DGA Detection Using Similarity-Preserving Bloom Encodings

Lasse Nitz
lasse.nitz@fit.fraunhofer.de
Fraunhofer FIT
Sankt Augustin, Germany
RWTH Aachen University
Aachen, Germany

Avikarsha Mandal
avikarsha.mandal@fit.fraunhofer.de
Fraunhofer FIT
Sankt Augustin, Germany

ABSTRACT

The sanitization of concise data samples can be challenging, as they do not provide a clear distinction between sensitive and non-sensitive parts within individual samples. In this context, traditional sanitization and anonymization measures are not applicable. We consider the detection of algorithmically generated domains through machine learning as an example of such a case, where the benign samples may leak sensitive information. Within this scenario, we evaluate the use of a similarity-preserving Bloom encoding technique to obscure the training samples.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; Malware and its mitigation; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Bloom encoding, sanitization, privacy-preserving data publishing, DGA detection

ACM Reference Format:
Lasse Nitz and Avikarsha Mandal. 2023. DGA Detection Using Similarity-Preserving Bloom Encodings. In *European Interdisciplinary Cybersecurity Conference (EICC 2023), June 14–15, 2023, Stavanger, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3590777.3590795

1 INTRODUCTION

For many applications related to classification, machine learning has become the go-to solution. Use cases involving sensitive training data and the rise of privacy regulations such as the GDPR, however, have led to concerns about potential leakage of sensitive information. While techniques such as federated learning or differentially private machine learning aim to address such concerns, their application requires to perform at least some machine learning tasks within an organization's trust boundary. A more general alternative is the sanitization of training data, which (if done successfully) does not just provide protection against leakage of sensitive information through attacks on trained classifiers (e.g., via membership inference attacks [12]), but also allows for a wider range of use cases, such as sharing of training data with other parties (e.g., to create more balanced training data sets) or the use of an honest-but-curious machine-learning-as-a-service (MLaaS) provider. Applying protective measures to the training data instead of the algorithms also simplifies explanation to non-technical people, as in-depth knowledge of the algorithms and techniques is not required to understand how respective measures have been applied. Nevertheless, training data sanitization also comes with various downsides. In some instances, altering training data can remove information which is crucial for classification, while in other instances it is not clear which parts of the training data are sensitive. Especially the classification of short strings can be problematic in this context, since they do not leave much information to remove.

As one example of such a use case, we consider the detection of algorithmically generated domains (AGDs), which has been motivated for finding bots in a monitored network. The key idea in this scenario is that bots use domain generation algorithms (DGAs) to generate domain names, which they then try to connect to. The command and control server, as the desired communication counterpart, is then registered under domains which are likely to be generated by the bots based on the properties of the used DGA. The ability to detect AGDs can hence help to identify infected machines by viewing AGDs as indicators of compromise.

Work in this area (e.g., [2]) has evaluated the use of machine learning classifiers trained on AGDs as malicious samples and non-existent (NX) domain requests as benign samples. However, as such benign NX requests can reveal information about browsing behavior as well as sensitive benign applications using AGDs (such as endpoint security software) in the network in which they have been collected in, they can be considered to be privacy-critical [1]. At the same time, both malicious and benign samples consist of relatively short strings, which do not follow a traditional format as it is required for the anonymization of table or graph data.

Our contribution: We propose the use of a collision-based Bloom encoding technique from the area of privacy-preserving record linkage (PPRL) [10] for the sanitization of training data. This approach obscures input samples while at the same time preserving similarity between the encoded samples, as a crucial property for machine learning applications. We further evaluate this approach in the use case of DGA detection, and discuss its shortcomings as well as potential extensions to address them.

The remainder of this work is structured as follows. In Section 2, we look at the context in which the Bloom encoding technique is used for PPRL, as well as basic properties provided by it. We then discuss its use for the purpose of sanitizing samples for DGA detection in Section 3. After this, we define two threat models in Section 4, followed by a conceptual discussion of privacy implications under consideration of these threat models in Section 5. In Section 6, we present the results of our initial practical evaluation of performing

EICC 2023, June 14–15, 2023, Stavanger, Norway

ACM ISBN 978-1-4503-9829-9/23/06.
https://doi.org/10.1145/3590777.3590795

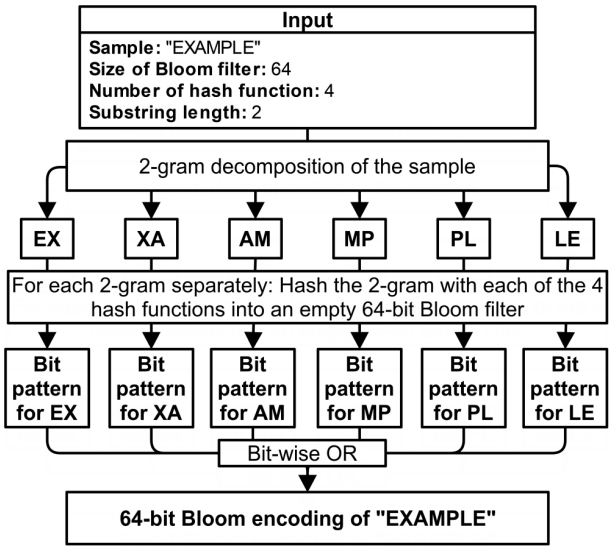

Figure 1: Visualized example of the flow for creating Bloom encodings without applying padding to the cleartext.

DGA detection on Bloom encodings. Lastly, we discuss our findings and outline potential directions for future work in Section 7.

2 BLOOM ENCODINGS IN PPRL

To understand how the requirements of machine learning on sensitive information relate to those of PPRL, we first introduce a general PPRL scenario as well as the approach proposed in [10], which we have used to encode the training samples for DGA detection.

The goal of PPRL is to enable linkage of records based on sensitive information without revealing the data on which the linkage is performed. We assume two data holders with their respective data sets D_1 and D_2, which consist of patient identities (e.g., a combination of first name, last name, date of birth, and ZIP code) and a medical attribute (e.g., heart rate or blood pressure), the latter of which differs between D_1 and D_2. If a research organization wanted to study the correlation between these two medical properties, it is necessary to link the medical attribute values based on the patient identities. However, as medical data is highly sensitive information, only the respective data holder should know a patient's identity.

The Bloom encoding approach proposed by [10] and depicted in Figure 1 can be used in a protocol, which encodes the identity tuples in a similarity-preserving way. Similarity of inputs is preserved, since not a full identity tuple is hashed into a Bloom filter, but its different substrings are hashed individually. This way the Bloom encodings of two inputs are similar, if their substring decompositions are similar. The encodings are sent to a linkage party, which links the medical attributes based on the encodings. The resulting mapping is then sent to the research institute without containing the identifying information in direct or encoded form.

In this scenario, the linkage party is considered to be honest-but-curious. Preserving similarity in this context is important, since the identifying information is considered to contain errors, such as typographical mistakes or inconsistent use of special characters. Preserving similarity in this context thus allows to also link records, for which the identifying information does not match exactly.

The Bloom encoding uses three primary parameters: The size of the Bloom filter l, the number of hash functions h, and the character length q for q-grams. These parameters can be used to provoke collisions between encodings. Intuitively, smaller l and larger h increase collisions. Larger values for h, however, also lead to more distinct bit patterns for q-grams, which can benefit decoding attempts. It further has been shown that smaller values for q are preferable from a privacy point of view [6].

3 APPLICATION IN DGA DETECTION

The first aspect to note is that preserving the similarity of encodings is crucial for machine learning. As DGA detection via machine learning aims to identify patterns in the training data, it is essential that such patterns are preserved across different sanitized samples. Creating one-way hashes of inputs or hashing domain names in full into a Bloom filter does not preserve such patterns, and can thus only be used for exact matching, rendering these approaches useless for machine learning use cases. In contrast to this, the approach of hashing individual q-grams into a Bloom filter implies that two input strings with similar q-grams also have similar Bloom encodings, meaning that some structural information is carried over to the encodings. Due to the basic properties of Bloom filters, however, collisions can occur, which also allow non-similar inputs to be mapped to similar or identical encodings.

It should also be noted that the Bloom encoding approach is sensitive to varying input length, as in general, shorter inputs result in sparser encodings (high fraction of 0-bits) while longer inputs result in dense ones (high fraction of 1-bits). As a consequence, the Bloom encoding parameters have to be chosen wisely to avoid that all long inputs result in the Bloom filter in which all bits are set. Further, also too sparse Bloom filters should be avoided, as they commonly include fewer collisions, which can benefit decoding attempts. This means that the Bloom encoding approach is not suitable, if inputs vary too much in length. For DGA detection, however, this is commonly not an issue.

One key aspect of the PPRL scenario is that the encoding parameters remain hidden from the honest-but-curious adversary. This differs from the requirements for various DGA-related use cases, as the encoding parameters are required to encode the training data set and any samples, which should be classified by a model trained on the encoded training data. Scenarios such as collaborative data set creation or sharing a trained classifier hence require the other parties to also know the encoding parameters. This introduces new attacks, which exploit the determinism as well as the known parameters to map a given encoding back to a set of inputs for which the encodings exactly match the given one. An example of this was formulated as a graph traversal problem [6].

Within the area of PPRL, the use of the permanent randomized response step of RAPPOR [4] has been proposed to harden the encodings [11]. The randomization is applied individually to the bits b_i of a Bloom encoding B for the randomization parameter $f \in [0, 1]$ to create a noisy version B' with bits b'_i. It is thus applied

as post-processing to the procedure depicted in Figure 1. Specifically, the noisy bits are generated as follows [4]:

$$b'_i = \begin{cases} 0, & \text{with probability } 0.5 \cdot f \\ 1, & \text{with probability } 0.5 \cdot f \\ b_i, & \text{with probability } 1 - f \end{cases}$$

The graph traversal attack presented in [6] relies on the determinism of the procedure and the basic Bloom filter property, which allows to prove that a specific element has not been hashed in it based on 0-bits. As this property is not preserved in the randomized encodings, the attack is no longer applicable, if an adequately high value for f is used.

4 THREAT MODELS AND ASSUMPTIONS

We consider two threat models, which align with different use cases. In both threat models, the adversary has access to the sanitized training data set (i.e., a set of Bloom encodings), in the following referred to as the target encodings, and the goal of the adversary is to fully decode at least some of these target encodings, i.e., to find the exact cleartext, which resulted in a given Bloom encoding. In the context of DGA detection, this would allow to reveal NX domains captured in an organization's network. The two different adversaries we consider, however, differ in the degree of available background knowledge:

(1) **Unknown parameter values:** The first adversary is considered to have knowledge of the encoding procedure and the set of characters, which can appear in cleartexts. The parameter values, however, are not known to the adversary, with the exception of the Bloom filter length l, as it is directly revealed by the target encodings.
(2) **Known parameter values:** The second adversary has access to all information the first adversary has, but additionally knows all parameter values used, except for the randomization parameter f (if randomization is used). This implies that the adversary can generate Bloom encodings for arbitrary inputs.

In regard to use cases, the capabilities of the first adversary align with the one of an honest-but-curious MLaaS provider, who trains a machine learning model based on Bloom encoded training data. For this, the adversary is not required to know the encoding parameters. In contrast to this, the second adversary is aligned with use cases such as the collaborative collection of a training data set, to which the adversary gets access (as an honest-but-curious collaborator, or a third party data consumer). In this case, the adversary needs access to the parameter values, since any samples which should be put into context with the encoded data set (e.g., as samples classified by a machine learning model trained on the encoded training data set) also need to be encoded.
Note that the chosen threat models assume the adversary to have direct access to the target encodings. The assumed adversaries are hence stronger than adversaries, who first have to extract the encodings from a trained classifier, e.g., via membership inference attacks.

5 PRIVACY IMPLICATIONS

These two threat models do not just align with different use cases. It is to note that the second threat model poses a notably larger threat to the encodings. The first threat model requires the adversary to first find out patterns of individual q-grams, before being able to generate encodings for chosen cleartexts. In contrast to this, the second threat model allows to generate encodings for arbitrary q-grams, also for arbitrary values of q.
For the first threat model, this means that taking measures to protect against the discovery of q-gram patterns also protect against decoding attempts. As a consequence, avoiding sparse Bloom encodings via the choice of the parameters l, h and q could be considered adequate in some settings. Using randomization would complicate the detection of q-gram patterns even further, since any bit position in any encoding could have potentially changed its value during randomization. It is to note that the application of RAPPOR's permanent randomized response step adds noise to each encoding independently. There hence is no consistent noise pattern. Consequently, discovering q-gram patterns based on individual sparse encodings will commonly not be possible, as sparse encodings of similar cleartexts would need to be contextualized to filter out noisy bit values. Thus, using randomization can allow for a few sparse encodings without significant risk of leaking q-gram patterns.
For the second threat model, it no longer suffices to protect against the discovery on q-gram patterns, as the adversary has all information necessary to generate encodings for arbitrary cleartexts. The level of protection provided hence depends on how the adversary can utilize the generated patterns to decode the target encodings. In this context, two abilities of the adversary become important:

(1) In a setting without randomization, the adversary can match the generated patterns against a target encoding. If, and only if, all bit positions which are set in the pattern of a q-gram are also set in the target encoding, the respective q-gram can potentially be part of the cleartext used to generate the target encoding. Note that this, however, does not necessarily guarantee that the q-gram was part of the original cleartext, as respective bits could have been set by patterns of other q-grams.
(2) Since we assume the adversary to know the encoding approach, the adversary also knows that the q-grams derived from a cleartext during the decomposition step are overlapping, meaning that the last $q-1$ characters of a q-gram are the same as the first $q-1$ characters of its successor q-gram. This allows to make some assumptions about the order of different q-grams with patterns matching a target encoding when reconstructing a cleartext.

Based on the combination of these two abilities it is possible to generate the set of all cleartexts, for which the encoding is exactly the same as a given target encoding [6]. While the clearxtext might be retrievable uniquely for sparse encodings, the set of potential cleartexts might contain hundreds of thousands or more cleartexts for dense ones. Nevertheless, relying only on colliding cleartexts should not be considered to provide adequate privacy protection under the second threat model. To impede this attack, randomization should be applied, as it disrupts the ability to rule out certain q-grams through pattern matching. Specifically, 0-bits can no longer prove

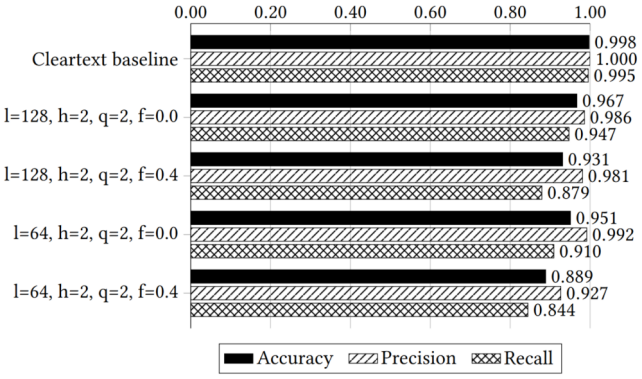

Figure 2: Evaluation of CNNs trained on the cleartext data set (as a baseline) and different encoded versions of it. The parameter values used for encoding the samples are given on the y-axis (l: Bloom filter size, h: number of hash functions, q: length of q-grams, f: randomization parameter of RAPPOR). An f-value of 0.0 implies that no randomization has been used. Each sample has been encoded in a separate Bloom filter.

that a specific q-gram is not included in a target encoding, if the randomization procedure can unset bits again. Randomization via RAPPOR's randomized response step hence targets and eliminates one of the key properties required for decoding encodings under the second threat model, and significantly improves the provided privacy guarantee.

6 EVALUATION

Since both the Bloom encoding procedure itself and the randomization applied to the generated encodings can produce colliding Bloom filters for differing cleartexts, the question about the impact on utility arises. We performed an initial practical evaluation by training convolutional neural networks (CNNs) on the cleartext samples (as a baseline) as well as on the encoded samples, which have been generated using different parameter values. The parameter values have been chosen to prioritize privacy over utility, by using relatively short Bloom filters, only two hash functions (resulting in less distinct q-gram patterns) and a decomposition into 2-grams, as it has been shown that smaller values for q provide stronger privacy guarantees [6].

For our initial evaluation, we used CNNs for the binary classification task (malicious/benign) in regard to AGDs. We used the NYU model with two stacked 1-dimensional CNN layers, each with 128 filters, as related work has shown that it performs well in the binary DGA classification task [14][2]. The model is featureless, which makes it possible to use the same approach for training on cleartexts as well as on encodings without substantial changes. This allows for a fair comparison when evaluating the impact of the encoding on the classification performance. We expect that also other featureless deep learning models could be used.

To provide comparability to other works, we used a data set kindly provided to us by the authors of [2][3]. It has been pre-processed as detailed in [3], and consists of 67 018 benign and 67 018 malicious samples. The benign samples are NX domain requests collected in a university environment, and also include AGDs generated by benign applications such as antivirus software. Since these AGDs are not tied to malicious activities, they were considered as benign and not removed from the data set. The malicious samples were taken from DGArchive[1] [8]. DGArchive provides a collection of AGDs generated by reverse engineered DGAs for known seeds. At the time of sampling, DGArchive contained roughly 100 million AGDs for 91 DGAs.

The data set used for our initial evaluation consists of 134 036 samples, 101 866 of which have been used for training, 5 362 as the holdout set, and 26 808 for the final evaluation of the trained classifier. Each of the data sets has a 50/50 label distribution with regard to AGDs. The training process was stopped, when the classifier's performance did not improve for three consecutive training epochs. The models have been trained via the implementation used in [2], which utilizes Python, Keras, and TensorFlow. The configuration of the used NYU model can be found in the appendix of [14], but was slightly modified to allow for inputs of up to 253 characters [2].

The results of our initial evaluation are depicted in Figure 2. The initial observation is that the encoding indeed negatively impacts the utility of classifiers. While the classifier trained on cleartext data achieves accuracy, precision and recall values beyond 0.99, none of the classifiers trained on sanitized data sets manages to score this high on each of the three metrics. Forcing collisions by using shorter Bloom filters further impacts the accuracy and recall negatively. The shorter Bloom filters also seem less resistant to the application of noise, as seen in the drastic impact on all three metrics for the noisy encodings of Bloom filter size 64. While these results are not unexpected, the small impact of the encoding on the precision of the classifiers does surprise. Except for the short noisy encodings, the sanitized versions hold up well, with precision values above 0.98.

7 DISCUSSION

The sharing of training data for DGA detection is just one of many use cases dealing with the broader question of how to sanitize data samples of which every part is potentially sensitive. AGDs (and their benign training sample counterparts) face the problem that they are too concise to make a distinction between sensitive and non-sensitive parts. In this case, traditional anonymization techniques such as k-anonymity [13] or l-diversity [5] are conceptually not applicable. While other approaches such as multi-party computation or differential privacy could be considered as alternatives, they specifically target the computation procedure, and not directly the data behind it. These approaches further come with their own sets of problems, such as a high communication overhead [1] or a high degree of noise required to fulfill formal guarantees.

With the Bloom encoding approach, which has been established for PPRL as an efficient alternative to cryptographic protocols, we have looked into a technique that tries to obscure data samples, while preserving similarity as an essential property for machine learning. The efficiency of the approach and the property of preserving similarity, however, come at the cost of weaker privacy guarantees

[1] https://dgarchive.caad.fkie.fraunhofer.de

compared to cryptographic approaches. Collisions between the encodings are important to obscure samples, but also impact utility. The same holds for the use of hardening techniques, such as the addition of noise. Our preliminary evaluation showed that both have a practical impact on the quality of the encoded data.
In the context of future work, a better understanding of guarantees provided by hardening techniques for Bloom encodings as well as their impact on utility is of interest. While various hardening approaches have been proposed for PPRL [9], some of them are not applicable in a setting in which the adversary is considered to know the encoding procedure and parameters. The approach could further be considered for scenarios, in which more traditional approaches to data sanitization are not applicable, such as the sanitization of URLs for phishing detection via machine learning [7]. Nevertheless, respective privacy requirements need to be considered carefully.

ACKNOWLEDGMENTS

We would like to thank the IT-Security Research Group at RWTH University for providing us access to their proof of concept data set and implementation for training DGA classifiers. This work has received funding from the EU H2020 projects SAPPAN under grant agreement No. 833418 and CyberSEAS under grant agreement No. 101020560.

REFERENCES

[1] Arthur Drichel, Mehdi Akbari Gurabi, Tim Amelung, and Ulrike Meyer. 2021. Towards Privacy-Preserving Classification-as-a-Service for DGA Detection. In *2021 18th International Conference on Privacy, Security and Trust (PST)*. 1–10. https://doi.org/10.1109/PST52912.2021.9647755

[2] Arthur Drichel, Ulrike Meyer, Samuel Schüppen, and Dominik Teubert. 2020. Analyzing the Real-World Applicability of DGA Classifiers. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 15, 11 pages. https://doi.org/10.1145/3407023.3407030

[3] Arthur Drichel, Ulrike Meyer, Samuel Schüppen, and Dominik Teubert. 2020. Making Use of NXt to Nothing: The Effect of Class Imbalances on DGA Detection Classifiers. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 85, 9 pages. https://doi.org/10.1145/3407023.3409190

[4] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067. https://doi.org/10.1145/2660267.2660348

[5] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. L-Diversity: Privacy beyond k-Anonymity. *ACM Trans. Knowl. Discov. Data* 1, 1 (mar 2007), 3–es. https://doi.org/10.1145/1217299.1217302

[6] William Mitchell, Rinku Dewri, Ramakrishna Thurimella, and Max Roschke. 2017. A graph traversal attack on Bloom filter-based medical data aggregation. *Int. J. Big Data Intell.* 4, 4 (2017), 217–226.

[7] Lasse Nitz, Mehdi Akbari Gurabi, Avikarsha Mandal, and Benjamin Heitmann. 2021. Towards Privacy-Preserving Sharing of Cyber Threat Intelligence for Effective Response and Recovery. *ERCIM NEWS* 126 (2021), 33–34.

[8] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*. 263–278.

[9] Thilina Ranbaduge and Rainer Schnell. 2020. Securing Bloom Filters for Privacy-Preserving Record Linkage. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 2185–2188. https://doi.org/10.1145/3340531.3412105

[10] Rainer Schnell, Tobias Bachteler, and Jörg Reiher. 2009. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making* 9, 1 (2009), 41.

[11] Rainer Schnell and Christian Borgs. 2016. Randomized Response and Balanced Bloom Filters for Privacy Preserving Record Linkage. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 218–224. https://doi.org/10.1109/ICDMW.2016.0038

[12] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[13] Latanya Sweeney. 2002. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[14] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. 2018. Character Level based Detection of DGA Domain Names. In *2018 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2018.8489147