2021 RIVF International Conference on Computing and Communication Technologies (RIVF) | 978-1-6654-0435-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/RIVF51545.2021.9642098

A GAN-based approach for password guessing

Bao Ngoc Vi
Le Quy Don Technical University
Hanoi, Vietnam
ngocvb@lqdtu.edu.vn

Nguyen Ngoc Tran
Le Quy Don Technical University
Hanoi, Vietnam
ngoctn@lqdtu.edu.vn

Trung Giap Vu The
Le Quy Don Technical University
Hanoi, Vietnam
vuthetrunggiap@lqdtu.edu.vn

***Abstract*—Password is the most widely used authenticate method. Individuals ordinarily have numerous passwords for their documents or devices, and, in some cases, they need to recover them with password guessing tools. Most popular guessing tools require a dictionary of common passwords to check with password hashes. Thus, generative adversarial networks (GANs) are suitable choices to automatically create a high-quality dictionary without any additional information from experts or password structures. One of the successful GAN-based models is the PassGAN. However, existing GAN-based models suffer from the discrete nature of passwords. Therefore, we proposed and evaluated two improvement of the PassGAN model to tackle this problem: the GS-PassGAN model using Gumbel-Softmax relaxation and the S-PassGAN using a smooth representation of a real password obtained by an additional Auto-Encoder. Experiment results on three different popular datasets show that the proposed method is better than the PassGAN both in the standalone and combining cases. Moreover, the matching rate of the proposed method can be increased by more than 5%.**

***Index Terms*—generative adversarial network ; password guessing; Gumbel-Softmax; Auto-Encoder**

I. Introduction

Nowadays, textual password remains a common method of authenticating users because of the easy implementation and no special hardware requirements. Individuals ordinarily have numerous passwords for their documents or devices but sometimes a few of their passwords are forgotten, and password guessing tools are required to recover them. Law enforcement agencies, such as police and prosecutors are often required to crack passwords whereas conducting examinations. Meanwhile, multiple password database leaks have shown that predictable passwords often chosen, which empowers password guessing tools being developed.

Most common password guessing tools, such as HashCat [7] and John the Ripper (JtR) [21] provide user ability to check billions of passwords per second against password hashes. But instead of trying all possible character's combinations (brute-force attack), these tools start with pre-defined dictionaries or leak passwords then expand these using generation rules (dictionary-based attack). Using rules to extend password dictionary has some drawbacks. Firstly, creating the rules requires the expert's knowledge. Secondly, these methods only generate limited wordlists. Therefore, researchers study how to automatically analyze leak passwords then generate candidates which are highly used. Recently, motivated by the success of deep learning, there are some deep learning - based method have been proposed to enhance dictionary. There are two main deep learning approaches: recurrent neural network (RNN)-based [14] and GAN-based [8], [16]). RNN-based models like language models suffer from exposure bias because at training time these models is exposed to gold data only, but at test time they observe their own predictions. Hence, wrong predictions quickly accumulate and result in poor text generation quality. GAN-based models suffer less from mentions above, but similar to other GAN-based text generators, these are challenging due to the discrete nature of the text. Back-propagation would not be feasible for discrete outputs and it is not straightforward to pass the gradients through the discrete output words of the generator.

Current works focus on dealing with the non-differentiable issue brought by the discrete data either by considering the Reinforcement Learning (RL) methods or by reformulating the problem in continuous space. A large class of GANs for text generation relies on the intensive RL heuristics [2], [23], [24]. These model often require a pre-trained model. For non-reinforcement approaches, some proposed methods used an addtional Auto-Encoder (AE). AE can be used to derive a latent space representation of the text then GAN model attempts to learn data manifold of that space [13] or to get the smooth representation of an input text before feeding it to GAN [5], [6]. Experiment results show that learning from "smooth text" is more efficient. However, these models require training an additional AE, which is time consuming and make training stage more complicated. Authors in [10] proposed the first GAN model using Gumbel Softmax technique, but they only provided experiment on the synthetic task, which generate the string from a simple artificial context-free grammar. RelGAN [18] is the first architecture to demonstrate that GANs with Gumbel-Softmax relaxation are capable of generating realistic text. However, they investigate under the relation memory based generator and discriminator with multiple embedded representations. Moreover, the generator and the discriminator are word-level models and pre-trained via Maximum Likelihood Estimates in advanced. Experiments to compare the performance of Gumbel-Softmax relaxation and the vanilla RL methods show that the variance of generator gradients in the vanilla RL methods is too large to provide any useful update for generator.

Therefore, in the context of generating password, in this paper, we proposed GS-PassGAN, S-PassGAN model which is an improvement of PassGAN with Gumbel-Softmax relaxation and smooth representation of real passwords. To the best of my knowledge, GS-PassGAN is the first CNN-based GAN model at character level, which applied Gumbel-Softmax relaxation for generate real sequences. The cracking performance is evaluated in cross-site scenarios. That is, the model is trained on Rockyou dataset [20] and testing in the different datasets. For testing phase, we use three different password datasets: LinkedIn [11], MySpace [15] and phpBb [19].

The rest of this paper is organized as follows. Section II shows the introduction of the traditional and deep learning approaches in the password guessing field. Section III presents our proposed models the GS-PassGAN and the S-PassGAN . Section IV shows the experimental setup and results. Finally, section V makes a conclusion and states future works.

II. Related work

A. Traditional Password Guessing Methods

In a secure system, the hashed value of passwords are stored in stead of their plain text. Therefore, hashing password tools such as HashCat [7] and JtR [21] are popular for recovering passwords. There are multiple options of password cracking method in both tools, such as brute-force attacks, dictionary-based attacks and rule-based attacks.

Markov model was first utilized for guessing password in 2005 by Narayanan et al. [17] and its improvement has been proposed recently [12]. They used given password rules, such as which portion of the generated passwords is composed of letters and numbers. Its improvement with Probabilistic Context-Free Grammars (PCFGs) was introduced by Weir et al. [22]. PCFG derives from word-mangling rules based on a given leak passwords. A passwords' dictionary then generated by the grammar examples with the trained probabilistic model.

A neural network was first used in password guessing by Ciaramella et al in [1]. Recently, Melicher et al. [14] proposed a password cracking model FLA based on RNNs. The experiment results show that a wordlist generated by FLA outperforms the Markov model, PCFG, Hashcat and JtR at guessing numbers above 10^{10}. However, the main purpose of these works consists in providing means for password strength estimation. In contrast, PassGAN [8] and its variants R-PassGAN [16] focus on the task of password guessing and attempts to do so with no a prior knowledge or assumption on the Markovian structure of user-chosen passwords.

B. GAN-based password guessing methods

GAN [3] has recently become an important generative model. The GAN training strategy is to define a game between two deep neural networks: a generator G and a discriminator D. The generator take a random noise z, normaly generated from normal distribution, as input. The generator then maps this noise into target distribution $\mathbb{P}_x$. The discriminator try to distinguish between a true data sample and a fake sample created by generator. While the generator is trained to created sample close to real data as much as possible to fool discriminator. After Goodfellow et al. [3] proposed the first GAN, various GAN models with better performance have been suggested including IWGAN [4].

Authors [4] also showed that the CNN based IWGAN model can be applied to a text generation area. PassGAN [8] was derived from these experimental results. The experimental results showed that PassGAN is competitive with the state-of-the-art password generation tools without any additional information such as rules or structure of passwords. Figure 1 shows the architecture of PassGAN's. In [16], authors proposed R-PassGAN which based on RNN. The experiment results show that this combination model perform better than PassGAN. However, the detailed structure of RNN was not stated.

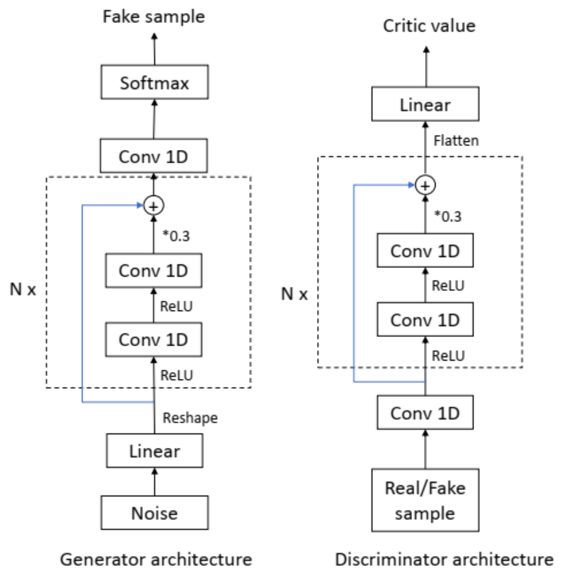


Fig. 1: Overview of PassGAN's structure

Training GAN-based models is challenging due to the discrete nature of passwords. In particular, a real password is encoded as a sequence of one-hot representations of its characters. However, a fake password, which is also encoded using a one-hot representation, can be only sampled from a multinomial distribution with probabilities given by the output of a softmax layer in generator. The resulting sampling process is not differentiable so in training time of PassGAN, output of a softmax layer is directly fed to discriminator. Then, discriminator is responsible for distinguishing the one-hot representation of real password from the softmax representation of the fake password. Thus discriminator is able to tell apart the one-hot input from the softmax input very easily. Therefore, it is difficult to generator to fool discriminator and vanishing gradient problem is highly probable. As stated before, there are some approaches to tackle this issue. Among these, Gumbel-Softmax relaxation and learning from smooth representation of password obtained from output of AE are time-saving and competitive methods. A detail of these methods is represented in the next section.

III. Methodology

A. Gumbel-Softmax relaxation

As stated above, PassGAN feeds output of a softmax vector directly to discriminator, instead of one-hot representation due to non-differentiable of sampling form multi nominal distribution.

Let h^j be the j^{th} output logits n-dimension vector of the generator where $j = \overline{0, m}$ with m is length of generated password and n is the number of characters in the vocabulary. The one-hot representation of j^{th} character y^j will obtained by sampling:

$$y^j \sim softmax(h^j) \tag{1}$$

This sampling operator is not differentiable. If using y^j the gradients of the generator loss cannot pass back to the generator via the discriminator. Then, PassGAN use $softmax(h^j)$ instead. But it is not a good approximation of sampling operator. In [9], Gumbel-Max trick is proposed to re-parameterize the sampling operator as below:

$$y^j = one\_hot(argmax(h_i^j + g_i)) \tag{2}$$

where the g_i is an independent variable and follow a Gumbel distribution with zero location and unit scale. But, "one hot argmax" operator still non-differentiable, an approximation using softmax with temperature is proposed. Denote g be the vector which consists of n elements g_i ($i = \overline{0, m}$), then the approximation is represented as below:

$$\widehat{y}^j = softmax(\frac{1}{\tau}(h^j + g)) \tag{3}$$

where

$$softmax(\frac{1}{\tau}(h^j + g))_i = \frac{e^{\frac{1}{\tau}(h_i^j + g_i)}}{\sum_k e^{\frac{1}{\tau}(h_k^j + g_k)}}$$

and $\tau > 0$ is a tunable parameter called temperature. As y^j is s differentiable with respect to h^j, it can used instead of y^j. When $\tau \to 0$, $\widehat{y}^j$ is one-hot representation. However, authors in [9] showed that the variance of gradients will be very large as $\text{Var}(\frac{\partial \widehat{y}^j}{\partial h^j}) \propto \frac{1}{\tau^2}$ thus the parameter updates will become very sensitive to the input noise. Intuitively, this might cause poor sample quality. In contrast, with larger temperature τ, the generator will pay more attention to making sharp distribution of entries in $\widehat{y}^{t+1}$ due to the larger (initial) approximation gap between $\widehat{y}^{t+1}$ and y^{t+1}.

B. Smooth representation of real passwords with Auto-Encoder

This ideal inspires of the knowledge distillation model in [5]. In this model, the generator (Student) tries to learn the distribution of the reconstructed output representation of an Auto-encoder (Teacher), which is continuous, instead of the distribution of one-hot representations. Overview of S-PassGAN is showed in Figure 2. The softmax output of the generator still feeds to the discriminator. However, rather than directly feeding to the discriminator, the one-hot representation of a password is firstly fed to the AE, then its output is sent to the discriminator.S-PassGAn train AE and GAN alternately with AE reconstruction loss and min-max loss of GAN.

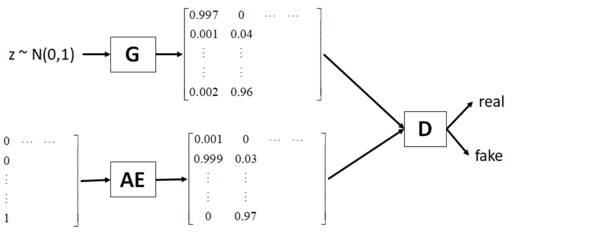


Fig. 2: S-PassGAN with smooth representation from AE

IV. Experiments

A. Datasets

To evaluate our proposed model, we train the models with Rockyou [20], and evaluate the caracking performance in three differen datasets including Myspace [15], phpBB [19] and LinkedIn [11]. Rockyou is chosen as training dataset because it is a rich dataset with frequency counts. We selected all passwords of length 10 characters or less with printable ACSII characters (95 characters in total). Detail of all datasets is represented in table I.

B. Experiment setup

For GS-PassGAN, during the training phase, we linearly anneal the temperature τ of the Gumbel-softmax relaxation from 0.1, and reaches 0.01 at iteration $100,000$ and then kept at $\tau = 0.01$ until training ends. For S-PassGAN, AE is trained with one layer with 512 LSTM cells for both the encoder and the decoder. The parameters of AE and GAN are alternately updated. And difference from GS-PassGAN, at each training step, the parameters of the discriminator is updated 5 times in S-PassGAN. Other hyper-parameters are the same with PassGAN as show in Table II.

Batch size	64
Number of iterations	200,000
Ratio of update D/G	10:1
Number of residual blocks	5
Gradient penalty coefficient	10
Size of noise	128
Learning rate	1e-4
β_1 (Adam optimizer)	0.5
β_2 (Adam optimizer)	0.9

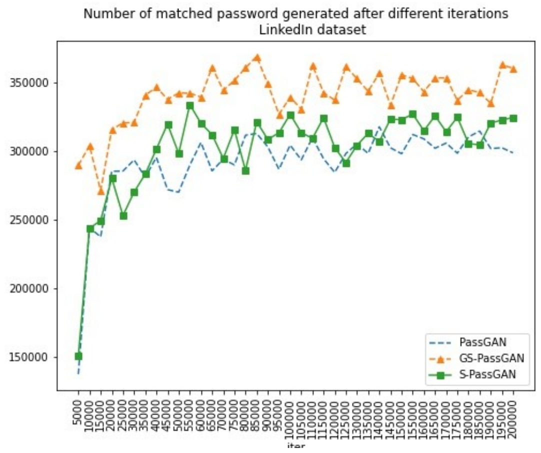
TABLE II: Hyper-parameters setting in our models

C. Results and Discussion

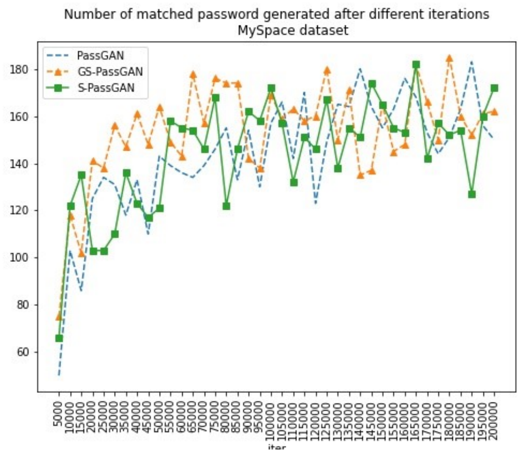
Our proposed models and PassGAN are compared in term of the number of password cracked. Firstly, the number of matched passwords after each $5,000$ iterations of generator's parameters updated are compared. For each step, 10^7 passwords is generated. Then the lists is enumerated if they were listed in testing set and calculate the number of matched passwords. Before that, we remove passwords in three test datasets, which is contained in training dataset. Thus, we obtained

Datasets	Number of password	Number of unique passwords	Number of password created from 95 printable ASCII characters			
			Number of passwords	Number of unique password	Number of passwords with length <= 10	Number of unique passwords with length <= 10
Rockyou	32,603,388	14,344,391	32,585,350	14,329,850	29,591,069	11,899,053
MySpace	41,545	37,144	41,537	37,136	39,145	34,808
PhpBB	255,421	184,389	255,376	184,344	245,268	174,922
LinkedIn	60,637,684	60,637,572	60,637,684	60,637,572	43,390,943	43,390,930

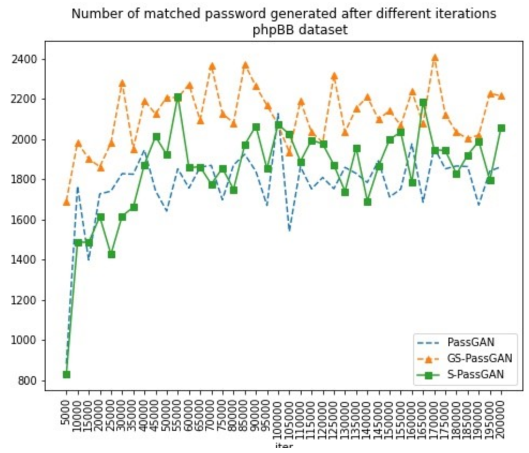
TABLE I: Leak passwords datasets



(a)



(b)



(c)

Fig. 3: Number of matched passwords in three different testing datasets after each 5000 iterations of generator's parameters updated

three new test datasets: LinkeIn test set with $40,255,271$ passwords, MySpace test set with $16,734$ passwords and phpBB test set with $104,288$ passwords. Figure 3 show these results. The results show that GS-PassGAN outperform other models excepted when cracking MySpace dataset. However, GS-PassGAN obtain higher number of matched passwords at early step than other models. While the number of matched passwords S-PassGAN only slightly higher than this of baseline model.

Morever, similar to the work in [8], we also test the performance when combining our models with Hashcat. For each model a dictionary of 10^9 passwords are generated then it is be combined with the dictionary created by HashCat Best64 to make new password dictionary. Table 3 show the results. It can be seen that our model is better than the baseline model both in the standalone and combining cases. The matching rate

Datasets	**HashCat only**	**PassGAN only**	**HashCat + PassGAN**	**S-PassGAN only**	**HashCat + S-PassGAN**	**GS-PassGAN only**	**HashCat + GS-PassGAN**
LinkedIn	7,857,580 (19.52%)	2,893,152 (7.19%)	8,961,578 (22.26%)	3,316,644 (8.24%)	9,136,601 (22.70%)	4,265,046 (10.60%)	**9,565,156 (23.76%)**
PhpBB	30,238 (29.00%)	12,103 (11.61%)	33,204 (31.84%)	14,065 (13.49%)	33,613 (32.23%)	17,667 (16.94%)	**34,945 (33.51%)**
MySpace	6,067 (36.26%)	1,414 (8.45%)	6,425 (38.40%)	1,539 (9.20%)	6,453 (38.56%)	1,848 (11.04%)	**6,524 (38.99%)**

TABLE III: The number of matched passwords of the models when combining with HashCat

of the proposed method can be increased by more than 5% from 11.61% to 16.94% in comparison to the one of baseline model in the standalone case of PhpBb dataset. The highest matching rate is archived on MySpace when combining with Hashcat.

V. Conclusion and future Work

We proposed two improvements GS-PassGAN and S-PassGAN of PassGAN with two different techniques to deal with discrete nature of text in GAN model. GS-PassGAN uses Gumbel-Softmax relaxation to get a continuous approximation of sampling operation. While in stead of directly learning the distribution of one-hot representation of passwords, S-PassGAN tries to learn the distribution of "soft" representation which is a reconstruct output of an Auto-Encoder. The experiment results show that, our models outperform the baseline model in term of the number of guessed passwords. GS-PassGAN, the model with simple improvements archived the best performance when testing in three different common datasets including LinkedIn, MySpace, and phpBb. However, choosing suitable temperature and reduce the variance of the gradient of Gumbell trick approximation are challenging problems and need to be explored further in future.

References

[1] A. Ciaramella, P. D'Arco, A. De Santis, C. Galdi, and R. Tagliaferri. Neural network techniques for proactive password checking. *IEEE Transactions on Dependable and Secure Computing*, 3(4):327–339, 2006.

[2] W. Fedus, I. J. Goodfellow, and A. M. Dai. Maskgan: Better text generation via filling in the ______. *ArXiv*, abs/1801.07736, 2018.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777. Curran Associates, Inc., 2017.

[5] M. A. Haidar and M. Rezagholizadeh. Textkd-gan: Text generation using knowledge distillation and generative adversarial networks. In *Canadian Conference on AI*, 2019.

[6] M. A. Haidar, M. Rezagholizadeh, A. Do-Omri, and A. Rashid. Latent code and text-based generative adversarial networks for soft-text generation. In *NAACL-HLT*, 2020.

[7] HashCat. https://hashcat.net/hashcat/. Accessed: 2020-04-18.

[8] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz. *PassGAN: A Deep Learning Approach for Password Guessing*, pages 217–237. 2019.

[9] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax, 2017.

[10] M. J. Kusner and J. M. Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *ArXiv*, abs/1611.04051, 2016.

[11] LinkeIn. https://hashes.org/leaks.php?id=68. Accessed: 2020-04-18.

[12] J. Ma, W. Yang, M. Luo, and N. Li. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy*, pages 689–704. IEEE, 2014.

[13] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow. Adversarial autoencoders. *ArXiv*, abs/1511.05644, 2015.

[14] W. Melicher, B. Ur, S. Segreti, S. Komanduri, N. Christin, and L. Crano. *Fast, lean, and accurate: Modeling password guessability using neural networks*. USENIX Security Symposium. 2016.

[15] MySpace. https://downloads.skullsecurity.org/passwords/. Accessed: 2020-04-18.

[16] S. Nam, S. Jeon, and J. Moon. A new password cracking model with generative adversarial networks. In I. You, editor, *Information Security Applications*, pages 247–258. Springer International Publishing, 2019.

[17] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 364–372, 2005.

[18] W. Nie, N. Narodytska, and A. B. Patel. Relgan: Relational generative adversarial networks for text generation. In *ICLR*, 2019.

[19] PhpBb. https://downloads.skullsecurity.org/passwords/. Accessed: 2020-04-18.

[20] Rockyou. https://downloads.skullsecurity.org/passwords/. Accessed: 2020-04-18.

[21] J. the Ripper. https://www.openwall.com/john/. Accessed: 2020-04-18.

[22] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek. *Password Cracking Using Probabilistic Context-Free Grammars*. 2009.

[23] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[24] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin. Adversarial feature matching for text generation. volume 70 of *Proceedings of Machine Learning Research*, pages 4006–4015. PMLR, 06–11 Aug 2017.