Web Bot Detection Evasion Using Generative Adversarial Networks

Christos Iliou
Information Technologies Institute
CERTH
Thessaloniki, Greece
BU-CERT
Bournemouth University
Bournemouth, United Kingdom
iliouchristos@iti.gr

Theodoros Kostoulas
Department of Information
& Communication Systems Engineering
University of the Aegean
Samos, Greece
theodoros.kostoulas@aegean.gr

Theodora Tsikrika
Information Technologies Institute
CERTH
Thessaloniki, Greece
theodora.tsikrika@iti.gr

Vasilis Katos
BU-CERT
Bournemouth University
Bournemouth, United Kingdom
vkatos@bournemouth.ac.uk

Stefanos Vrochidis
Information Technologies Institute
CERTH
Thessaloniki, Greece
stefanos@iti.gr

Ioannis Kompatsiaris
Information Technologies Institute
CERTH
Thessaloniki, Greece
ikom@iti.gr

***Abstract*—Web bots are programs that can be used to browse the web and perform automated actions. These actions can be benign, such as web indexing and website monitoring, or malicious, such as content scraping and scalping. To detect bots, web servers consider bots' fingerprint and behaviour, with research showing that techniques that examine the visitor's mouse movements can be very effective. In this work, we showcase that web bots can leverage the latest advances in machine learning to evade detection based on their mouse movements and touchscreen trajectories (for the case of mobile web bots). More specifically, the proposed web bots utilise Generative Adversarial Networks (GANs) to generate images of trajectories similar to those of humans, which can then be used by bots to evade detection. We show that, even if the web server is aware of the attack method, web bots can generate behaviours that can evade detection.**

***Index Terms*—advanced web bots, generative adversarial networks, evasive web bots, mouse movements, humanlike behaviour**

I. Introduction

Web bots allow for the automation of several tasks which would be impossible to perform otherwise. Such tasks include indexing the web, monitoring websites and validating hyperlinks and HTML code, testing the functionality of web applications, and more. Imperva's 2020 report [1] showed that web bots accounted for 37.2% of the total traffic that they monitored.

The aforementioned tasks require web bots to perform some actions automatically, such as mousing over actions, clicking on items, filling forms, etc. This resulted in the introduction of browsing automation software, such as Selenium[1], that supports the main functionalities of a browser, including mouse movements, clicks, and keystroke actions.

Web bots that can automatically perform such advanced functions have also been used for malicious purposes [1]. Thus, web servers employ web bot detection techniques, with the most common ones being based on CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) challenges [2]. While such challenges have been very effective in the past, the usability issues that they introduce along with their reduced effectiveness due to the advances in image processing [3] and speech recognition [4] led current research to focus on web bot detection techniques that are based on the visitors' fingerprint (i.e., browser fingerprinting) and behaviour. For example, Google's latest version of its CAPTCHA challenge[2] considers the visitor's fingerprint and actions to decide whether a visitor is a bot or not [5].

Browser fingerprinting is the process of enumerating several characteristics of the visitor (e.g., font detection, plugin enumeration, WebGL fingerprinting, unique JavaScript variables, etc. [6]–[8]) to identify whether a visitor uses browsing automation software. This approach can effectively detect simple crawling scripts, as well as common browsing automation tools (e.g., Selenium) in their default configurations, since their characteristics are different from those of common browsers.

Even though detection based on fingerprints can adequately detect several types of web bots, research has shown that it has several flaws and can be bypassed by using specially configured browsing automation software [6], [9], or by using common browsers instead of browsing automation software [5]. Thus, to effectively detect web bots, current

[1]https://www.seleniumhq.org/

978-1-7281-5684-2/20/$31.00 ©2021 IEEE

[2]https://www.google.com/recaptcha

research examines, besides their fingerprint, their behaviour regarding the web pages that they visit [10]–[13] and the mouse movements that they perform [13]–[15].

In this work, we examine how well current state-of-the-art web bot detection approaches work against web bots that generate humanlike mouse and touchscreen trajectories (for the case of mobile web bots) in a novel way by using Generative Adversarial Networks (GANs). As opposed to previous research, this work takes advantage of the fact that mouse and touchscreen trajectories can be depicted as images and thus can be used as input to GANs to create new humanlike trajectories. To the best of our knowledge, this is the first work that proposes the generation of images with humanlike mouse or touchscreen trajectories using GANs. This technique can be combined with appropriately configured browsing automation software to increase the evasiveness of the web bots. The main contributions of this work are:

- The proposal of a novel technique that can be used by web bots for the generation of humanlike mouse and touchscreen trajectories by utilising GANs.
- The evaluation of the proposed technique against a web server that performs web bot detection and is aware of this type of attack. We demonstrate that even if the same technique configurations are used by the web server, the bots can still evade detection.

Through this research, we aim to increase the awareness on the possibility of evasive web bots using the latest advances in machine and deep learning and highlight the importance of defense mechanisms that take such adversaries into account.

The remainder is structured as follows. Section II covers related work. Section III describes the employed web bot detection server, while Section IV presents the proposed evasive web bots. Section V describes the evaluation methodology and experimental setup, while Section VI discusses the results. Finally, Section VII concludes and outlines future work.

II. Related work

The web bot detection problem aims to distinguish web bots from human visitors [14], [16]–[18], and, in some cases, to further categorise bots based on their functionality [19], purpose [20], [21], or complexity [11]. In this section, we first present techniques for the detection of web bots and then techniques used by web bots to avoid detection.

A. Web bot detection

To detect web bots based on their behaviour, proposed methods typically examine: (i) web logs that visitors generate [10]–[13], and (ii) visitor's mouse movements [13]–[15].

Regarding the web logs, several measurable features are extracted from the pages that the visitors accessed, along with the access frequency and access patterns [11], [22]; the semantics of the content of each page have also been considered [12]. Features extracted from web logs are used to train machine learning models to classify the new visitors as bots or humans, mainly through the use of classification [11], [16] or clustering algorithms [22], [23]. Additionally, the detection process can be performed either offline (after the end of a session) [11], [22] or online by performing an estimation during the session [10], [14].

More recent research has proposed detection methods that use mouse movements. Such approaches either transform the mouse trajectories into images and use them as input to Convolutional Neural Networks (CNNs) [13], [15], or initially extract several high level actions from them (e.g., click, point-and-click, and drag-and-drop) and then extract additional features from each action (e.g., duration, distance, displacement, etc.) that are used as input to machine learning algorithms [14].

Current web bot detection techniques that are based on mouse and touchscreen trajectories have shown very promising results when faced with web bots that try to present a humanlike behaviour through heuristic approaches [13]–[15]. Such works tested bots that perform mouse movements following a specific type of lines (such as straight lines, curves, etc.) [13], [15], or a combination of such lines with keystrokes [14], or web bots that try to generate touchscreen trajectories with similar features as the ones extracted from humans' (such as duration of mouse movements, velocity, etc.) using statistics [24].

B. Web bot detection evasion

Several evasion approaches can be used by web bots to avoid detection. Web bots can use statistics to generate a humanlike behaviour in regards to the web pages they visit [25]. Similar techniques are also used by mobile web bots to generate swipe and accelerometer data [24]. Moreover, Reinforcement Learning (RL) techniques are used by web bots to learn an evasive behaviour [5]. Finally, GANs taking advantage of Long Short Term Memory network layers are used to generate synthetic swipe and accelerometer data [24].

In this work, we leverage the high performance of GANs in their ability to generate images with the same features as the ones used for training [26] for the generation of mouse movements and touchscreen trajectories that can be used by web bots to evade detection. As opposed to techniques that are based either on statistics of specific features extracted from the trajectories [24] or on RL techniques that learn through trying several times [5], we consider that the trajectories can be directly generated as images that can then be used as guidance for the web bots to perform humanlike mouse movements and mobile touch events to exhibit a more evasive behaviour.

III. Web bot detection method

Before presenting the proposed method for evading web bot detection, a suitable detection method is described. A highly accurate method for detecting web bots based on their mouse movements is to generate images depicting the mouse movements that visitors perform on each web page and feed those into Convolutional Neural Networks (CNNs) [13], [15]. A similar approach can be used for mobile touchscreen trajectories, as they can also be processed as images. The general architecture of this web bot detection framework is presented in Fig. 1.

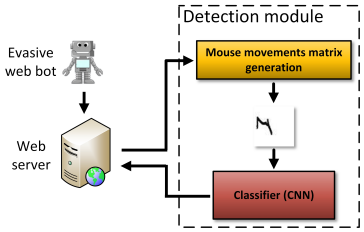


Fig. 1. Web bot detection framework

The first step is to generate images from the sequence of mouse movements each visitor performs on each web page. These sequences include the coordinates of all pixels that the mouse passed at a specific time. The data are collected as $\{(x_1, y_1, t_1), ..., (x_n, y_n, t_n)\}$, where x_i and y_i are the coordinates of the current mouse point, t_i is the respective timestamp, and n is the total number of points over which the mouse hovered. These sequences are mapped into 2-dimensional matrices, where each (x_i, y_i) value pair corresponds to the index of an element in the matrix and $dt_i = t_{i+1} - t_i$ to its value [13], [15]. The same approach can be used for touchscreen actions on mobile devices. To collect such data, a JavaScript file is embedded in each web page to periodically (e.g., every few seconds) send the data to the server.

The extracted images of the visitors' mouse movements are used as input into a (trained) CNN and the classification result indicates whether the visitor is detected as a bot or a human. In this work, a simple CNN architecture that combines a series of Convolution and Max-pooling layers was used; details of the architecture are presented in the Appendix. Additionally, images were normalised so that their values become between '0' and '1', a commonly used technique in CNNs. The CNN was implemented using Tensorflow[3] and the Keras API[4].

IV. Evasive web bots using GANs

Evasive web bots try to hide their bot nature so as to not be detected and blocked by the web servers of interest. Given the current web bot detection approaches, evasive web bots try to (i) present a fingerprint that is very similar to a browser one, and (ii) exhibit a humanlike behaviour (regarding both the web pages they visit and the mouse or touchscreen trajectories that they perform). This work examines the case of detecting web bots based on the mouse or touchscreen trajectories, a technique shown to be very effective in the case of mouse movements [13]–[15], [24]. If the bots can generate humanlike trajectories, then these trajectories can be combined with additional techniques, such as using a browser-like fingerprint and selecting web pages to visit in a humanlike manner [13], [25], to enable a more evasive behaviour.

The general architecture of the proposed web bots is presented in Fig. 2. More specifically, the proposed web bots utilise GANs to generate humanlike mouse movements based on real ones, since GANs have proven to generate realistic

[3]https://www.tensorflow.org/

[4]https://keras.io/

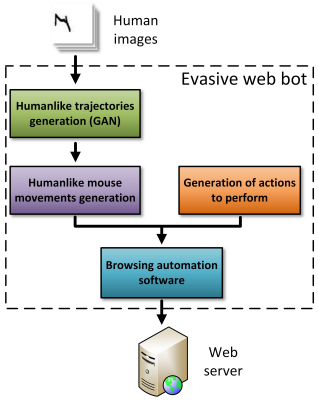


Fig. 2. Evasive web bots

images based on the images they were trained on [26]. GANs allow the generation of several mouse trajectories with different starting and ending points each. From the available trajectories, web bots can select the most appropriate ones based on the actions that they want to perform.

GANs are architectures where two neural networks, the Generator and the Discriminator, contest with each other and are trained simultaneously in an "adversarial" setting. The Generator constantly tries to generate images that can fool the Discriminator into thinking that those images are real. The Generator is trained to take as input points from a latent space (in our case, these points correspond to vectors with values drawn from a Gaussian distribution) and map them into new images. Diversely, the Discriminator is trained to distinguish fake images from real ones, using both images from human trajectories and images created by the Generator. During that process, the Generator becomes better at generating images that look similar to the input images, while the Discriminator becomes better at identifying which images are real.

In this work, a Deep Convolutional Generative Adversarial Network (DCGAN) architecture is used for the generation of humanlike mouse and touchscreen trajectories following the recommendations in [26]. Differentiating from [26], we used LeakyReLU for both the Generator and the Discriminator, instead of only the Discriminator, because in our preliminary experiments it gererated slightly better images. Details of the architecture are presented in the Appendix. For the GAN implementation, Tensorflow and the Keras API were used.

V. Evaluation

To assess the effectiveness of the proposed approach, two types of web bots are considered: (i) the scraping web bots that crawl web servers to harvest their content, and (ii) the mobile web bots that are requested to perform a specific task to prove that they are humans, which in our case is to generate touchscreen trajectories of specific numbers [24], [27].

A. Evaluation methodology

To examine the evasiveness of web bots that utilise GANs for the generation of humanlike mouse movements and touchscreen trajectories, we assume that web bots have already

in their possession several human mouse movements and touchscreen events. Bots can then use these data as input to GANs to generate new humanlike behaviours with similar characteristics as the ones they have previously seen.

Additionally, for simplicity, we consider that the web server also uses GANs to train its detection models and that both GANs (the GAN used by the web bots and the GAN used by the web server) have the same architecture and configurations (see Appendix) and were trained for 20k epochs. We argue that these similarities in the aforementioned methods make the evasion process more difficult.

On the other hand, we consider that the human behaviours used by the web bots to train their models should be different from the ones used by the web server. This choice was made to present a more realistic scenario, where the web server should be faced with new, unseen behaviours. Thus, in our experiments, different human images are used by the web bot detection module and by the evasive web bots.

To evaluate how well the web bots can evade detection, we initially train and evaluate the performance of the web bot detection framework. For that, the considered data sets were split into 80% for the training and 20% for the testing, and the CNN was trained for 30 epochs. Then, this detection framework was used for evaluating the evasive web bots.

Finally, to account for the fact that the performance of both the detection module and the web bots might be affected by the images selected as the training set in each case, we repeated the experiments considering different combinations of the sets to be used by the web server and the web bots.

B. Datasets

The evasive web bots were evaluated on two datasets: (i) a *Web dataset* generated by humans while browsing a web server which hosted content copied from Wikipedia[5] [13], and (ii) part of the *HuMIdb dataset*[6], in which humans were requested to "draw" digits on mobile devices [24], [27].

Web: For this dataset, 27 human subjects were requested to browse a web server that hosted 110 Wikipedia pages from 11 categories/topics [13]. Each human subject was instructed to create two sessions, each session being between 15-20 minutes. All sessions were anonymised and only information that indicates which sessions belong to the same user has been kept. The dataset is available upon request.

HuMIdb: The second dataset is part of the HuMIdb dataset [24], [27], a dataset that contains a wide range of mobile sensors values acquired during a natural human-mobile interaction performed by more than 600 users; this database is not public but available upon request. To generate this dataset, users were requested to perform eight simple tasks between one and five times (indicating different sessions). In this work, we utilised data from the task where a user had to draw with their finger the digits '0' to '9' over the touchscreen.

In both datasets, and to account for the fact that different visitors have varying monitor resolutions, we re-scaled all images to the same dimensions with a lower resolution. For that, we initially increased the size of each mouse move by 30, i.e., for each pixel where a mouse move was performed (and thus has a non-zero value), its neighbours within distance less or equal to 30 pixels were given the value of that pixel. Then we re-scaled the images to 56x56 dimensions using the Pillow[7] library and with the "antialias" (high-quality downsampling filter) configuration. The 56x56 dimensions were selected to account also for the fact that high resolution images consume a lot of memory when used for training the networks.

Moreover, the dataset was split into three sets to facilitate the evaluation process: two used by the web detection server for training, and one used by the evasive web bots to generate a humanlike behaviour. To account for the fact that different images of the same user should not be in different sets, the split was performed on a per user basis.

Finally, to account for the randomness introduced when the models are trained on GPU experiments were run five times and the average of all runs was considered.

TABLE I
USERS, SESSIONS, AND IMAGES FOR EACH DATASET

		1	**2**	**3**
Web	*Users*	9	9	9
	Sessions	18	18	18
	Images	367	369	561
HuMIdb	*Users*	200	200	200
	Sessions	839	847	798
	Images	8390	8470	7980

C. Evaluation metrics

To assess the effectiveness of web detection, we used the *balanced accuracy* evaluation metric, which is used in the web bot detection problem when datasets are unbalanced [11]. For evaluating the evasiveness of the web bots and since we have only one class, we used *recall*, i.e., the percentage of the web bots that were correctly identified (True Positive) divided by the total number of web bots used (True Positive + False Negatives).

VI. RESULTS

A. Web bot detection performance

The performance of the detection framework is presented in Table II. Experiments are performed on *set X & set Y* (denoted as *X & Y*), where a random 80% of *X* and a random 80% of *Y* are used for training the CNN detection framework, and the remaining 20% of *X* and *Y* are used for testing. Data from *set X* are considered as the 'human' class, while *set Y* is used as input to the GAN of the bot detection; this GAN generates the same number of images as the number of images in its input.

The detection framework manages to achieve a very high accuracy and recall in both datasets. The high performance was expected, since frameworks that use mouse movements for web bot detection have shown very good results [13], [15].

[5]https://www.wikipedia.org/

[6]https://github.com/BiDAlab/HuMIdb

[7]https://pillow.readthedocs.io/en/stable/

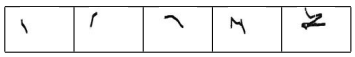
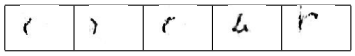
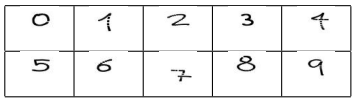
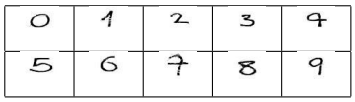
TABLE II
PERFORMANCE OF THE DETECTION FRAMEWORK

	Balanced accuracy						
X&Y	**1&2**	**2&1**	**2&3**	**3&2**	**1&3**	**3&1**	**Avg**
Web	0.986	1.000	0.970	0.988	0.974	0.994	0.985
HuMIdb	0.995	0.996	0.995	0.997	0.995	0.997	0.996
	Recall						
Web	1.000	1.000	0.996	1.000	1.000	1.000	0.999
HuMIdb	0.992	0.996	0.996	0.997	0.995	0.997	0.995

TABLE III
PERFORMANCE OF EVASIVE WEB BOTS

	Recall						
X&Y	**1&2**	**2&1**	**2&3**	**3&2**	**1&3**	**3&1**	**Avg**
Z	**3**		**1**		**2**		
Web	0.531	0.648	0.693	0.437	0.263	0.139	0.452
HuMIdb	0.950	0.943	0.925	0.928	0.950	0.928	0.937

TABLE IV
SELECTED IMAGES BY HUMANS AND GENERATED BY GANS

Web	
Images by humans	
Images generated by GANs	
HuMIdb	
Images by humans	
Images generated by GANs	

Since the same GAN was used to generate (different) images for training and for testing, the CNN was able to identify this behaviour.

B. Web bot detection against evasive web bots

To evaluate the proposed web bots, the already trained detection framework that takes advantage of CNNs was used. The evasive web bots train their GAN by using different human images from the ones used by the detection framework. The results are presented in Table III where the *set X & set Y* indicates the sets that were used for training the web bot detection framework (see Table II) and the *set Z* indicates the set that was used for training the evasive web bots. In this case, the recall indicates the percentage of images that were correctly identified as images generated by web bots divided by the total generated images.

The drop in performance of the detection module when tested on the evasive web bots shows that generating human-like trajectories using GANs is very effective against web bot detection techniques, even if the same architecture and configurations are used by the detection module. This was expected, since the web server used the same GAN to generate images for its training and validation, but was evaluated with images generated from a different GAN used by the web bots. Additionally, the drop in performance is very high in the case of the *Web dataset* as opposed to the *HuMIdb*. This is also something to be expected, since mouse trajectories on a web server vary more between different humans compared to "drawing" numbers on smartphones, making the "modeling" of this behaviour more difficult.

Finally, to qualitatively evaluate the effectiveness of the use of GANs for the generation of humanlike trajectories, selected images are presented in Table IV. We observe that web bots have a tendency to select simpler mouse movements to follow instead of more complex ones. This could be attributed to the fact that it is difficult for the latter to be modelled.

VII. CONCLUSIONS AND FUTURE WORK

This work proposed a novel way for bots to generate humanlike mouse and touchscreen trajectories using GANs that can be used by web bots to exhibit a more evasive behaviour. We evaluated the proposed approach against web bot detection techniques that consider the mouse trajectories of visitors, an approach that has shown to be very effective. Additionally, we assume that the web bot detection framework is aware of and uses the same methods as web bots, as well as their configurations to train its detection models. We show that, even if the web server knows of this attack, it is challenging to effectively detect those web bots.

When applying this technique to a real world scenario, where, in case of crawling bots, a web bot aims to move from one location to another in a humanlike manner, bots should generate several potential images from the GAN until they find one that fits their purpose. However, as we show, web bots generated simple evasive mouse movements, which might be a problem when trying to perform more complex actions. Thus, future work will further investigate the use of GANs to generate more advanced humanlike mouse movements. Additionally, the proposed approach will be compared with different approaches proposed in literature, such as the ones that use Reinforcement Learning to evade detection [5].

ACKNOWLEDGMENT

We would like to thank the creators of *HuMIdb*, and the human subjects that participated in our research. This work was supported by the FORESIGHT (H2020 833673), ECHO (H2020 830943), and IDEAL-CITIES (H2020 778229) projects, funded by the European Commission.

REFERENCES

[1] Imperva, "Bad bot report," 2020. [Online]. Available: https://www.imperva.com/resources/resource-library/reports/2020-bad-bot-report/

[2] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2003, pp. 294–311.

[3] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot: (deep) learning to break semantic image captchas," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 388–403.

[4] K. Bock, D. Patel, G. Hughey, and D. Levin, "uncaptcha: a low-resource defeat of recaptcha's audio challenge," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[5] I. Akrout, A. Feriani, and M. Akrout, "Hacking google recaptcha v3 using reinforcement learning," *arXiv preprint arXiv:1903.01003*, 2019.

[6] B. Amin Azad, O. Starov, P. Laperdrix, and N. Nikiforakis, "Web runner 2049: Evaluating third-party anti-bot services," 2020.

[7] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: a survey," *ACM Transactions on the Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.

[8] M. Schwarz, F. Lackner, and D. Gruss, "Javascript template attacks: Automatically inferring host information for targeted exploits." in *NDSS*, 2019.

[9] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, "Fp-crawlers: Studying the resilience of browser fingerprinting to block crawlers," in *NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'20)*, 2020.

[10] S. Rovetta, A. Cabri, F. Masulli, and G. Suchacka, "Bot or not? a case study on bot recognition from web session logs," in *Italian Workshop on Neural Nets*. Springer, 2017, pp. 197–206.

[11] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and Y. Kompatsiaris, "Towards a framework for detecting advanced web bots," in *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019.*, 2019, pp. 18:1–18:10.

[12] A. Lagopoulos and G. Tsoumakas, "Content-aware web robot detection," *Applied Intelligence*, vol. 50, no. 11, pp. 4017–4028, 2020.

[13] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and Y. Kompatsiaris, "Detection of advanced web bots by combining web logs with mouse behavioural biometrics," *Digital Threats: Research and Practice*, in press.

[14] Z. Chu, S. Gianvecchio, and H. Wang, "Bot or human? A behavior-based online bot detection system," in *From Database to Cyber Security - Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*, 2018, pp. 432–449.

[15] A. Wei, Y. Zhao, and Z. Cai, "A deep learning approach to web bot detection using mouse behavioral biometrics," in *Biometric Recognition - 14th Chinese Conference, CCBR 2019, Zhuzhou, China, October 12-13, 2019, Proceedings*, 2019, pp. 388–395.

[16] D. S. Sisodia, S. Verma, and O. P. Vyas, "Agglomerative approach for identification and elimination of web robots from web server logs to extract knowledge about actual visitors," *Journal of Data Analysis and Information Processing*, vol. 3, no. 01, p. 1, 2015.

[17] A. Cabri, G. Suchacka, S. Rovetta, and F. Masulli, "Online web bot detection using a sequential classification approach," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2018, pp. 1536–1540.

[18] D. Doran and S. S. Gokhale, "An integrated method for real time and offline web robot detection," *Expert Systems*, vol. 33, no. 6, pp. 592–606, 2016.

[19] ——, "A classification framework for web robots," *Journal of the Association for Information Science and Technology*, vol. 63, no. 12, pp. 2549–2554, 2012.

[20] Q. Bai, G. Xiong, Y. Zhao, and L. He, "Analysis and detection of bogus behavior in web crawler measurement," in *Proceedings of the Second International Conference on Information Technology and Quantitative Management, ITQM 2014, National Research University Higher School of Economics (HSE), Moscow, Russia, June 3-5, 2014*, ser. Procedia Computer Science, F. Aleskerov, Y. Shi, and A. Lepskiy, Eds., vol. 31. Elsevier, 2014, pp. 1084–1091.

[21] M. Zabihimayvan, R. Sadeghi, H. N. Rude, and D. Doran, "A soft computing approach for benign and malicious web robot detection," *Expert Syst. Appl.*, vol. 87, pp. 129–140, 2017.

[22] D. Stevanovic, N. Vlajic, and A. An, "Detection of malicious and non-malicious website visitors using unsupervised neural network learning," *Applied Soft Computing*, vol. 13, no. 1, pp. 698–708, 2013.

[23] Z. Dewa and L. A. Maglaras, "Data mining and intrusion detection systems," *vol*, vol. 7, pp. 62–71, 2016.

[24] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and O. Delgado-Mohatar, "Becaptcha: Bot detection in smartphone interaction using touchscreen biometrics and mobile sensors," *arXiv preprint arXiv:2005.13655*, 2020.

[25] C. Iliou, T. Tsikrika, S. Vrochidis, and Y. Kompatsiaris, "Evasive focused crawling by exploiting human browsing behaviour: a study on terrorism-related content," in *Proceedings of the 1st International Workshop on Cyber Deviance Detection co-located with the Tenth International Conference on Web Search and Data Mining CyberDD @ WSDM 2017), Cambridge, UK, February, 10, 2017.*, 2017.

[26] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[27] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and I. Bartolome, "Becaptcha: Detecting human behavior in smartphone interaction using multiple inbuilt sensors," in *AAAI Workshop on Artificial for Cyber Security (AICS)*, 2020.

APPENDIX

This section details the used neural network architectures.

CNN architecture for Web bot detection			
Layer type	**Kernel size / stride**	**Output Shape**	**Activation**
InputLayer	–	(56, 56, 1)	–
Conv	3x3 / 1	(54, 54, 64)	ReLU
M-Pool	2x2 / 2	(27, 27, 64)	–
Conv	3x3 / 1	(25, 25, 64)	ReLU
M-Pool	2x2 / 2	(12, 12, 64)	–
Flatten	–	(9216)	–
Dense	–	(2)	Softmax

Layer type	**Configs**	**Output Shape**
Generator architecture		
InputLayer	–	(100)
Dense	–	(12544)
BatchNormalisation	–	(12544)
LeakyReLU	alpha=0.3	(12544)
Reshape	–	(7, 7, 256)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(7, 7, 128)
BatchNormalisation	–	(7, 7, 128)
LeakyReLU	alpha=0.3	(7, 7, 128)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(14, 14, 64)
BatchNormalisation	–	(14, 14, 64)
LeakyReLU	alpha=0.3	(14, 14, 64)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(28, 28, 32)
BatchNormalisation	–	(28, 28, 32)
LeakyReLU	alpha=0.3	(28, 28, 32)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(56, 56, 1)
Discriminator architecture		
InputLayer	–	(56, 56, 1)
Conv	kernel=5x5, stride=2, padding=same	(28, 28, 64)
LeakyReLU	alpha=0.3	(28, 28, 64)
Dropout	rate=0.2	(28, 28, 64)
Conv	kernel=5x5, stride=2, padding=same	(14, 14, 128)
LeakyReLU	alpha=0.3	(14, 14, 128)
Dropout	rate=0.2	(14, 14, 128)
Flatten	–	(25088)
Dense	–	(1)