lower sampling rate but with more channels, and (ii) a *fast* pathway that has higher sampling rate but with less channels. It also creates an audio pathway that uses the $2D$ Log Mel Spectrogram [21] that has even higher temporal resolution than the fast video pathway. Then each pathway is processed by the backbone network ResNet-50. During this step, the audio pathway output are fused into the fast pathway output which are in turn fused into the slow pathway output. In addition, the audio data is also temporally down-sampled till it has the same resolution as fast pathway data, i.e., NT_A equals to NT_F. At the end, it performs XY averaging pooling on the tensors of each pathways, so that the three output feature vectors have a unique spatial dimension of 1×1. Compared with the original AVSlowFast network [32], we increased the fast pathway sampling rate from 2 to 4 so that the input clip length is doubled to 4.3 seconds. This is based on our observation that when a clip is too short, e.g., less than 3 seconds, even a human has difficulty to accurately label it. Similarly, we also doubled the audio frame number from 128 to 256.

3.2. Temporal Attention Model

Now we have three feature vectors from slow, fast and audio pathways, respectively. To run a classifier, we need to fuse them into one feature vector. In the original AvSlowFast network, the authors used $3D$ convolution to temporally down-sample the fast and audio pathway tensors and then concatenated them with the slow pathway tensor. In addition to this approach, we also studied fusing into the fast channel: we used a $3D$ dilated convolution to temporally up-sample the slow pathway tensor, and then concatenated the three pathway tensors.

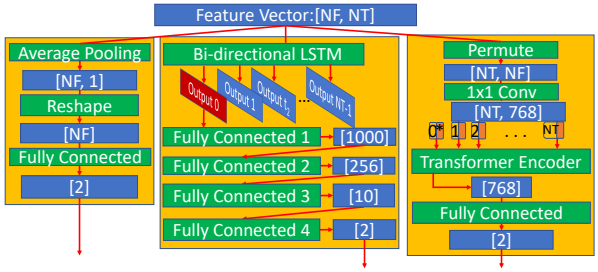

Figure 4. Three temporal attention networks studied in our algorithm. The left one is the Temporal Average Pooling Model, the middle one is the LSTM model, and the right one is the vision transformer (ViT) model. Note NF is the concatenated features from the slow, fast and audio pathway, and NT is the time resolution that can be either NF_S if we fused into the slow pathway tensor or NT_F if we fused into the fast pathway tensor.

After the cross pathway feature fusion, we performed the classification step. Here we employed the Temporal Attention Networks that have shown promising performance in the in activity recognition area [12, 33, 18, 17]. Specifically, we studied three models: the first one is the simple Temporal Average Pooling (TAP) layer followed by a fully connected layer. The second one uses a bi-directional Long Short-Term Memory (LSTM) model [15] to learn the temporal weights, then it takes the features of T_0 and sends it to a bank of fully connected layers. The third one is the Vision Transformer (ViT) recently proposed by D.Ãlexey *et al.* [11]. Here we employed the hybrid architecture that takes a sequence of flattened patch embeddings with a dimension of 768 generated by a CNN framework, adds the positional embeddings into them, then prepends a learnable embedding whose state at the output of the transformer encoder serves as the class representation. In our algorithm, we first applied 1×1 convolution to get a temporal sequence of embeddings with size 768, then ran the ViT model that produces a tensor with 768 channels, which is then sent into a fully connected layer to make a binary classification. All these three temporal attention models has an output vector of size 2×1, which is used to compute the loss function at the training stage, or is activated by a Softmax layer at

the inference stage to generate the probabilities of a binary class: Ad vs. Content. Fig. 4 shows the architectures of these three models.

3.3. Loss Function

For the loss function, we picked the popular *cross-entropy* loss as defined in Eq. 3:

$$L_{ce} = -\frac{1}{NB}\sum_{i=1}^{NB}\sum_{c=1}^{C}(y_i(c)log(x_i(c)) \quad (3)$$

where NB is training batch size, C is the number of classes with a value of two in our network, $x_i(c)$ and $y_i(c)$ and are the model predicted probability and ground truth probability for class c, respectively.

3.4. Video Segment Level Classification

The AD model described above classifies a video clip of 4.3 seconds long. To label a full segment, we cropped it into a list of 4.3 seconds clips with 2 seconds hopping size. Then we run the AD classifier on each clip to compute the ratio of clips labeled as Ad. To come up with a binary decision, we applied the similar technology as the Canny edge detector [4]: first, we set double thresholds: $T_{ad}(high)$ and $T_{ad}(low)$ to label segments as *strong* Ad, *weak* Ad or *non* Ad, based on their ratio values. Then we track Ad segments by hysteresis where we suppress those are weak and not connected to strong Ad segments. In our implementation, we set two threshold values to 0.3333 and 0.1, respectively.

Note that a streaming event such as a golf game or a movie can take hours. Unlike the CP classifier described in Sec. 2 that only uses audio data, the AD classifier needs to process both video and audio data so that it is expensive to process long video segments. On the other hand, we observed that one single Ad usually lasts less than 60 seconds. So we optimized our algorithm by creating a rule: if a video segment are longer than a certain time window, then we skip the AD classifier and directly label it as content. Here the window is empirically set to 85 seconds.

4. Post-processing Algorithm

In addition to the two DNN models to perform segmentation and classification, we also built a post-processing algorithm that consists of two steps to further improve performance.

The first step is the Ad segment under segmentation correction. In the segmentation step as described in Sec. 2, a video segment can be under-segmented by two reasons: (i) we set the minimum duration of silent segment detection to 10 milliseconds where in reality, the transition time between two Ads can be shorter than that, and (ii) we merge a video segment less than 8 seconds with its temporal adjacent neighbors. However, there are short Ads that are only 4-5 seconds. So to address this problem, we used more aggressive thresholds: (i) 2 milliseconds as opposed to 10 milliseconds for silent segment detection, and (ii) 4 seconds as opposed to 8 seconds as short video segment cut-off, to see if an Ad segment can be further split into a list subsegments. However, this process may over correct, particularly for vocal only scenes where there are pauses during conversations. To prevent this, we employed an audio neural network: $PANNs$ [16], into our system. The $PANNs$ model is trained on the large-scale AudioSet dataset and classifies an audio clip into 527 types of voices. We run PANNs model on each Ad segment and group the output into two categories: the *continuous* category including the instrument sound, fire sound, water sound, etc., and the *non-continuous* category including the human vocal sound, bird sound, reptile sound, etc. Generally speaking, it is less likely for the continuous category sounds to have false positive silent segments. On the other hand, we need to be careful to mixed sounds, e.g., human vocals with background music. This kind of sounds usually don't have false positive so that they should be classified as continuous. However, the $PANNs$ model will output higher non-continuous probability value when the foreground vocal is louder. So in our algorithm, we first weaken the vocal signal using the voice music separation algorithm [26] as implemented in Librosa [25]. As a result, vocal/music mixed audio is classified as continuous but vocal only audio is still classified as non-continuous.

We applied the $PANNs$ model based classification method described above into every video segment. Then for each adjacent pairs, if both of them are non-continuous, we merge them. Next, we run the CP classifier to further validate the remaining newly created silent segments using the aggressive thresholds.

The second step is the content segment under segmentation correction: for a long video segment (over 85 seconds in our algorithm), we check if there is Ad inside it using similar algorithm as described in Sec. 2. If it can be partitioned into multiple segments, instead of using audio based CP classifier to validate each, we run the audio plus video based AD classifier on head and tail sub segments only. If either sub-segment is classified as Ad type, then we cut it out from the video segment and create a new Ad type segment.

These two steps are designed for reduce under-segmentation error caused by pre-defined thresholds and the errors from the CP classifier. As mentioned at the end of Sec. 2, we perform these steps here because longer video clips helps increase the AD classifier's accuracy. In addition, using a conservative segmentation in the beginning will help preserve long clips. As we skip the AD classifier on long video segments, it improves the overall processing time.

Table 3. The number of video sub-playbacks in the training, testing and validation set for the CP and AD classifiers. These sub-playbacks were created from the short video playbacks (part 1) with 1 to 5 minutes long each.

	Ad Count	Content Count
Train	5609	5367
Test	1038	1154
Validate	1040	1154

5. Experiments and Performance

5.1. Dataset Description

As we are not able to find a dataset in the literature for Ad detection, we created our own dataset for performance evaluation. It consists of two parts: (i) short video playbacks with 1 to 5 minutes long each. They were collected from the Freevee catalog, the PV catalog and so on. These playbacks are further broken into sub-playbacks of $10-30$ seconds for better scene coverage in the training and testing process. (ii) Long video playbacks with each 20 to 120 minutes long. They are provided by the Freevee and PV catalogs that consist of 32 movies of different titles and 16 live sport streaming events.

5.2. Training Process Description

We used the short video dataset (part 1) to train the models. Specifically, we selected 70% of the sub-playbacks as the training data and 15% as the validation data. To have a balanced data points across classes, we performed down-sampling so that the ratio between samples of Ad and Content is within 0.9 to 1.1. We trained the CP classifier and the AD classifier separately using the same dataset. During the training process, we iterate for 196 epochs and pick the one with the best validation performance.

Transferred learning is employed for our classifiers. Specifically, we initialized the CP classifier shown in Fig. 2 using the pre-trained model provided in [6], the feature extraction model shown in Fig. 3 using the pre-trained AvSlowFast network model [32]. For the temporal attention models shown in Fig. 4, we performed initialization as below: the Temporal Average Pooling model and the LSTM model were initialized using the uniform distribution function proposed by KM He *et. al* [14]; the ViT model was initialized using the $jx\_vit\_base\_resnet50\_384$ pre-trained model downloaded from Ross Wightman's library [30].

5.3. Individual Classifiers Performance Evaluation

We evaluated the performance of our models using the remaining 15% of sub-playbacks of the short video dataset (part 1). Similar to the training process, we also performed data down-sampling to have a balanced dataset. Specifically, we have 1038 content clips and 1154 Ad clips from 121 unique playbacks.

To evaluate the CP classifier, we sequentially selected one sub-playback from each playback, randomly sampled a pair of temporally adjacent video clips from it and then computed their feature vectors. By iterating all testing playbacks, we got 121 pairs of feature vectors. Then we computed the distance matrix of 121×121, where the diagonal 121 values are non-CP distances and the remaining off-diagonal values are CP distances. Since one playback may have up to 38 sub playback, we repeated this process 38 times to iterate over these sub playbacks, and then concatenate all the distances. So in total we got $121 \times 121 \times 38 = 556358$ distances with $121 \times 38 = 4598$ of them are non-CP type.

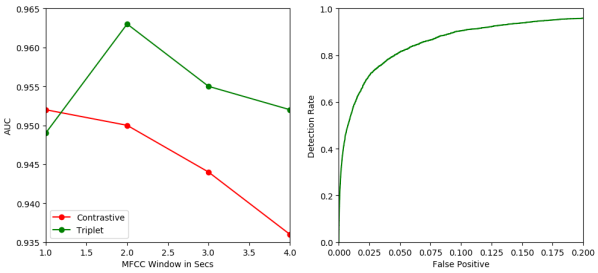

Figure 5. The performance study of the CP classifier on the testing dataset with a total of 556358 distances. The left figure is the Area Under Curve (AUC) for models trained using different values of win and loss functions L_{tr} and L_{ct}, and the right figure is the ROC curve up to 0.2 false positive rate for the model trained using the triplet loss and window size of 2 seconds.

Fig. 5 lists the performance of LC_{CP}. The left sub-figure lists the impacts in terms parameter values of win, L_{tr} as defined in Eq. 1 and L_{ct} as defined in Eq. 2. We can see that L_{tr} has better performance than L_{ct} overall. We also see that the CP classifier has best accuracy when win is set to 2 seconds. This confirms our hypothesis proposed in Sec. 2. The right sub-figure plots the ROC curve when L_{tr} is used and win is set to 2 seconds. We can see that the $\sim 90\%$ detection rate is achieved at 10% false positive rate.

To evaluate the AD classifier, we employed the same testing data sampling method as F. Xiao *et. al*'s work [32]. Briefly speaking, given a testing video clips, we performed 5 temporal uniform sampling. Then for each sample, we performed 3 spatial sampling: left, middle and right. So in total we created 15 sub-clips from a video clip. Then we ran the AD classifier on each of them, and reported the performance in Table. 4. We can see that for these Ad only or content only short videos, the TAP model achieved the best Ad recall rate and Content precision rate, and $LSTM$ model on the slow pathway output achieved the best Ad precision rate and content recall rate. Overall, similar and better balanced

Table 4. The performance of the AD classifier in terms of precision rate (PR.) and recall rate (RC.) tested on a total of 17310 Ad video clips and 17310 Content video clips. Here we compared the output of three temporal models on the slow and fast fusion pathways and the numbers are reported in percentage.

Alg.	Fusion Pathway	PR. Ad	RC. Ad	PR. Content	RC. Content
TAP	Slow	91.9	**97.5**	**97.3**	91.4
$LSTM$	Slow	**94.9**	96.4	96.4	**94.9**
$LSTM$	Fast	93.4	96.9	96.8	93.2
ViT	Slow	94.7	96.8	96.8	94.6
ViT	Fast	94.8	96.8	96.8	94.6

performance numbers are reported by the RNN models.

5.4. End-to-end algorithm Performance Evaluation

We also performed end-to-end test on the 48 long playbacks (part 2 of our database). For each playback, we ran our algorithm to compute the Ads segments, then compared them with the manually labeled segments. For simplicity, we defined that video segment i maps video segment j if their RT as defined in Eq. 4 is greater than 0.5. This simple majority rule allows each computed segment maps to 0 or 1 manual segment and vice versa. Then we quantified the performance using the five metrics as Fig. 6 defines.

$$RT_{SG_i}(SG_j) = \frac{Intersect(SG_i, SG_j)}{duration(SG_i)} \quad (4)$$

where SG_i and SG_j are two video segments.

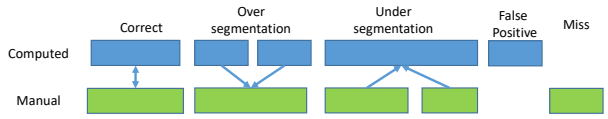

Figure 6. The five metrics defined to measure the end-to-end algorithm performance. Here an arrow from video segment i to video segment j means i maps to j.

Table. 5 listed the end-to-end performances of our algorithm on the 48 long video dataset (part 2) consisting of the 16 golf games and 32 movies of different titles. The ground truth data were created manually where we identified 621 Ad segments in this dataset with each duration ranging from 4 seconds to 60 seconds. Here the *overseg*, *underseg* and *miss* metrics are normalized with the number of ground truth Ad segments (621), and *correct* and *False Pos.* metrics are normalized with the number of computed Ads. (Col. 3 of Table. 5) We can see that all algorithms achieved $96\%+$ correct rate, less than 1% over-seg, under-seg and miss rate, and false positive rate below 2.5%. The result suggests that our model is able to work decently for both sport and movie streaming, as well as multiple streaming service providers such as the Freevee. In addition, we also noticed that for playbacks that do not contain any Ad, our algorithm does not generate any false positives.

Table 5. The end-to-end algorithm performance of the Ad detection algorithm using the long video dataset (20 minutes to 120 minutes) consisting of the 16 golf games and 32 movies of different titles. These playbacks have 621 Ad segments with duration ranging from 4 to 60 seconds. The performance numbers are in percentage where *overseg*, *underseg* and *miss* metrics are normalized with the number of ground truth Ad segments (621), and *correct* and *False Pos.* metrics are normalized with the number of computed Ads. (Col. 3).

Alg.	Fusion Pathway	# of Computed Ads	Correct	Overseg	Underseg	Miss	False Pos.
TAP	Slow	635	96.2	0.6	0.5	0.3	2.4
$LSTM$	Slow	627	**97.1**	0.6	0.5	0.3	**1.1**
$LSTM$	Fast	628	97.0	0.6	0.5	0.6	1.6
ViT	Slow	632	96.7	0.6	0.5	0.3	1.9
ViT	Fast	627	96.8	0.6	0.5	1.0	1.8

From Table. 5 we can see that the RNN based models $LSTM$ and ViT have overall better performance than the TAP model. This is expected because RNN models can learn better temporal weights than simple averaging over time. On the other hand, the $LSTM$ model on the slow pathway output achieved best performance: 97.4% correct rate and 1.1% false positive rate. This is kind of surprising as we were expecting the ViT to stand out, because many other research works [11, 29] showed that transformer is a better RNN model than $LSTM$. We speculate that this is caused by the relatively small size of our training dataset.

We performed further study on the $LSTM$ model on the slow pathway output that reported the best performance. Fig. 7 showed the *error rate* and the *loss value* changes over the training time. We can see that both of them converged after 90 epochs and achieved the best values at 190 epochs. We also computed the recognition rate of each individual component in this framework and listed the results in Table. 6. We can see that the audio-based segmentation module itself is able to achieve very descent performance.

5.5. Computational Speed Evaluation

We also studied the computational speed of our algorithm. We tested the algorithm on a Ubuntu 18.04 machine with one NVIDIA GeForce Titan X Pascal GPU with $12G$ memory and one Intel(R) Xeon(R) CPU @2.30GHz. We

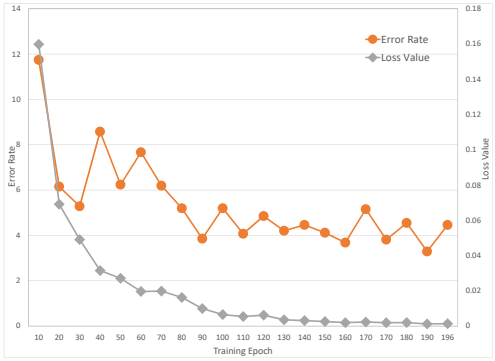

Figure 7. The error rate in percentage on the validation dataset and the loss value changes over the training epochs for the $LSTM$ model on the slow pathway output.

Table 6. The Ads segmentation accuracy using only the Audio-based segmentation algorithm vs. the full algorithm. The numbers are computed on the long video dataset with 48 playbacks captured from 16 golf games and 32 movies of different titles, and we picked the $LSTM$ model on the slow pathway output that reported the best performance.

Metrics	Audio Seg. Only	Full Algorithm
GT Ads	621	621
Computed Ads	610	627
Correct	584	609
Overseg	3	4
Underseg	13	3
Miss	8	2
False Pos.	7	7

used a mp4 testing file of 20 minutes long with 48k hz audio sampling rate and 60 fps video sampling rate. For the segmentation, the I/O time is ~ 9.8 seconds to load the full audio data and the CPU time is ~ 4.7 seconds to search silent segment. For the CP classifier, the I/O time is ~ 0.9 seconds to load a pair of two seconds long audio clips and the GPU time is ~ 0.013 seconds. For the AD classifier, the I/O time is ~ 12 seconds to load a 4.3 seconds long AV clip and the GPU time is ~ 0.04. We can see that the AD classifier is the most expensive component. Fortunately, we do not run it on the very long clips so that it substantially saves the processing time.

As mentioned earlier, one key contribution of our work is the audio-based non-reference segmentation module as described in Sec. 2. We compared its running speed with the *segmentation only* module of two state-of-art scene change detection algorithms. The first one is proposed by A. Rao *et. al.* [27] in 2020. It is a DNN based multi-model algorithm that extracts semantic features from both video and audio channels. The second one is the open source pydetect library [2]. It is a traditional CV based algorithm that uses colors from the video data. Three sample videos are tested: two of them are from our evaluation dataset and another one is from the multi-model algorithm paper [27]. Table. 7 listed the results, where we can see that our segmentation module is $6-10$ times faster than the one in the traditional CV based algorithm and $8-12$ times faster than the one in the DNN based Multi-model algorithm.

Table 7. The computation time comparison in terms of seconds between the *segmentation modules* of a DNN based Multi-modal (MM) segmentation algorithm [27], a traditional CV based scene detection algorithm [2] using colors, and our algorithm. The *Golf Game* and *Movie* testing videos are sampled from our evaluation dataset, and the *Demo* testing video is provided in the multi-model segmentation algorithm website [27] .

Alg.	Golf Game (20 mins)	Movie (60 mins)	Demo (8 minutes)
MM [27]	379	1136	42
Color [2]	283	839	30
Ours	**25**	**87**	**5**

6. Conclusions and Future Study

In this paper, we presented a non-reference algorithm to detect individual Ad from a captured video playback. It consists of three components: a segmentation step, an Ad classification step and a post-processing step. We trained and tested individual component using Ad only or content only short videos of 1-5 minutes long. We also tested the algorithm end-to-end using 48 long captured playbacks from popular streaming service providers such the Freevee, PV, etc. The playbacks are crossing multiple domains including movies, TVs and live streaming sports. The results showed that our algorithm has achieved promising performance with 97.4% accuracy. We also studied the false positive and miss segments reported by the $LSTM$ model on the slow pathway output, and listed the sample images in Fig. 1 and 2 in the Supplementary doc. We can see that the false positive errors came from the beginning and credits sections, and the miss errors came from an Ad segment that are very similar to contents.

Today, Ad research such as sentiment analysis, relevancy study, etc. are mainly using the content and Ad clips from providers where they have already been segmented. With this framework, we can study these metrics from end-user point of view by analyzing videos captured from streaming service providers. In terms of future study, one direction is to combine two DNN models into one so that we can have the whole system end-to-end trainable. Also, we are building a large Ads detection dataset including NBA and NFL live streaming contents that contain more complicated Ads scene, and will release it to the community in the near future.

References

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

[2] https://pyscenedetect.readthedocs.io/en/latest/.

[3] Weicheng Cai, Jinkun Chen, and Ming Li. In *Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System*, 04 2018.

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.

[6] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han. In defence of metric learning for speaker recognition. In *Interspeech*, 2020.

[7] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[8] 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.

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

[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.

[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.

[12] W. Du, Y. Wang, and Y. Qiao. Recurrent spatial-temporal attention network for action recognition in videos. *IEEE Transactions on Image Processing*, 27(3):1347–1360, 2018.

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] 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.

[17] D. Li, T. Yao, L. Duan, T. Mei, and Y. Rui. Unified spatio-temporal attention networks for action recognition in videos. *IEEE Transactions on Multimedia*, 21(2):416–428, 2019.

[18] J. Li, X. Liu, W. Zhang, M. Zhang, J. Song, and N. Sebe. Spatio-temporal attention networks for action recognition and detection. *IEEE Transactions on Multimedia*, 22(11):2990–3001, 2020.

[19] 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.

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

[21] Xu Min, Duan Ling-Yu, Cai Jianfei, Chia Liang-Tien, Xu Changsheng, and Tian Qi. Hmm-based audio keyword generation. In *Advances in Multimedia Information Processing - PCM 2004, 5th Pacific Rim Conference on Multimedia*, pages 566–574, Nov. 2004.

[22] 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.

[23] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. Voxceleb: Large-scale speaker verification in the wild. *Computer Science and Language*, 2019.

[24] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[25] Zafar RAFII and Bryan PARDO. Librosa library. https://librosa.org/doc/latest/auto_examples/plot_vocal_separation.html, 2012.

[26] Zafar RAFII and Bryan PARDO. Music/voice separation using the similarity matrix. In *The 13th International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2012.

[27] Anyi Rao, Linning Xu, Yu Xiong, Guodong Xu, Qingqiu Huang, Bolei Zhou, and Dahua Lin. A local-to-global approach to multi-modal movie scene segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10143–10152, 2020.

[28] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[29] 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.

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

[31] 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.

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

[33] H. Yang, C. Yuan, L. Zhang, Y. Sun, W. Hu, and S. J. Maybank. Sta-cnn: Convolutional spatial-temporal attention learning for action recognition. *IEEE Transactions on Image Processing*, 29:5783–5793, 2020.