CS230

DeepSponsorBlock

Detecting Sponsored Content in YouTube Videos

Nikhil Athreya
Dept. of Computer Science
Stanford University
nathreya@stanford.edu

Cem Gokmen
Dept. of Computer Science
Stanford University
cgokmen@stanford.edu

Jennie Yang
Dept. of Computer Science
Stanford University
jenniey@stanford.edu

Abstract

In the past few years, advertisers have begun to pay YouTube creators to integrate product promotions into their videos. Our work aims to predict where sponsored segments in videos occur solely from their frames using an encoder-decoder architecture. In the encoder step, a CNN produces an embedding for each frame, while in the decoder step, two bidirectional LSTMs predict the starting and ending frame of the sponsored segment. Our model achieves a median intersection-over-union (IOU) of 0.69 on a test set of $\sim$3000 videos, which means our model's predictions tend to have a high degree of overlap with the ground-truth labels.

1 Introduction

Internet users have taken a variety of measures to limit the presence of ads in their Internet experiences, including installing ad blockers [1] and paying for ad-free versions of services, like YouTube Premium. However, in the past few years, advertisers have started paying content creators across many platforms, from blogs to YouTube videos, to integrate product promotions into their content itself. [10].

SponsorBlock is a browser extension that was developed to combat sponsored content in YouTube videos [8]. SponsorBlock crowd-sources timestamps for sponsored segments in YouTube videos, sometimes referred to as "ad reads," from its users, who in turn can automatically skip portions of videos that have been flagged as sponsored. A user-based voting system helps verify submitted timestamps; those with lower rankings are less likely to be reliable.

While this crowd-sourcing system works, our project aims to automatically detect sponsored segments in videos using deep neural networks. That way, a sponsor-blocking tool can skip sponsored segments for any video, not just ones that have previously been marked by another user. The precise problem statement is this: given the frames of a video, can a neural model accurately predict where the sponsored segment occurs? Our approach is to use an encoder-decoder architecture, where the Convolutional Neural Network (CNN) encoder produces an embedding for each frame of a YouTube video and the Recurent Neural Network (RNN) decoder predicts which frames correspond to the start and end of sponsored segments.

One noteworthy feature of our approach is that it uses video rather than audio or text data. While audio or text would correlate very strongly with whether part of a video is sponsored or not, as the vast majority of ad reads are driven by speech the resulting model would be language-dependent, which is less than ideal given that a majority of the content coming from popular channels on YouTube are in non-English languages [3]. On the other hand, a video-only approach is independent of language and thus could be viable with any video on the platform, not just the ones in English.

CS230: Deep Learning, Fall 2020, Stanford University, CA.

2 Related work

With NeuralBlock [4], Lee takes an alternate approach to this problem with promising preliminary results. NeuralBlock uses a time-annotated transcript of the video and a Bidirectional LSTM to classify whether or not a section of the transcript corresponds to a sponsored portion of video, achieving an accuracy of >90%. By finding sequences of substrings with high probabilities of being sponsored, Lee translates these predictions into timestamps for sponsored segment. One drawback to this approach is that it requires the intermediate step of getting a transcript for a given video, rather than being able make predictions from the video itself. It also relies on the video transcript being accurate; given that captions for most YouTube videos are automatically generated, this is not always a guarantee, especially for uncommon tokens like brand names.

On the other hand, our approach to the problem draws ideas from both Video Classification and Machine Comprehension. To analyze a video, we follow an approach similar to that of Ng et al. [5], who use an CNN encoder-LSTM decoder architecture to perform classification on longer videos. To produce timestamps as output, we draw from approaches to the Machine Reading Comprehension task posed by the Stanford Question Answering Dataset (SQuAD) [7], where the aim is to extract a section of a paragraph that answers a given question. In particular, we emulate the output scheme of Xiong et al. [12], who use two separate recurrent neural networks of identical architecture to predict the starting and ending token of the desired section, respectively.

3 Dataset and Features

Our dataset consists of the $\sim$36,000 video IDs from SponsorBlock with at least five upvotes on its sponsored segment. We divided our dataset into a train/dev/test split of 30,000/3,000/3,000. Each video ID in our dataset contains exactly one labelled sponsored segment, although we later found that some videos actually have multiple sponsored segments, with only one of them actually labelled. We then downloaded the frames (at 1 frame-per-second) of these videos from YouTube and labeled each frame as sponsored ($y = 1$) or non-sponsored ($y = 0$) by checking if its timestamp falls within the known sponsored segment of that video, as obtained from the SponsorBlock database. We assume that granular frame-by-frame motion is not necessary for distinguishing between sponsored and non-sponsored content, allowing us to sample videos at this low frame rate. In the training set of $\sim$30,000 videos, there are about 33.6 million frames total, about 1.5 million of which have positive labels.

4 Methods

4.1 Baseline

The baseline model is a binary classifier that predicts whether or not a single given frame is part of a sponsored segment, without paying attention to any temporal cues. The baseline is built upon a 50-layer residual network (ResNet-50) [2] pre-trained on the ImageNet [9] dataset and fine-tuned on the binary classification task of predicting whether a given frame is sponsored or not (see figure 2 for a diagram).

4.2 Encoder-Decoder

Our final model uses an encoder-decoder architecture to produce an encoding for each frame and then predict where the sponsored segment occurs given a sequence of frame embeddings.

The encoder is built upon the baseline CNN (see figure 3 for a diagram). To integrate it into our final encoder-decoder model, we replace the output layer with a series of fully-connected layers and batch-normalization layers that generate the final 300-dimensional embedding for each frame.

The decoder consists of two recurrent models, where each model is a two-layer bidirectional Long Short-Term Memory (LSTM) cells (see figure 4 for a diagram). One of the LSTMs is designated to predict the timestamps at which sponsored segments begin, and the other predicts the timestamps at which sponsored segments end. Both LSTMs take the frame embeddings produced by the CNN encoder as input. The outputs from each timestep for each LSTM are fed through a fully-connected

layer and finally a softmax layer, which produces a probability distribution over the whole video as to which frame is most likely to correspond to the start or end of the sponsored segment. The frame with highest probability is taken to be the starting or ending timestamp.

5 Experiments, Results, and Discussion

5.1 Training

To train the baseline model, we used a sample of ∼150,000 sponsored frames and ∼300,000 non-sponsored frames from our dataset. The sampling was necessitated by the prohibitive training speeds on the model. We used a Stochastic Gradient Descent optimizer with a starting learning rate $\alpha = 0.01$ and momentum factor of 0.9, which were commonly used parameters on classification problems with this model. We also applied a learning rate decay of $\gamma = 0.01$ every 7 epochs, and trained the model for 21 epochs where the train loss converged. The training process took around 3 days.

To train the encoder/decoder model, we defined the post-ResNet fully-connected layers of the encoder to be 2 linear layers with 512 neurons each, and we set the LSTM hidden state to have 2 layers with 256 neurons each. For training data, we randomly picked half of our training videos and pre-ran the baseline ResNet on all frames in the sample, storing the output vectors. We then used these as the training input for the decoder model (and the post-ResNet FC layers of the encoder). Due to lack of time for hyperparameter tuning, we chose an Adam optimizer with $\alpha = 0.01, \beta_1 = 0.9, \beta_2 = 0.999$, for the ability to quickly converge to a reasonable result without much need for tuning. We also applied a learning rate decay of $\gamma = 0.01$ every 7 epochs, and trained the model for 19 epochs where the train loss converged. The training process took around 20 hours.

5.2 Evaluation Metric

Intuitively, the more overlap there is between the duration of the predicted sponsored segment and the ground-truth timestamps, the better the prediction is. Thus, to evaluate the quality of a given prediction, we use a 1-dimensional intersection-over-union (IOU) metric. For the baseline, which makes separate predictions for each frame, the IOU is computed as follows:

$$IOU = \frac{\sum(pred_{mh} \& gt_{mh})}{\sum(pred_{mh} \parallel gt_{mh})}$$

where $pred_{mh}$ and gt_{mh} are multi-hot representations of the video with 1's for sponsored frames and 0's for non-sponsored frames as given by the prediction and ground-truth, respectively.

For the encoder-decoder model, which outputs timestamps, the IOU is computed as follows:

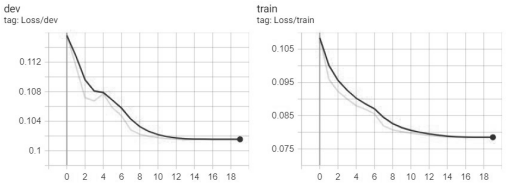
$$IOU = \frac{\max(0, \min(end_{pred}, end_{gt}))}{len_{pred} + len_{gt} - \max(0, \min(end_{pred}, end_{gt}))}$$

where end_{pred} and end_{gt} are the ending timestamps of the prediction and ground-truth, respectively, and len_{pred} and len_{gt} are the durations of the predicted segment and ground-truth, respectively. The optimal value for IOU is 1, when the prediction aligns perfectly with the ground-truth.

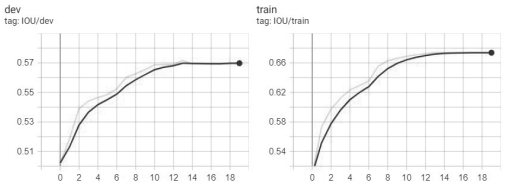
5.3 Performance

After training the baseline and the encoder-decoder models, we can see from table 1 and figure 1c, that training on the baseline model converged with an accuracy greater than 0.99 and a development set accuracy of 0.79. Here, accuracy is on a frame-level basis, where each frame is independently evaluated to determine whether it is part of a sponsored segment or not. Note that approximately 70% of the examples in the dataset were non-sponsored segment frames, indicating that while the model did learn something, the large difference between the training and development set accuracies suggests overfitting on the training set.

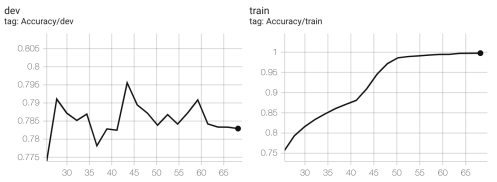
Meanwhile, for the encoder-decoder model, we can see from figures 1a and 1b, that as the training loss converged, the loss on the development set also converged. Correspondingly, the training set IOU of the model converged to 0.66 and development set IOU converged to 0.54 (table 1). We use IOU as a proxy for accuracy here, since the encoder-decoder model worked over videos while the



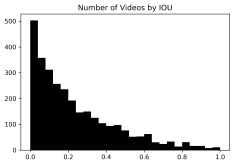
(a) Loss curves for the training and development sets on the encoder-decoder model.



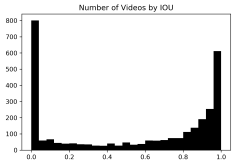
(b) IOU curves for the training and development sets on the encoder-decoder model.



(c) Training and development set accuracies for the baseline model.



(d) Histogram of IOUs for the baseline model on the test set.



(e) Histogram of IOUs for the encoder-decoder model on the test set.

	Baseline (Accuracy)	**Encoder-Decoder (IOU)**
Training Set	> 0.99	0.66
Development Set	0.79	0.57

Table 1: Training and development set metrics on the baseline and encoder-decoder models.

baseline model worked on individual frames. Note that since the training set IOU is higher than the development set IOU, this indicates some overfitting to the training set, but to a much smaller degree than the baseline model.

	Baseline	**Encoder-Decoder**
Mean	0.23	0.54
Median	0.17	0.69

Table 2: Summary statistics for test set IOUs.

Finally we evaluate the IOUs of both models on the test set to standardize our comparisons (table 2). Here, we can clearly see the encoder-decoder model with a median IOU of almost 0.7 outperforming the baseline model with a median IOU of less than 0.2. Median is the summary statistic of choice here, since the distribution of IOUs for both models are not normal (figures 1d and 1e). For the baseline model, we can plainly see the results of overfitting; most IOUs are very low and the distribution is heavily skewed to the right. However, the distribution for the encoder-decoder model is sharply bimodal. The fact that one peak is exactly at a zero-IOU while the curve grows more steady towards an IOU of one warrants further analysis (see below).

5.4 Error Analysis

Because the IOU distribution for the final model on the test set had so many 0's, we investigated at a random sample of 16 test examples that had an IOU of 0 and classified them based on the following failure modes:

1. Video had *multiple* sponsored segments, found one of the unlabeled segments
2. Video had *multiple* sponsored segments, predicted timestamps from different segments
3. SponsorBlock's timestamps don't actually correspond to a sponsored segment
4. Sponsored segment not visually apparent (i.e. Bayes error)
5. Model predicted a meaningful but non-sponsored segment (e.g. a normal intro or outro)
6. Other

As shown in Table 3, of our sample of 0-IOU videos, half of the them can be attributed to incorrect or incomplete SponsorBlock labels, which suggests that a significant portion of our test examples that achieved 0 IOU are not due to our model not being able to properly detect sponsored segments. Furthermore, another three of the poorly-predicted examples result from the sponsored segments being visually indistinguishable from the rest of the video, which a video-only approach would never be expected to perform well on; this falls under the Bayes Error for this problem. Only five of the 16 examples were strictly due to the model failing to properly detect a sponsored segment.

Source of Error	Failure Mode	# of Videos	Total # of Videos
Dataset Error	1	4	8
	2	2	
	3	2	
Bayes Error	4	3	3
Model Error	5	3	5
	6	2	

Table 3: Failure modes for a random sample of 16 test examples that achieved 0 IOU with our full model.

6 Conclusion/Future Work

In this paper, we present a CNN-encoder/RNN-decoder model to predict the occurrence of sponsored segments in YouTube videos. Our model achieves a median IOU of 0.69 across our test examples, indicating a generally high degree of overlap between the predicted sponsored segments and the ground-truth. Furthermore, analysis of the badly-predicted test examples shows that much of the poor performance stems from incorrect or incomplete labels in our dataset or unavoidable bias.

The next steps for this project include relaxing the one-sponsored-segment assumption and adjusting the model architecture to allow for predicting multiple sponsored segments or no sponsored segments. Another direction of improvement would be to incorporate audio data in addition to video data to help combat examples where the sponsored segment is visually indiscernible; extra care would have to be taken to allow such a model to work for a variety of languages. Once a robust model is developed, the final step would be to build a browser extension, or some other tool, to deploy the model and allow users to easily skip sponsored content in any YouTube video they watch.

7 Contributions

All group members met weekly throughout the project period for progress discussion, brainstorming, and group/pair programming. The approach (the ResNet transfer baseline and the Encoder/Decoder) was discussed and developed with equal contribution from all members, and most of the model and training code were written in group programming sessions.

Nikhil additionally contributed to the dataset downloader, wrote some sections of the proposal (introduction), milestone (discussion about the encoder-decoder model's approach), and final paper (discussion of results), and primarily authored the presentation for the video (e.g. select YouTube videos to display, create diagrams for the models, write text on slides, etc.).

Cem additionally wrote most of the dataset downloader; managed, tuned and monitored the AWS instances through the iterations of the download, training, and evaluation processes; cleaned up and maintained codebase on GitHub; and contributed the Training section to the paper as well as the raw data from the training and evaluation.

Jennie additionally wrote the bulk of the final report (abstract, intro, related work, dataset, methods, eval metric, error analysis, conclusion) as well as much of the proposal and milestone. She also did much of the literature review for the final paper as well as the error analysis. Finally, the fake Audible sponsorship in the video was definitely her idea.

References

[1] Raymond Gorhill. ublock. `https://github.com/gorhill/ublock`.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Patrick Van Kessel, Skye Toor, and Aaron Smith. A week in the life of popular youtube channels, 2019.

[4] Andrew Lee. Neuralblock. `https://github.com/andrewzlee/NeuralBlock`.

[5] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification, 2015.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[8] Ajay Ramachandran. Sponsorblock. `https://github.com/ajayyy/SponsorBlock`.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[10] Carsten Schwemmer and Sandra Ziewiecki. Social media sellout: The increasing role of product promotion on YouTube. *Social Media + Society*, 4(3):205630511878672, July 2018.

[11] Huan-Hsin Tseng. Video classification. `https://github.com/HHTseng/video-classification`.

[12] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering, 2018.

8 Appendix

8.1 Implementation

The architecture was implemented using the PyTorch [6] library in Python and is available open-source at https://github.com/DeepSponsorBlock/DeepSponsorBlock.

8.2 Test Set Examples

In Table 4, we present a small sample of test set videos and describe our model's performance on those videos. (These are the same videos we showed in our video presentation.)

Video ID	Pred segment	Labeled segment	IOU	Description
-Sa6B1AkFwc	(36, 49)	(36, 49)	1	Correct.
-e4BvZz6H8Q	(68, 79)	(589, 623)	0	Discovered unlabelled sponsored segment
-O42UHl_6Js	(0, 12)	(434, 468)	0	Mistook a visually distinct intro for the sponsored segment
03_uwi_zrcI	(174, 182)	(173, 207)	0.235	Switches back to pre-sponsored visual content before the sponsored segment ends

Table 4: A sample of four test videos with varying quality of predictions.

8.3 Error Analysis

Table 5 shows our full error-analysis for the 16 test examples described in the Error Analysis section. All of these examples had an IOU of 0.

Video ID	Pred	Label	Mode	Notes
-e4BvZz6H8Q	(68, 79)	(589, 623)	1	Discovered a new sponsored segment
9wQFeiR_U_U	(564, 594)	(0, 10)	1	Discovered a new sponsored segment
mp__Bx0aodw	(589, 641)	(12, 20)	1	Discovered a new sponsored segment
viNw-H3SloQ	(818, 934)	(1, 11)	1	Discovered a new sponsored segment
pb8edcvgjX0	(0, 3)	(323, 385)	1	Predicted the part in the beginning where they say "this video is sponsored by NordVPN"
EELBDtM3FGs	(632, 5)	(0, 5)	2	Found new sponsored segment but mixed up their timestamps
-PpU6Mh024g	(0, 3)	(63, 70)	3	SponsorBlock's label is actually for the "like and subscribe" portion
xbAV4dO8gvM	(14, 270)	(0, 11)	3	SponsorBlock's label is for a non-sponsored intro
-v_8ed2_2xc	(247, 260)	(114, 133)	4	
-O42UHl_6Js	(0, 12)	(434, 468)	4	
jO1AqGyjlLc	(181, 739)	(20, 35)	4	
-x9ASGFJvjg	(0, 4)	(1312, 1359)	5	Predicted a segment that looks more like sponsored content than the actual sponsored content (which looks like the rest of the video)
K3oXD-WNHzs	(0, 17)	(22, 50)	5	Predicted the intro
HTnHhw9K434	(1030, 1055)	(901, 995)	5	Predicted the outro
IWgL-eozo7U	(234, 613)	(614, 720)	6	
xRhsJhhSe6Q	(460, 77)	(15, 32)	6	Visually hard for human to tell what's going on in this video

Table 5: Full results from our error analysis. The "Mode" column corresponds to the enumerated failure modes from the Error Analysis section of the main paper.

Model Diagrams

Baseline

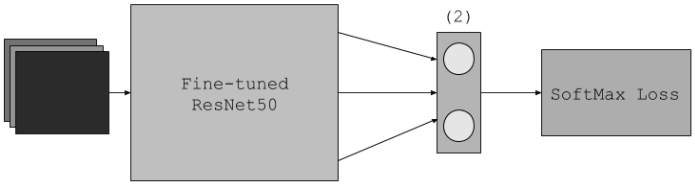


Figure 2: Model architecture for the baseline model.

Encoder

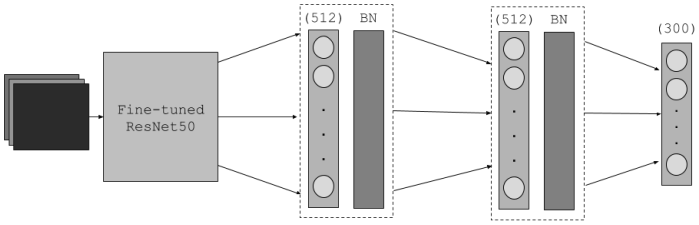


Figure 3: Model architecture for the encoder model.

Decoder

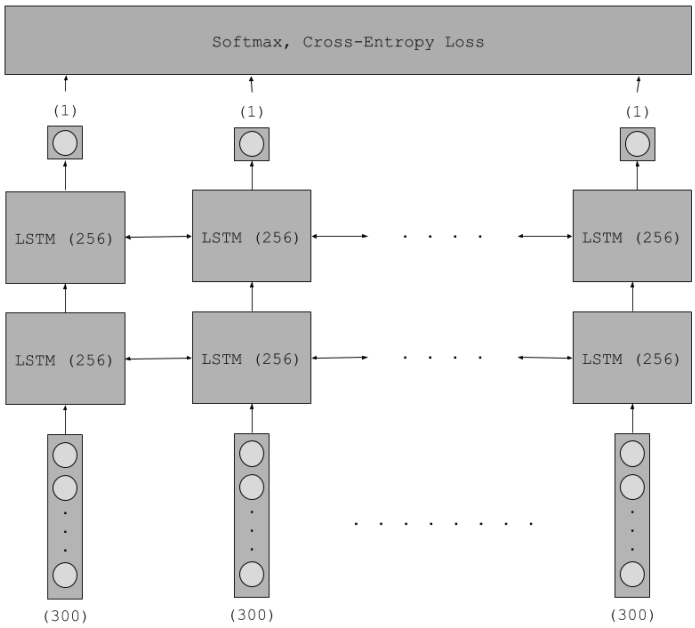


Figure 4: Model architecture diagram for the decoder model. Note the 300-element input vector for each frame comes from the encoder.