Deep Learning-based Model to Fight Against Ad Click Fraud

Thejas G.S.
Florida International University
Miami, FL, USA
tgs001@fiu.edu

Kianoosh G. Boroojeni
Florida International University
Miami, FL, USA
kgholami@cs.fiu.edu

Kshitij Chandna
Siddaganga Institute of Technology
Tumkur, Karnataka, India
kshitijchandna@gmail.com

Isha Bhatia
R.V.College of Engineering
Bangalore, Karnataka, India
ishabhatia97@gmail.com

S.S. Iyengar
Florida International University
Miami, FL, USA
iyengar@cs.fiu.edu

N.R. Sunitha
Siddaganga Institute of Technology
Tumkur, Karnataka, India
nrsunitha@sit.ac.in

ABSTRACT

Click fraud is a fast-growing cyber-criminal activity with the aim of deceptively clicking on the advertisements to make the profit to the publisher or cause loss to the advertiser. Due to the popularity of smartphones since the last decade, most of the modern-day advertisement businesses have been shifting their focus toward mobile platforms. Nowadays, in-app advertisement on mobile platforms is the most targeted victim of click fraud. Malicious entities launch attacks by clicking ads to artificially increase the click rates of specific ads without the intention of using them for legitimate purposes. The fraud clicks are supposed to be caught by the ad providers as part of their service to the advertisers; however, there is a lack of research in the current literature for addressing and evaluating different techniques of click fraud detection and prevention. Another challenge toward click fraud detection is that the attack model can itself be an active learning system (smart attacker) with the aim of actively misleading the training process of fraud detection model via polluting the training data. In this paper, we propose a deep-learning based model to address the challenges as mentioned above. The model is a hybrid of artificial neural network (ANN), auto-encoder and semi-supervised generative adversarial network (GAN). Our proposed approach triumphs excellent accuracy than other models.

CCS CONCEPTS

• **Information systems** → *Online advertising*; • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Click Fraud, Invalid Click, Mobile Advertisement, Botnets, Click Spam, Deep Learning, Adversarial Environment

ACM Reference Format:
Thejas G.S., Kianoosh G. Boroojeni, Kshitij Chandna, Isha Bhatia, S.S. Iyengar, and N.R. Sunitha. 2019. Deep Learning-based Model to Fight Against Ad Click Fraud. In *2019 ACM Southeast Conference (ACMSE 2019), April 18–20, 2019, Kennesaw, GA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299815.3314453

ACMSE 2019, April 18–20, 2019, Kennesaw, GA, USA

ACM ISBN 978-1-4503-6251-1/19/04.
https://doi.org/10.1145/3299815.3314453

1 INTRODUCTION

With the invent of smart-phones, people have started using it addictively in their daily life. On the other hands, as the digital advertising industry business model rely on the people usage of computing devices, even they have shifted their concentration on the mobile platform with the transformation of promoting ads through in-app advertisement [26]. In their business model, "click and impression" plays a significant role as it is one of the parameters to calculate the payment for the publisher by the ad network. Whenever a person clicks an ad, then it will be recorded by the ad network as a count that the number of times the ad was clicked. In the same manner, whenever the ad loaded in the user devices, then it will be recorded by the ad network as a count that the number of times the ad was loaded [15]. Most common parameters that are considered by the ad network to calculate the payment for the publisher are: pay per click, cost per click, click through rate, average ads position, cost per mile, conversion rate and cost per conversion.

However, in the business revenue model there exists a security problem, i.e., click fraud. Of course, the typical way of interacting with the computing device is through clicks. For example, we click an app to open it, we click a hyperlink to redirect to the destination page, or we click on like icon on social network posts or YouTube videos to express our emotion, we click on exciting ads to explore them, etc. [31]. In all the above-said example the click operation plays a vital role in the expectation of genuine purpose. Because too many clicks on a social network post say that the post is becoming prevalent and too many clicks on the ad, show the demand on the product that is promoted through ads. It is not apparent to believe that all clicks are valid (legitimate) or fraudulent clicks. There is no protocol defined to authenticate these clicks based on trust [30]. Click fraud is nothing but the action that happens by fraudulent clicks where the click made on an ad without the intention of using them for legitimate purposes. Types of attacks to perform click fraud are as follows: *Brute force attack:* One of the straightforward ways to perform click fraud is by using a single computing machine (computer/laptop/mobile). This attack can be as basic as a person physically tapping on an advertisement again and again. *Attacks through crowdsourcing:* Here publishers deploy the ad in their app by mentioning some help or support for a cause which makes the audience to click the ad with mercy. *Attacks through reward traffic:*

Rewarding the person for clicking the ad is the next level of click fraud that can generate high traffic of clicks. Here reward could be in the form of providing discount coupon codes or bonus points to play some games. *Click farm attack:* A publisher influences a group of people to generate click traffic on his or her ads in exchange for small profitable money from the ads. *Hit inflation attacks:* In these attacks, a legitimate user click traffic is redirected to the ad unknowingly. *Botnet attacks:* Some professional attackers use malware to infect a large number of computing devices. From there onwards, the attacker can instruct the botnet to perform repetitive clicks by installing apps in the background without the owner's knowledge [34].

Here, an advertiser is the one who wants to make a profit from his/her business or product by mean of promoting it to the world. These promotions are done in the form of advertisement on websites or in-app platforms. Hence, advertisers pay ad networks to promote their ads on the websites or in-app. The publisher is the one who develops the website or mobile application and deploys ads on them. Ad Network provides a platform for advertiser and publisher. They are responsible for receiving an advertisement request and required payment form the advertiser for promotion. They are also responsible for approving the ad publishing request from the publisher, allowing them to deploy ads on their website or in-app. With these, the ad network is also responsible for evaluating the clicks made on ads and making payment to all valid clicks. Detection/Prevention models or the learning-based models or manual analysis are used by ad networks to evaluate the clicks as either valid or invalid. Here clicks are made by Clickers who can be the regular users, i.e., clicks with genuine interest, or fraud motive users like click farms, botnets or smart attacker. The smart attacker is the one who misleads the learning models to perform wrongly via polluting the training model and directing the other real-time attacks [2–5, 25].

Summary of Contribution We propose a way to handle the imbalanced dataset based on our pre-analysis on attributes like ip and app id. We simulated learning models like logistic regression, random forest, naive bayes, and support vector machine from which it concludes that logistic regression and random forest competitively performs well. We developed a hybrid model consisting of Semi-supervised Generative Adversarial Network (GAN), Auto Encoder and Neural Network to solve the problem of fake clicks in adversarial environment. We used the semi-supervised GAN to create adversarial attacks in order to improve the accuracy of the Neural Network. To the best of our knowledge, we are the first to use $Cos(\theta)$, as a supervised loss in a semi-supervised GAN and the hybrid model performs better than other models.

Organization of Paper in Sect. 2 we review the related work on click fraud in ad networks, in Sect. 3 we describe the dataset characteristics, challenges and experimental setup, in Sect. 4 we discuss and present the preliminary results obtained on Logistic Regression, Support Vector Machine, Random Forest, and Naive Bayes, in Sect. 5 we propose our deep learning approach and provide details about the hybrid model, building blocks of the hybrid model, attacker model GAN and discussions on the presented results, and in Sect. 6 we conclude our discussion.

2 RELATED WORKS

Research shows that the more asymmetric in quality the ad networks are, the more asymmetric their equilibrium prices will be[10]. A study showed that one of the largest click fraud botnets, called ZeroAccess, induces advertising losses on the order of $100,000 per day [23].

Non-Statistical Models: In [14], they take the average clicks on bluff ads as a way of discrimination; In [7], they use CAPTCHAs to make sure that the user is real. In [11], they used Social Network Analysis to find top three ad networks that were being used to spread the fraud click malware. In [1], they use Splay trees to store the IPs via which fraud clicks occur based on a burst. In [22], they first find the eigenvalues of displayed ad images, if the ad is shown it is attested, based on if the eigenvalues of the image match those stored in their server, the user is certified as honest, if not then it is analysed. Researchers build on several published theoretical results to devise the Similarity-Seeker algorithm that discovers coalitions made by pairs of fraudsters [24].

Statistical Models: These use sophisticated statistical models to find out which IP addresses are behind fraud attacks [21] or analyse periodic activity of DNS to find out nefarious activities. In [8], they take certain features of the Android app and use machine learning to flag fake ads and in [9], they find gold standard users to find probability of fraud. In [19], they use reverse ad algorithm which checks whether a system is a robot or not. In [20], the research deals in the earlier stages of working with click fraud, it deals with the selection of appropriate features for best results. In [27], deals with the usage of ensemble model cascading gradient boosting model. By comparing all the related works and to the best of our knowledge, our work is the very first attempt done with deep learning hybrid model.

By comparing all the related works and to the best of our knowledge, our work is the very first attempt done with deep learning hybrid model consisting of ANN, auto-encoder, and semi-supervised GAN.

3 DATASET CHARACTERISTICS, CHALLENGES AND EXPERIMENTAL SETUP

Dataset Characteristics: For our experiment, we have used real-time dataset provided by kaggle [16]. The data were collected for four days (2017/11/06 to 2017/11/09). The dataset contains 184903890 number of real-time ad clicks observations collected on a mobile platform. It contains eight attributes in which seven are features (independent attributes), and one is a label (dependent attribute). The overall size of the dataset is around 7 GB. Table 1 describes the characteristics of the dataset. The dataset consists of 277396 unique ip's, 706 unique app id's, 202 unique channel id's, 3475 unique device id's, and 800 unique os version id's. All these are encoded due to the privacy factor. The Figure (1) shows the histogram plots on unique ip's and app id's. The plot in Figure (1a) shows fake clicks ratio per app id where 90 percent of the clicks generated are suspicious, the plot in Figure (1b) shows the number of observation per famous app id's where there are 5 app id's which has 5.1 (+ or -) 1.7 millions of observations and the plot in Figure (1c) shows ip's participation in fake and valid clicks. Figure (1d) shows the heat map generated for each attribute compiling the null values in the

dataset. We see areas with lighter blank spaces which consist of null values. Hence, we drop attributes with maximum null values to have a clean dataset. We see that attributed time has maximum null values. Hence we drop the column and achieve a clean dataset.

Dataset Challenges: The vast dataset is not a problem, actually very beneficial, so long as it is evenly distributed. However, our dataset was overwhelmingly slanted towards negative or fake clicks, which created a problem in determining accuracy, i.e. even if we do not predict a single positive or real click, it will still give approximately 100% accuracy. However, we did not get 100% accuracy in the case of deep learning model implementation when we took an evenly distributed dataset. For this kind of distribution, we were able to achieve an accuracy of 94-95%. Based on ip address of click and app id for marketing we divide the dataset into six classes as as follows: *Class 1: Hypo-active users of non-suspicious apps:* Combination of observations with unique ip participation count less than 20 times and app id frequency of participation less than 70% with 25974 rows. *Class 2: Active users of non-suspicious apps:* Combination of observations with unique ip participation count in the range (greater than or equal to 20 times and less than 1000 times) and app id frequency of participation less than 70% with 27174 rows. *Class 3: Hyperactive users of non-suspicious apps:* Combination of observations with unique ip participation count greater than or equal to 1000 times and app id frequency of participation less than 70% with 112790 rows. *Class 4: Hypo-active users of suspicious apps:* Combination of observations with unique ip participation count less than 20 times and app id frequency of participation greater than or equal to 70% with 784964 rows. *Class 5: Active users of suspicious apps:* Combination of observations with unique ip participation count in the range (greater than or equal to 20 times and less than 1000 times) and app id frequency of participation greater than or equal to 70% with 19914810 rows. *Class 6: Hyperactive users of suspicious apps:* Combination of observations with unique ip participation count greater than or equal to 1000 times and app id frequency of participation greater than or equal to 70% with 164038178 rows.

Experimental Setup: All methods and models are experimented on Intel Xeon 8 cores CPU with 32 GB RAM, 100 GB SSD and Tesla K80 GPU with 12 GB memory. Implementation is done in python where train to test data ratio is 3:2 by randomly selecting the rows in the dataset.

4 PRELIMINARY EXPERIMENTS

In order to analyze the relation between the dependent and independent attributes we train and test the following prediction models: Logistic Regression (*LR*), Support Vector Machine (*SVM*), Random Forest (*RF*) and Multinomial Naive Bayes (NB_M). *Performance matrices:* To evaluate the performance of each model we have considered Precision, Recall Accuracy. *Observations:* From Table 2, we can state that RF performs better with minimal error rate and gives us the highest accuracy rate.

5 DEEP LEARNING APPROACH

After having some degree of success in the above approaches, we wanted to develop an algorithm to detect and discard fake clicks in real time. For that, we used Deep Learning methods, which also substantially improved our accuracy. We built a multi-layered Neural Network with an attached autoencoder using Keras backend Tensorflow [6, 28]. We developed an approach based on Replicator Neural Network [32] for detecting bots. Due to a heavily imbalanced dataset, we used a semi-supervised GAN to generate fake data in order to act as an attack and also increase the accuracy of our Neural Network. Figure 2 depicts our hybrid deep learning model.

Work Flow: (1) User clicks on the link; the info is saved and sent to an autoencoder, (2) A trained Autoencoder regenerates the data after adding some noise to it, (3) If regeneration loss is more than the threshold, the user is discarded as a bot, (4) If not, the user is considered human and his/her info is passed to a neural network, and (5) The Neural Network predicts whether the user has an intention of downloading the app or not.

Below the concepts behind our scheme are explained. We use the dataset to train both the supervised Neural Network and Unsupervised Autoencoder. Since the data used is based on human clicks, it is easy to train the autoencoder to recognize human behavior and discard the bots. Our model follows the pattern of Batch Normalization after every Dense layer. To stop the model from overfitting, we put a dropout after every layer with the probability of 0.2 [29]. We initialize the parameters of the hidden layers using Xavier[12] and and trained the network using adam[17]. Table 3 shows the results.

5.1 Auto Encoder

We trained the autoencoder [33] on a dataset of 3 million humans. Since the autoencoder is unsupervised, it cannot predict whether the user is real or fake. However, the autoencoder was trained to regenerate the data. It learned the distribution of data for which it was trained. For example, if an autoencoder that has only been trained on the human dataset and the Regeneration loss for click is higher than a decided threshold, then we can discard that as a bot. Since we did not have the data classified as bots and humans, therefore we could not show how well this method will work. However, a similar technique was used by Veeramachaneni et al. [32] with their Replicator Neural Network. They got some great results for detecting frauds based on regeneration error.

5.2 Semi-supervised GAN

We develop a semi-supervised GAN [13, 18] to generate fake samples as a smart attacker for the Neural Network.

5.2.1 Generator. In Figure 3, we show how we take care of generating different attributes.

$$x_{noise} = uniform\_distribution(-1, 1) \tag{1}$$

$$x_{generated}, x_{generated\_logits} = generator(x_{noise}) \tag{2}$$

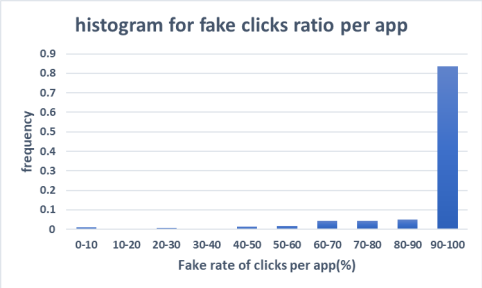
After the $softmax$ function is applied to each attribute vector, $max_one_hot_encode$ to take the highest value of vector and make it 1, while others are converted to zero. We also concatenate the vectors with $softmax$ activation: $x_{generated\_logits}$.

5.2.2 Discriminator. It gives two outputs, $discriminator\_1$ value denotes whether the given data is real or generated, $discriminator\_2$ value denotes whether the given data indicates a valid click or not.

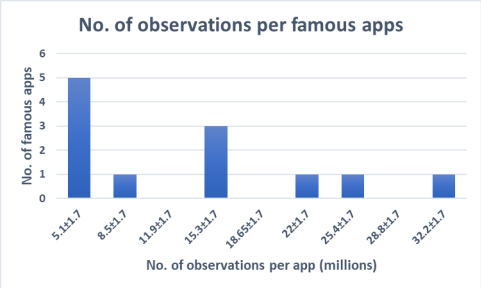
$$x_{g\_out} = discriminator(x_{generated}) \tag{3}$$

Table 1: Characteristics of the Ad Click Dataset

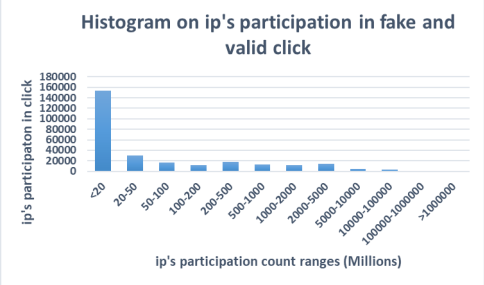
Attributes	Description	Attributes	Description
ip	Ip address of click	app	App id for marketing
device	Device type id of user mobile phone (example: iPhone 6, iPhone 6 plus, Samsung Galaxy 8, etc.)	attributed_time	If user download the app for after clicking an ad, this is the time of the app download
channel	Channel id of mobile ad publisher	os	Os version id of user mobile phone
Is_attributed	The target that is to be predicted, indicating the app was downloaded	click_time	The time stamp of click (UTC)



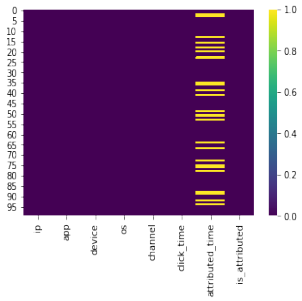
(a) Histogram for Fake Clicks Ratio Per App



(b) No. of Observations Per Famous App's



(c) Histogram on Ip's Participation in Fake and Valid Clicks



(d) Heat Map Portraying Null Values

Figure 1: Pre-Analysis on the Dataset with Respect to Ip's and App Id's

Table 2: Preliminary Results

Models	LR			SVM			RF			NB_M		
Performance Metric	**P**	**R**	**A(%)**	**P**	**R**	**A(%)**	**P**	**R**	**A(%)**	**P**	**R**	**A(%)**
Hypo-active users of non-suspicious apps	0.18	0.70	71.33	0.13	0.75	70.87	0.46	0.61	73.43	0.22	0.57	69.74
Active users of non-suspicious apps	0.42	0.57	64.23	0.13	0.87	64.60	0.64	0.69	74.26	0.25	0.61	64.00
Hyperactive users of non-suspicious apps	0.65	0.78	74.19	0.06	0.90	54.57	0.84	0.81	82.83	0.33	0.60	57.36
Hypo-active users of suspicious apps	0.99	0.82	81.45	0.91	0.81	80.22	0.91	0.94	87.28	0.54	0.87	56.23
Active users of suspicious apps	0.99	1.00	99.59	0.99	1.00	99.60	0.99	0.99	99.59	0.66	1.00	66.44
Hyperactive users of suspicious apps	0.99	0.99	99.91	0.99	1.00	99.70	0.99	1.00	99.71	0.93	1.00	92.60

Note: P: Precision, R: Recall, A: Accuracy

$$x_{r\_out1} = discriminator\_1(x_{real\_data}) \tag{4}$$

$$x_{r\_out2} = discriminator\_2(x_{real\_data}) \tag{5}$$

$$x_{classified\_1} = \frac{1}{1 + e^{-x_{g\_out}}} \tag{6}$$

$$x_{classified\_2} = \frac{1}{1 + e^{-x_{r\_out1}}} \tag{7}$$

$$x_{classified\_3} = \frac{1}{1 + e^{-x_{r\_out2}}} \tag{8}$$

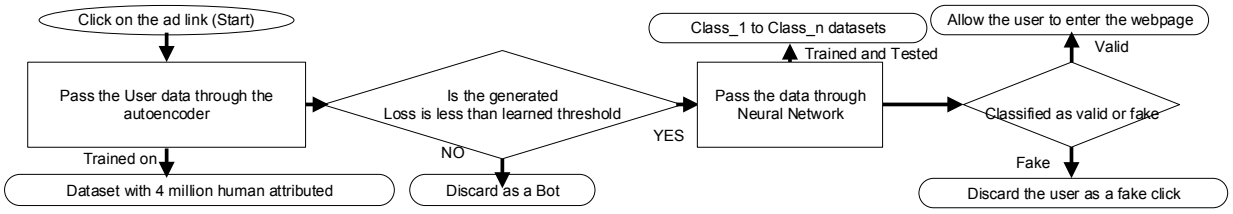


Figure 2: Multi-layered Neural Network with an Attached Autoencoder Model

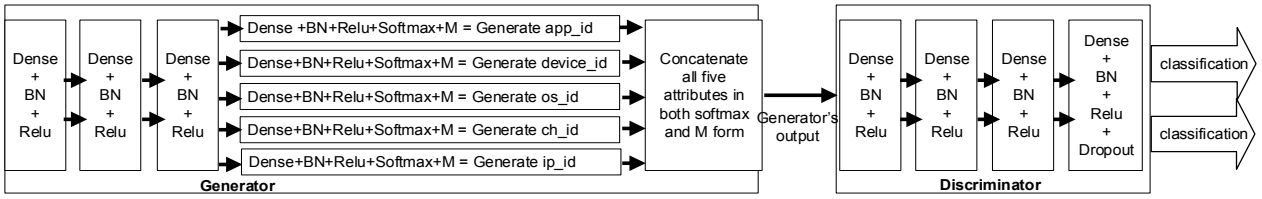


Figure 3: GAN Structure

Note: $(BN : BatchNormalization, M : max\_one\_hot\_encode, Dense : Singlelayerneuralnet, Relu : max(0.2 * x, x), x : Outputof BatchNormalization)$

Table 3: Results from Neural Network

Dataset	P	R	A(%)	AUC
class 0	0.95	0.94	94.00	0.979
class 1	0.71	0.73	73.53	0.899
class 2	0.74	0.75	74.86	0.877
class 3	0.81	0.81	81.40	0.897
class 4	0.89	0.88	90.01	0.802
class 5	0.98	0.97	99.50	0.947
class 6	0.99	0.99	99.80	0.713

Note: Class 0: Equal distribution of valid and fake, P: Precision, R: Recall,A: Accuracy

$x_{classified\_1}$: Probability that the generated data is real. $x_{classified\_2}$: Probability that the real data is real. $x_{classified\_3}$: Probability that the click in real data is valid.

5.2.3 Loss. Discriminator and Generator loss are as follows.
Discriminator Loss let *supervised* be *sp* and *unsupervised* be *usp*

$$D_{usp\_gloss} = -\log(1 - x_{classified\_1}) \tag{9}$$

$$D_{usp\_dloss} = -\log(x_{classified\_2}) \tag{10}$$

$$D_{sp\_loss} = -y * logx_{classified\_3} + (-(1-y)) * log(1 - x_{classified\_3}) \tag{11}$$

$$D_{loss} = D_{usp\_gloss} + D_{usp\_dloss} + D_{sp\_loss} \tag{12}$$

Generator Loss

$$G_{usp\_loss} = -\log(x_{classified\_1}) \tag{13}$$

$$G_{sp\_loss} = -(log(Cos(\theta))) = \frac{-log(x_{real\_data} * x_{generated\_logits})}{(|x_{real\_data}| * |x_{generated\_logits}|)} \tag{14}$$

$$G_{loss} = G_{usp\_loss} + G_{sp\_loss} \tag{15}$$

Both these losses are used to train the Generator and discriminator separately.

5.2.4 Results. We trained our GAN on a dataset with equally valid and fake clicks; the discriminator got an accuracy of 89.7%. Based on discriminator's classification, the generator created 63000 valid clicks, given 100000 randomized inputs. To make sure that we were getting the correct results, we did a dot product of randomly selected real valid clicks and the ones generated by the GAN, we got an average result of 0.65, while the average results for the dot products of the same real valid clicks with each other was 0.85, showing that there is a great variance even within the real valid clicks. On the other hand, the dot product of the generated valid clicks and real fake clicks was 0.24. We even added this data to the Equally distributed training dataset to train the Neural Network, and saw an increase of 0.6-1%, from 94% to 94.6-95% on the same test cases as used in Table 3, depending on the structure of the Neural Network. The increase may be a minute for the given dataset since the neural network was already performing well, but this technique can be used in other one-sided sparse data sets too. To the best of our knowledge, this is the first work to use $Cos(\theta)$, as a supervised loss function in a semi-supervised GAN.

6 CONCLUSION

As shown by the above results, we have achieved good high accuracy even on small datasets, showing the capability of the neural network to understand the probability distribution of fake and real

users. The reason for the success of the neural network in this form of fraud detection is because of multiple layers, with each layer learning the distribution to a certain extent and passing that knowledge to the next layers. The Auto-Encoder can understand the distribution of human clicks since the data available belongs to human beings. This allows it to encode and then decode the data; it is a type of decryption and encryption. Since it has only been trained to encrypt and decrypt human data, it can be said that the loss error will be high for bots. However due to lack of bot clicks available, the auto-encoder in current form will not be able to detect bot with a distribution similar to humans, but when put in practice it will be able to learn botnet distribution too. It is different from a standard neural network since the loss generated is ambiguous, which means, we will have to change the loss threshold from time to time depending on the success and failure of auto-encoder. Neural Network will fail to detect bots if they maliciously download, but the autoencoder will discard them.

REFERENCES

[1] D. Antoniou, M. Paschou, E. Sakkopoulos, E. Sourla, G. Tzimas, A. Tsakalidis, and E. Viennas. 2011. Exposing Click-fraud using a Burst Detection Algorithm. In *2011 IEEE Symposium on Computers and Communications (ISCC)*. Kerkyra, Greece, pp. 1111–1116. https://doi.org/10.1109/ISCC.2011.5983854

[2] M. Barreno, B. Nelson, A. Joseph, and J. Tygar. 2010. The Security of Machine Learning. *Machine Learning* 81, 2 (01 Nov 2010), pp. 121–148. https://doi.org/10.1007/s10994-010-5188-5

[3] B. Biggio, I. Corona, B. Nelson, B. B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. 2014. *Security Evaluation of Support Vector Machines in Adversarial Environments.* Springer International Publishing, Cham, pp. 105–153. https://doi.org/10.1007/978-3-319-02300-7_4

[4] B. Biggio, G. Fumera, and F. Roli. 2014. Security Evaluation of Pattern Classifiers under Attack. *IEEE Transactions on Knowledge and Data Engineering* 26, 4 (April 2014), pp. 984–996. https://doi.org/10.1109/TKDE.2013.57

[5] B. Biggio, B. Nelson, and P. Laskov. 2011. Support Vector Machines Under Adversarial Label Noise. In *Proceedings of the Asian Conference on Machine Learning (Proceedings of Machine Learning Research)*, Chun-Nan Hsu and Wee Sun Lee (Eds.), Vol. 20. PMLR, South Garden Hotels and Resorts, Taoyuan, Taiwain, pp. 97–112. http://proceedings.mlr.press/v20/biggio11.html

[6] F. Chollet. 2015. Keras: Deep Learning Library for Theano and Tensorflow. https://keras.io/

[7] R. A. Costa, R. J. G. B. de Queiroz, and E. R. Cavalcanti. 2012. A Proposal to Prevent Click-Fraud Using Clickable CAPTCHAs. In *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*. MD, USA, pp. 62–67. https://doi.org/10.1109/SERE-C.2012.13

[8] J. Crussell, R. Stevens, and H. Chen. 2014. MAdFraud: Investigating Ad Fraud in Android Applications. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '14)*. ACM, NY, USA, pp. 123–134. https://doi.org/10.1145/2594368.2594391

[9] V. Dave, S. Guha, and Y. Zhang. 2012. Measuring and Fingerprinting Click-spam in Ad Networks. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12)*. ACM, NY, USA, pp. 175–186. https://doi.org/10.1145/2342356.2342394

[10] L. Dritsoula and J. Musacchio. 2014. A Game of Clicks: Economic Incentives to Fight Click Fraud in Ad Networks. *SIGMETRICS Perform. Eval. Rev.* 41, 4 (April 2014), pp. 12–15. https://doi.org/10.1145/2627534.2627538

[11] M. Faou, A. Lemay, D. Décary-Hétu, J. Calvet, F. Labréche, M. Jean, B. Dupont, and J. M. Fernande. 2016. Follow the Traffic: Stopping Click Fraud by Disrupting the Value Chain. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. Auckland, New Zealand, pp. 464–476. https://doi.org/10.1109/PST.2016.7907001

[12] X. Glorot and Y. Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. pp. 249–256.

[13] I. Goodfellow, J. Pouget-Abadie, M Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680.

[14] H. Haddadi. 2010. Fighting Online Click-fraud Using Bluff Ads. *SIGCOMM Comput. Commun. Rev.* 40, 2 (April 2010), pp. 21–25. https://doi.org/10.1145/1764873.1764877

[15] Ch. Md. Rakin Haider, A. Iqbal, A. Hasan Rahman, and M. Sohel Rahman. 2018. An Ensemble Learning Based Approach for Impression Fraud Detection in Mobile Advertising. *Journal of Network and Computer Applications* 112 (2018), pp. 126 – 141. https://doi.org/10.1016/j.jnca.2018.02.021

[16] Kaggle.com. 2018. TalkingData AdTracking Fraud Detection Challenge. https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection

[17] D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv e-prints* (Dec. 2014). arXiv:1412.6980

[18] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. 2014. Semi-supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems* (2014), pp. 3581–9.

[19] B. J. Kitts, T. Najm, and B. Burdick. 2008. Identifying Automated Click Fraud Programs. Abandoned May. 7th., 2007, Application on Nov. 13th. 2008.

[20] R. Kohavi and G. H. John. 1997. Wrappers for Feature Subset Selection. *Artificial Intelligence* 97, 1 (1997), pp. 273 – 324. https://doi.org/10.1016/S0004-3702(97)00043-X Relevance.

[21] J. Kwon, J. Kim, J. Lee, H. Lee, and A. Perrig. 2014. PsyBoG: Power Spectral Density Analysis for Detecting Botnet Groups. In *2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*. PR, USA, pp. 85–92. https://doi.org/10.1109/MALWARE.2014.6999414

[22] W. Li, H. Li, H. Chen, and Y. Xia. 2015. AdAttester: Secure Online Mobile Advertisement Attestation Using TrustZone. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15)*. ACM, NY, USA, pp. 75–88. https://doi.org/10.1145/2742647.2742676

[23] B. Liu, S. Nath, R. Govindan, and J. Liu. 2014. DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, Berkeley, CA, USA, pp. 57–70.

[24] A. Metwally, D. Agrawal, and A. EI Abbadi. 2007. Detectives: Detecting Coalition Hit Inflation Attacks in Advertising Networks Streams. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, NY, USA, pp. 241–250. https://doi.org/10.1145/1242572.1242606

[25] B. Nelson, B. Rubinstein, L. Huang, A. Joseph, S. Lee, S. Rao, and J. Tygar. 2010. Query Strategies for Evading Convex-Inducing Classifiers. *CoRR* abs/1007.0484 (2010). arXiv:1007.0484 http://arxiv.org/abs/1007.0484

[26] A. Nisser and N. B. Weidman. 2016. Measuring Ethnic Preferences in Bosnia and Herzegovina with Mobile Advertising. *PLOS ONE* 11, 12 (12 2016), pp. 1–11. https://doi.org/10.1371/journal.pone.0167779

[27] X. Qiu, Y. Zuo, and G. Liu. 2018. ETCF: An Ensemble Model for CTR Prediction. In *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*. Hangzhou, China, pp. 1–5. https://doi.org/10.1109/ICSSSM.2018.8465044

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning Representations by Back-propagating Errors. *Nature International Journal of Science* (1986), pp. 323–536.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–58.

[30] G. S. Thejas, T. C. Pramod, S. S. Iyengar, and N. R. Sunitha. 2018. Intelligent Access Control: A Self-Adaptable Trust-Based Access Control (SATBAC) Framework Using Game Theory Strategy. In *Proceedings of International Symposium on Sensor Networks, Systems and Security*, Nageswara S.V. Rao, Richard R. Brooks, and Chase Q. Wu (Eds.). Springer International Publishing, Cham, pp. 97–111. https://doi.org/10.1007/978-3-319-75683-7_7

[31] G. S. Thejas, J. Soni, K. Chandna, S. S. Iyengar, N. R. Sunitha, and N. Prabakar. 2019. Learning-Based Model to Fight against Fake Like Clicks on Instagram Posts. In *SoutheastCon 2019*. Huntsville, Alabama, USA, pp. 1–8. In press.

[32] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. 2016. AI 2: Training a Big Data Machine to Defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud, HPSC, and IDS*. NY, USA, pp. 49–54. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.79

[33] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th Int. Conf. on Machine Learning (ICML '08)*. ACM, NY, USA, pp. 1096–1103. https://doi.org/10.1145/1390156.1390294

[34] J. Williams. 2014. Click Fraud - The 5 Most Common Forms of Click Fraud. https://fruition.net/about/blog/types-click-fraud-detect/