Towards Intelligent Ad-Blockers: Exploring deep learning techniques to classify website images as Ads

Abstract: Ad Blockers play a vital role by regulating entire online advertising market. Current ad-blockers still use a list of popular ad providers and block content from them. To build next generation of intelligent ad-blockers, we need to have a classifier which can dynamically detect ads. Ads don't have any perceptible characteristic features which identify them uniquely. Hence, Deep learning might be a powerful tool for classification than regular rule-based classifiers. In this project, I have built a new online images dataset by designing a crawler & then scrapping images from the internet. Images are classified as ads based on easylist's list of adservers. Finally, I am using Imagenet based frameworks such as Inception & Alexnet to classify images as ad or not.

1 Introduction: Online advertising is a billion-dollar industry. It is a major source of income for companies such as google, facebook, etc. Some websites go out of the way to earn more revenue by enabling popups, long video ads, security alerts and doing everything possible to attract user's attention. To keep a check on such websites, Users are rapidly adopting adblockers. AdBlockers are basically browser plugins that oversee the communication between browser and web server. Every adblocker maintains a list of filters to identify if a resource being fetched is an ad or not. Adblockers enhance user online experience by blocking ads, saves bandwidth and even prevents malware to an extent.

1.1 Motivation: Adblockers are maintained by a group of individuals who add a new filter for every new ad server or any new kind of ad encountered or reported. They work by matching URL patterns embedded in a webpage against a set of blacklist patterns, and cannot look deeper into the element and reason about it. The task is made even more difficult by complex DOM structures, deep nesting of elements, and dynamic JavaScript execution, that is found on a large fraction of pages on the Internet today. Thus this process is human intensive, time consuming & mayn't be effective. If we could develop a mechanism to detect ads automatically, it can revolutionize the way Ad blockers are built today. Adblockers are installed in over 615 million devices and this count increased by 30% just from last year [9]. Developing such next gen adblockers can have a huge impact given the number of users using it.

1.2 Complexity: When we think about ads, there is no unique picture that comes to our mind. Unlike real world objects like chair, car, truck, dog, cat, etc., Ads don't have a perceptible unique feature which is present in all ads. Following figure shows some categories of ads like car, shoes, tourism, software, etc. Ads can vary in size. They may or mayn't contain images/text. Text in ads can be of any language. Ads can be an image, animation or video. They can vary in brightness or contrast. To capture such diverse elements on webpages across the internet is a herculean task.

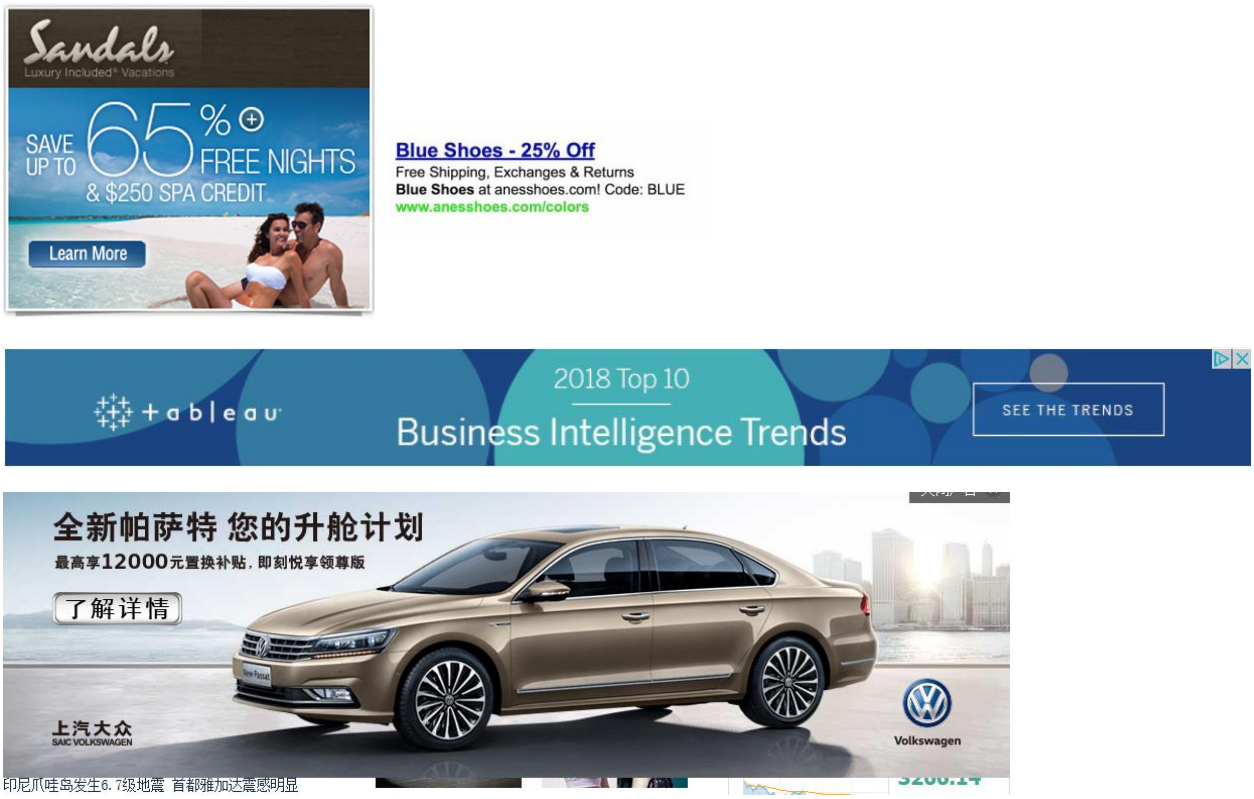

Fig 1: Type of ads

2 Related Work: A significant amount of literature on online ads is focused on increasing Ads Click-through rate [3,4]. Fire used deep learning to predict success of an ad [5]. Previous work on ads classification is rule based. There's a US patent in which ads are identified by extracting features from image using OCR. There are other rule based methods which analyze landing pages to detect ads [6]. There's not much work on automated ads classification using deep learning. This makes this problem more interesting.

3 Methodology

The first step was to collect dataset. I used two approaches to extract images: -

1) Beautiful Soup with urllib2
2) Selenium with Firefox

Based on the assumption that ad images are represented by img HTML tag, I wrote a crawler using urllib2 library to download webpage and beautiful soup to parse images from the page. After spending hours, I could download 10k images. I downloaded webpages based on Alexa [1] top ranked pages and quantcast [2] top webpages in US. Thereafter, I tried to classify images based on their url using easylist's list of adservers [7]. To my surprise, less than 1% of images were classified as ads. Based on my online experience, I was expecting a much bigger number. I investigated two major reasons for this small proportion of ads.

1) A majority of ads are represented as iframe tag instead of img tag.
2) Webservers can detect if a python script is accessing their server so they don't serve the same ads as they do for a browser.

iFrame is an html page embedded inside of parent html page. We can't use the same mechanism to extract images as we did for img tag. This is because we can't access iframe in a separate request. Iframes can be loaded only as a part of websites in which it is embedded.

To counter the second issue, I used Selenium over Firefox. Selenium literally runs firefox browser for every request and allows simulation user's action like clicking, filling form, etc. via code. The downside of this approach is that it's computationally expensive and very slow. Earlier approach was much faster but didn't downloaded ads. I decided to use best of both worlds. I used the first approach to download non-ad images and second approach to download ads.

***3.1 Data**:* After resolving both issues I was able to download 6500+ images from top 150+ websites ranked by Alexa. Based on the url of each image, it was classified as ad image or not.

3.2 Preprocessing

Internet is a noisy place so some images were corrupt and some were duplicate as well. All webpages strive to have less download time so many of the images were very small like <1kb. I filtered out all images whose height or width was less than 100. Secondly, I used md5 hash to find out which images were duplicate and filtered them out as well. Lastly, I use PIL library to figure out which image is corrupt. After performing all these operation I was left with around 1650 images in which around 400 were ads.

4. Results

Finally, we used deep learning models to train and test on our dataset. The problem is essentially a binary classification problem with two classes ad or no ads. As learned in last assignement, I used transfer learning based on Imagenet dataset. Since online images can be varied so imagenet might be a good fit. I used two frameworks:- Inception v3 and Alexnet. For both of these frameworks, I fine-tuned only the last fully connected layer and kept other layers frozen.

Alexnet	Inception
learning_rate = 0.01 num_epochs = 10 batch_size = 128 dropout_rate = 0.5 num_classes = 2 train_layers = ['fc8']	Learning_rate = 0.01 Training steps = 5000 Train batch size = 100 Test batch size = 1 Testing percentage = 10% Validation percentage = 10%

I got the following validation error for both models:-

AlexNet **95.193%**

Inception **94.6%**

5. Limitations:

We are only looking for main page of a website and not for subsequent pages. Subsequent pages might have ads and main page mightn't have any ads like google.com. So, we might get only a small proportion of ads from such websites. In the future, we should build a more robust crawler which can visit more pages from a website.

We are using a python script to act as browser so web servers can detect that & mightn't serve the same ads as for chrome, firefox, etc. To counter this, we started using selenium in the second part of data collection to get more realistic results.

We have achieved good accuracy but on a relatively small sample. We need to test our model on large and even more diverse datasets to make a difference. Furthermore, we should also consider animation, videos, etc.

6. Future Work

- Parameters like size of image, position of image, source url of image can also be fed along with image itself to increase the accuracy further.
- Make the process of extraction of images more robust to different ways in which an image can be coded in the HTML DOM.
- Currently, we identify whether an image is ad by using Easylist's list of adservers. We can also try to use AdBlocker plus list of filters which are parsed in a different fashion.

7. Conclusion: This project was a great learning experience. For the first time in months, I disabled my ad blocker to explore other side of the world. Contrary to my initial assumptions, Websites employ numerous coding techniques to display similar looking ads. So, it's difficult to design an extraction mechanism which fits all websites. I explored the limitations of python libraries such as urllib2 & request. Learnt & used Selenium for the first time. Explored tensorflow further & got a chance to work with sophisticated deep learning architectures such as Alexnet & Inception.

Deep learning techniques have not been applied to identify ads before. I was skeptical that deep learning algorithms might not give good results because unlike dogs, cats, etc. ads don't possess any perceptible unique features. Out of curiosity, I tried solving this seemingly difficult problem with deep learning algorithms. I am glad I attempted this problem and I am very happy with results. This project has motivated me to pursue this novel problem further with even larger dataset and more sophisticated deep learning algorithms with additional features.

References

[1] Alexa top 1 million websites list: http://www.seobook.com/download-alexa-top-1-000-000-websites-free. Accessed 10 Dec. 2017

[2] Top websites in US https://www.quantcast.com/top-sites/ Accessed 14th Dec. 2017

[3] Roffo, G., & Vinciarelli, A. (2016, August). Personality in computational advertising: A benchmark. In 4 th Workshop on Emotions and Personality in Personalized Systems (EMPIRE) 2016 (p. 18). (https://www.kaggle.com/groffo/ads16-dataset Accessed 15th Dec. 2017)

[4] Liu, Bin, et al. "AdReveal: improving transparency into online targeted advertising." *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*. ACM, 2013.

[5] Fire, Michael, and Jonathan Schler. "Exploring Online Ad Images Using a Deep Convolutional Neural Network Approach." arXiv preprint arXiv:1509.00568 (2015).

[6] Kae, Andrew, et al. "Categorization of display ads using image and landing page features." Proceedings of the Third Workshop on Large Scale Data Mining: Theory and Applications. ACM, 2011.

[7] List of Adservers by Easylist https://easylist.to/ Accessed 10 Dec. 2017

[8] Inception v3 model https://github.com/xuetsing/image-classification-tensorflow/blob/master/train.py Accessed 14 Dec. 2017.

[9] Facts about Ad Blocker usage https://pagefair.com/blog/2017/adblockreport/ Accessed 15th Dec. 2017