PassGAN: A Deep Learning Approach for Password Guessing

Briland Hitaj
Department of Computer Science
Stevens Institute of Technology
bhitaj@stevens.edu

Paolo Gasti
Department of Computer Science
New York Institute of Technology
pgasti@nyit.edu

Giuseppe Ateniese
Department of Computer Science
Stevens Institute of Technology
gatenies@stevens.edu

Fernando Perez-Cruz
Swiss Data Science Center
ETH Zurich and EPFL
fernando.perezcruz@sdsc.ethz.ch

Abstract

State-of-the-art password guessing tools, such as HashCat and John the Ripper, enable users to check billions of passwords per second against password hashes. In addition to performing straightforward dictionary attacks, these tools can expand password dictionaries using password generation rules, such as concatenation of words and *leet speak*. Although these rules work well in practice, creating and updating them is a labor-intensive task that requires specialized expertise.

To address this issue, in this paper we introduce PassGAN, a novel approach that replaces human-generated password rules with theory-grounded machine learning algorithms. PassGAN autonomously learns the distribution of real passwords from actual password leaks, and generates high-quality password guesses without any a-priori knowledge on passwords or common password structures. When we combined the output of PassGAN with the output of HashCat, we were able to match 51% more passwords than with HashCat alone, showing that PassGAN can autonomously extract a considerable number of password properties that current state-of-the art rules do not encode.

1 Introduction

Passwords are the most popular authentication methods. Unfortunately, multiple password database leaks have shown that users tend to choose easy-to-guess passwords [12, 15, 32], primarily composed of common strings (e.g., `password`, `123456`, `iloveyou`), and variants thereof. Password guessing tools exploit this structure to generate highly likely password guess. However, effective password guessing is primarily based on carefully-crafted manually-generated rules. Defining these rules is a labor-intensive task that demands specialized expertise. To address this issue, in this paper we introduce PassGAN, a new approach for generating password guesses based on Generative Adversarial Networks (GANs) [17]. PassGAN represents a principled and theory-grounded take on the generation of password guesses.

Contributions. **1)** We show that a properly-trained GAN can generate high-quality password guesses. In our experiments, we were able to match 1,350,178 (43.6%) of 3,094,199 *unique* passwords from a testing set composed of real user passwords from RockYou dataset [50]; **2)** To verify the ability of PassGAN to create unseen passwords, we removed from the testing set those samples that were present in the training set. This operation resulted in a subset of 1,978,367 passwords out of

NeurIPS 2018 Workshop on Security in Machine Learning, (SECML 2018), Montréal, Canada.

which, PassGAN was able to match 676,439 (34.6%) of the samples.; **3)** We show that PassGAN is competitive with state-of-the-art password generation rules.; **4)** With password generation rules, the number of unique passwords that can be generated is defined by the number of rules and by the size of the password dataset used to instantiate them. In contrast, PassGAN can output a practically unbounded number of password guesses. Crucially, our experiments show that with PassGAN the number of matches increases steadily with the number of passwords generated, Table 1. This is important, because it shows that the output of PassGAN is not restricted to a small subset of the password space.; **5)** PassGAN is competitive with current state of the art in password guessing algorithms based on neural networks [34], matching the performance of Melicher et al. [34], (indicated as FLA in the rest of the paper).; **6)** We show that PassGAN can be effectively used to *complement* password generation rules. When we combined the output of PassGAN with the output of HashCat, we were able to guess 51% additional unique passwords compared to HashCat alone.

The ability of PassGAN to autonomously learn characteristics and patterns constituting a password drew significant attention from several media outlets, see [25, 59, 33, 61, 36, 22, 20]. We consider this work the first step towards fully automated generation of high-quality password guesses, and to the best of our knowledge, this work is the first to use GANs for this purpose.

2 Background and Related Work

2.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) represent a remarkable advance in the area of deep learning. A GAN is composed of two neural networks, a generative deep neural network G, and a discriminative deep neural network D. Given an input dataset $\mathcal{I} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, the goal of G is to produce "fake" samples from the underlying probability distribution $\Pr(\mathbf{x})$, that are accepted by D. At the same time, D's goal is to learn to distinguish fake samples from G from the real ones coming from $\mathcal{I}$. More formally, on input a simple noise distribution $\mathbf{z}$, the optimization problem solved by GANs can be summarized as follows:

$$\min_{\theta_G} \max_{\theta_D} \sum_{i=1}^{n} \log f(\mathbf{x}_i; \theta_D) + \sum_{j=1}^{n} \log(1 - f(g(\mathbf{z}_j; \theta_G); \theta_D)) \quad (1)$$

where the model attempts to minimize with respect to θ_G, and simultaneously maximize with respect to θ_D. The learning phase is considered complete when D is unable to distinguish between the fake samples produced by G, and the real samples from $\mathcal{I}$. Since the original work by Goodfellow et al. [17], there have been several improvements on GANs, [45, 2, 3, 49, 55, 19, 31, 41, 60, 35, 43, 51, 42, 4, 26, 37, 9, 13, 65, 62, 46, 24, 7, 40, 6, 11, 23, 38, 66], where each new paper provides novel improvements in the domain. In this paper, we rely on IWGAN [19] as a building foundation for PassGAN, being that IWGAN [19] is among the first, most stable approaches for text generation via GANs.

2.2 Password Guessing

Password guessing attacks are probably as old as password themselves [5], with more formal studies dating back to 1979 [39]. In a password guessing attack, the adversary attempts to identify the password of one or more users by repeatedly testing multiple candidate passwords.

Two popular modern password guessing tools are John the Ripper (JTR) [57] and HashCat [21]. Both tools implement multiple types of password guessing strategies, including: exhaustive brute-force attacks; dictionary-based attacks; rule-based attacks, which consist in generating password guesses from transformations of dictionary words [53, 52]; and Markov-model-based attacks [58, 47]. JTR and HashCat are notably effective at guessing passwords. Specifically, there have been several instances in which well over 90% of the password leaked from online services have been successfully recovered [48].

Markov models were first used to generate password guesses by Narayanan et al. [44]. Their approach uses manually defined password rules, such as which portion of the generated passwords is composed of letters and numbers. This technique was subsequently improved by Weir et al. [64] with Probabilistic Context-Free Grammars (PCFGs). With PCFGs, Weir et al. [64] demonstrated how to

Passwords Generated	Unique Passwords	Passwords matched in testing set, and not in training set (1,978,367 unique samples)
10^4	9,738	103 (0.005%)
10^5	94,400	957 (0.048%)
10^6	855,972	7,543 (0.381%)
10^7	7,064,483	40,320 (2.038%)
10^8	52,815,412	133,061 (6.726%)
10^9	356,216,832	298,608 (15.094%)
10^{10}	2,152,819,961	515,079 (26.036%)
$2 \cdot 10^{10}$	3,617,982,306	584,466 (29.543%)
$3 \cdot 10^{10}$	4,877,585,915	625,245 (31.604%)
$4 \cdot 10^{10}$	6,015,716,395	653,978 (33.056%)
$5 \cdot 10^{10}$	7,069,285,569	676,439 (34.192%)

Table 1: Number of passwords generated by PassGAN that match passwords in the RockYou testing set. Results are shown in terms of unique matches.

Approach	(1) Unique Passwords	(2) Matches
JTR Spyderlab	10^9	461,395 (23.32%)
Markov Model 3-gram	$4.9 \cdot 10^8$	532,961 (26.93%)
HashCat gen2	10^9	597,899 (30.22%)
HashCat Best64	$3.6 \cdot 10^8$	630,068 (31.84%)
PCFG	10^{10}	650,695 (32.89%)
FLA (10^{-10})	$7.4 \cdot 10^8$	652,585 (32.99%)
PassGAN	$2.1 \cdot 10^9$	515,079 (26.04%)
PassGAN	$3.6 \cdot 10^9$	584,466 (29.54%)
PassGAN	$4.9 \cdot 10^9$	625,245 (31.60%)
PassGAN	$6.0 \cdot 10^9$	653,978 (33.06%)
PassGAN	$7.1 \cdot 10^9$	676,439 (34.19%)

Table 2: Number of matches generated by each password guessing tool against the RockYou testing set, and corresponding number of password generated by PassGAN to outperform each tool.

"learn" these rules from password distributions. This early work has been subsequently extended by Ma et al. [32] and by Durmuth et al. [15].

The first work in the domain of passwords utilizing neural networks dates back to 2006 by Ciaramella et al. [10]. Recently, Melicher et al. [34] introduced FLA, a password guessing method based on recurrent neural networks [18, 56]. However, the primary goal of these works consists in providing means for password strength estimation. In contrast, PassGAN focuses on the task of password guessing and attempts to do so with no a-priori knowledge or assumption on the Markovian structure of user-chosen passwords.

3 Evaluation

3.1 GAN Training and Testing

To evaluate the performance of PassGAN, and to compare it with state-of-the-art password generation rules, we first trained the GAN, as well as JTR, HashCat, the Markov model, PCFG, and FLA on a large set of passwords from the RockYou password leak [50].[1] Entries in this dataset represent a mixture of common and complex passwords.The RockYou dataset [50] contains 32,503,388 passwords. We selected all passwords of length 10 characters or less (29,599,680 passwords, which correspond to 90.8% of the dataset), and used 80% of them (23,679,744 total passwords, 9,926,278 unique passwords) to train each password guessing tool. We refer the reader to the Appendix Section for further details on the training procedure of each tool. For testing, we computed the difference between the remaining 20% of the dataset (5,919,936 total passwords, 3,094,199 unique passwords) and the training test. The resulting 1,978,367 entries correspond to passwords that were not previously observed by the password guessing tools. This allowed us to count only non-trivial matches in the testing set.

3.2 PassGAN's Output Space

To evaluate the size of the password space generated by PassGAN, we generated several password sets of sizes between 10^4 and 10^{10}. Our experiments show that, as the number of passwords increased, so did the number of unique (and therefore new) passwords. Results of this evaluation are reported in Table 1.

When we increased the number of passwords generated by PassGAN, the rate at which new unique passwords were generated decreased only slightly. Similarly, the rate of increase of the number of matches (shown in Table 1) diminished slightly as the number of passwords generated increased. This is to be expected, as the simpler passwords are matched early on, and the remaining (more complex) passwords require a substantially larger number of attempts in order to be matched.

[1]We consider the use of publicly available password datasets to be ethical, and consistent with security research best practices (see, e.g., [12, 34, 8]).

3.3 Evaluating the Passwords Generated by PassGAN

To evaluate the quality of the output of PassGAN, we generated $5 \cdot 10^{10}$ passwords, out of which roughly $7 \cdot 10^9$ were unique. We compared these passwords with the outputs of length 10 characters or less from HashCat Best64 and gen2, JTR SpiderLab, FLA, PCFG, and Markov model.

In our comparisons, we aimed at establishing whether PassGAN was able to meet the performance of the other tools. Our results show that, for each of the tools, PassGAN was able to generate a competitive number of matches, Table 2. However, in many cases it underperformed the best known methods with respect to the number of passwords needed to achieve a particular number of matches. This is not unexpected, because while other tools rely on prior knowledge on passwords for guessing, PassGAN does not. Nonetheless, the results presented in Table 2 depict the effectiveness of PassGAN as an *unsupervised* password guessing approach, as well as show that such a GAN based password guessing learning mechanism can substantially benefit from further improvements.

3.4 Combining the Output of PassGAN with that of other Password Guessing Tools

We also focused our evaluation on the following question: *How would the combination of the output of rule-based password guessing approaches with machine-learning based ones affect the guessing performance?* Our hypothesis was that, although rule-based tools are fast and effective when guessing passwords that follow the rules on which these tools rely, machine learning tools might be able to match additional passwords, at the cost of a larger number of attempts. To test this hypothesis, we removed all passwords matched by HashCat Best64 (the best performing set of rules in our experiments) from RockYou testing set. This led to a new test set, containing 1,348,300 passwords. We then calculated how many additional matches PassGAN was able to achieve. With $7 \cdot 10^9$ unique samples generated, PassGAN was able to match 320,365 (23.76%) passwords from the new testing set, which translates to an additional 51% passwords matched from the RockYou dataset compared to HashCat alone. Therefore, our results indicate that combining rules with machine learning password guessing is an effective strategy, as ML based strategies are able to capture different portions of the password space.

3.5 Comparing PassGAN with FLA

To investigate further on the differences between PassGAN and FLA, we computed the number of passwords in the RockYou testing set that PassGAN was able to guess within its first $7 \cdot 10^9$ samples, and for which FLA required at least 10^{10} attempts. These are the passwords to which FLA assigns low probabilities, despite being chosen by some users. Because PassGAN is able to model them, we conclude that the probabilities assigned by FLA to these passwords are incorrect. Figure 1, (moved to the Appendix Section due to space limitations), presents our result as the ratio between the passwords matched by FLA at a particular number of guessing attempts, and by PassGAN within its first $7 \cdot 10^9$ attempts. These results show that PassGAN is able to model a number of passwords more correctly than FLA. However, this advantage decreased as the number of attempts required for FLA to guess a password increased, i.e., as the estimated probability of that password decreased. This shows that, in general, the two tools agree on assigning probabilities to passwords.

4 Conclusions and Remarks

In this paper we introduced PassGAN, the first application of generative adversarial networks (GANs) to password guessing. In contrast to currents password guessing tools, PassGAN generates highly-likely passwords without relying on additional information on passwords (e.g., explicit rules), or on assumptions on the Markovian structure of user-chosen passwords. Our results show that PassGAN is competitive with state of the art password generation tools: in our experiments, PassGAN was always able to generate the same number of matches as the other password guessing tools. At present, PassGAN requires to output a larger number of passwords compared to other tools to achieve the same number of matches. However, we believe that by training PassGAN on a larger dataset (which also allows us to deploy more complex neural network structures and more comprehensive training), the underlying GAN will perform more accurate density estimation, thus reducing the number of passwords needed to achieve a specific number of matches.

Appendix

GAN Architecture and Hyperparameters

To leverage the ability of GANs to effectively estimate the probability distribution of passwords from the training set, we experimented with a variety of parameters. In this section, we report our choices on specific GAN architecture and hyperparameters.

We instantiated PassGAN using the *Improved training of Wasserstein GANs* (IWGAN) of Gulrajani et al. [19]. The IWGAN implementation used in this paper relies on the ADAM optimizer [27] to minimize the training error, i.e., to reduce the mismatch between the output of the model and its training data.

Our model is characterized by the following hyper-parameters:

- **Batch size**, which represents the number of passwords from the training set that propagate through the GAN at each step of the optimizer. We instantiated our model with a batch size of 64.
- **Number of iterations**, which indicates how many times the GAN invokes its forward step and its back-propagation step [54, 29, 30]. In each iteration, the GAN runs one generator iteration and one or more discriminator iterations. We trained the GAN using various number of iterations and eventually settled for 199,000 iterations, as further iterations provided diminishing returns in the number of matches.
- **Number of discriminator iterations per generator iteration**, which indicates how many iterations the generator performs in each GAN iteration. The number of discriminator iterations per generative iteration was set to 10, which is the default value used by IWGAN.
- **Model dimensionality**, which represents the number of dimensions (weights) for each convolutional layer. We experimented using 5 residual layers for both the generator and the discriminator, with each of the layers in both deep neural network having 128 dimensions.
- **Gradient penalty coefficient** (λ), which specifies the penalty applied to the norm of the gradient of the discriminator with respect to its input [19]. Increasing this parameter leads to a more stable training of the GAN [19]. In our experiments, we set the value of gradient penalty to 10.
- **Output sequence length**, which indicates the maximum length of the strings generated by the generator (G). We modified the length of the sequence generated by the GAN from 32 characters (default length for IWGAN) to 10 characters, to match the maximum length of passwords used during training.
- **Size of the input noise vector (seed)**, which determines how many random bits are fed as input to G for the purpose of generating samples. We set the size of the noise vector to 128 floating point numbers.
- **Maximum number of examples**, which represents the maximum number of training items (passwords, in the case of PassGAN) to load. The maximum number of examples loaded by the GAN was set to the size of the entire training dataset.
- **Adam optimizer's hyper-parameters**:
  - **Learning rate**, i.e., how quickly the weights of the model are adjusted
  - **Coefficient** β_1, which specifies the decaying rate of the running average of the gradient.
  - **Coefficient** β_2, which indicates the decaying rate of the running average of the square of the gradient.

  Coefficients β_1 and β_2 of the Adam optimizer were set to 0.5 and 0.9, respectively, while the learning rate was 10^{-4}. These parameters are the default values used by Gulrajani et al. [19].

Our experiments were run using the TensorFlow implementation of IWGAN. We used TensorFlow version 1.2.1 for GPUs [1], with Python version 2.7.12. All experiments were performed on a workstation running Ubuntu 16.04.2 LTS, with 64GB of RAM, a 12-core 2.0 GHz Intel Xeon CPU, and an NVIDIA GeForce GTX 1080 Ti GPU with 11GB of global memory.

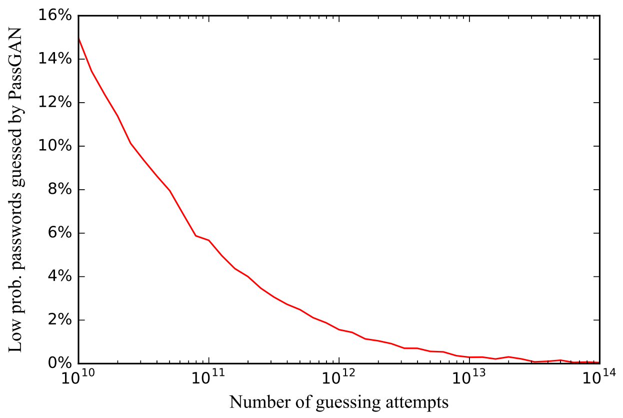

Figure 1: Percentage of passwords matched by FLA at a particular number of guesses, that are matched by PassGAN in at most $7 \cdot 10^9$ attempts.

Password Sampling Procedure for HashCat, JTR, Markov Model, PCFG and FLA

We used the portion of RockYou dataset utilized to train PassGAN, Section 3.1, as the input dataset to HashCat Best64, HashCat gen2, JTR Spiderlab rules, Markov Model, PCFG and FLA. The output passwords from each tool were computed as follows:

- We instantiated HashCat and JTR's rules using passwords from the training set sorted by frequency in descending order (as in [34]). HashCat Best64 generated 754,315,842 passwords, out of which 361,728,683 were unique and of length 10 characters or less. Note that this was the maximum number of samples produced by Best64 rule-set for the given input set, i.e. RockYou training set. With HashCat gen2 and JTR SpiderLab we uniformly sampled a random subset of size 10^9 from their output. This subset was composed of passwords of length 10 characters or less.
- For FLA, we set up the code from [28] according to the instruction provided at [16]. We trained a model containing 2-hidden layers and 1 dense layer of size 512. We did not perform any transformation (e.g., removing symbols, or transforming all characters to lowercase) on the training set for the sake of consistency with the other tools. Once trained, FLA enumerates a subset of its output space defined by a probability threshold p. A password belongs to FLA's output only if its probability is at least p. In our experiments, we set $p = 10^{-10}$. This resulted in a total of 747,542,984 passwords of length 10 characters or less. Before using these passwords in our evaluation, we sorted them by probability in descending order.
- We generated 494,369,794 unique passwords of length 10 or less using the 3-gram Markov model. We ran this model using its standard configuration [14].
- We generated 10^{10} unique passwords of length 10 or less using the PCFG implementation of Weir et al. [63].

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Martin Arjovsky and Leon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017.

[4] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[5] Hossein Bidgoli. *Handbook of Information Security, Information Warfare, Social, Legal, and International Issues and Security Foundations*, volume 2. John Wiley & Sons, 2006.

[6] Mikolaj Binkowski, Dougal Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. *International Conference on Learning Representations (ICLR)*, 2018.

[7] Yanshuai Cao, Gavin Weiguang Ding, Yik Chau Lui, and Ruitong Huang. Improving GAN training via binarized representation entropy (BRE) regularization. *International Conference on Learning Representations (ICLR)*, 2018.

[8] Claude Castelluccia, Markus Dürmuth, and Daniele Perito. Adaptive password-strength meters from markov models. In *NDSS*, 2012.

[9] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[10] Angelo Ciaramella, Paolo D'Arco, Alfredo De Santis, Clemente Galdi, and Roberto Tagliaferri. Neural network techniques for proactive password checking. *IEEE Transactions on Dependable and Secure Computing*, 3(4):327–339, 2006.

[11] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. *International Conference on Learning Representations (ICLR)*, 2018.

[12] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[13] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[14] Brannon Dorsey. Markov-chain password generator. `https://github.com/brannondorsey/markov-passwords`, 2017.

[15] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Chaabane Abdelberi. OMEN: Faster password guessing using an ordered markov enumerator. In *ESSoS*, pages 119–132. Springer, 2015.

[16] Maximilian Golla. Password guessing using recurrent neural networks - the missing manual. `https://www.password-guessing.org/blog/post/cupslab-neural-network-cracking-manual/`, 2017.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[19] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[20] Brad Harris. Passgan: Cracking passwords with generative adversarial networks - generative adversarial networks and cybersecurity: Part 2. `https://securityintelligence.com/generative-adversarial-networks-and-cybersecurity-part-2/`, 2018.

[21] HashCat. `https://hashcat.net`, 2017.

[22] Kelly Jackson Higgins. The coolest hacks of 2017. `https://www.darkreading.com/threat-intelligence/the-coolest-hacks-of-2017/d/d-id/1330699`, 2017.

[23] R Devon Hjelm, Athul P Jacob, Adam Trischler, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary seeking GANs. *International Conference on Learning Representations (ICLR)*, 2018.

[24] Quan Hoang, Tu D Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. *International Conference on Learning Representations (ICLR)*, 2018.

[25] Matthew Hutson. Artificial intelligence just made guessing your password a whole lot easier. `https://tinyurl.com/ybh2f617`, 2017.

[26] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] CUPS Lab. Fast, lean, and accurate: Modeling password guessability using neural networks (source code). `https://github.com/cupslab/neural_network_cracking`, 2016.

[29] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[30] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[31] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[32] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *IEEE Symposium on Security and Privacy (SP)*, pages 689–704. IEEE, 2014.

[33] Allee Manning. Researchers show how a.i. is the end of passwords as we know them. `https://www.inverse.com/article/36604-ai-cracking-passwords`, 2017.

[34] William Melicher, Blase Ur, Sean M Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate: Modeling password guessability using neural networks. In *USENIX Security Symposium*, pages 175–191, 2016.

[35] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[36] Michael Mimoso. Deep-learning passGAN tool improves password guessing. `https://threatpost.com/deep-learning-passgan-tool-improves-password-guessing/128039/`, 2017.

[37] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[38] Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. *International Conference on Learning Representations (ICLR)*, 2018.

[39] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[40] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev GAN. *International Conference on Learning Representations (ICLR)*, 2018.

[41] Youssef Mroueh and Tom Sercu. Fisher GAN. In *Advances in Neural Information Processing Systems*, pages 2513–2523, 2017.

[42] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching GAN. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2527–2535, 2017.

[43] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent GAN optimization is locally stable. In *Advances in Neural Information Processing Systems*, pages 5585–5595, 2017.

[44] Arvind Narayanan and Vitaly 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. ACM, 2005.

[45] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[46] Henning Petzka, Asja Fischer, and Denis Lukovnikov. On the regularization of wasserstein GANs. *International Conference on Learning Representations (ICLR)*, 2018.

[47] HashCat Per position Markov Chains. https://www.trustwave.com/Resources/SpiderLabs-Blog/Hashcat-Per-Position-Markov-Chains/, 2017.

[48] The Password Project. http://thepasswordproject.com/leaked_password_lists_and_dictionaries, 2017.

[49] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations*, 2016.

[50] RockYou. Rockyou. http://downloads.skullsecurity.org/passwords/rockyou.txt.bz2, 2010.

[51] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[52] HashCat Rules. https://github.com/hashcat/hashcat/tree/master/rules, 2017.

[53] John The Ripper KoreLogic Rules. http://contest-2010.korelogic.com/rules.html, 2017.

[54] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[55] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[56] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[57] John the Ripper. http://www.openwall.com/john/, 2017.

[58] John the Ripper Markov Generator. http://openwall.info/wiki/john/markov, 2017.

[59] Iain Thomson. AI slurps, learns millions of passwords to work out which ones you may use next. https://www.theregister.co.uk/2017/09/20/researchers_train_ai_bots_to_crack_passwords/, 2017.

[60] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5424–5433, 2017.

[61] Jai Vijayan. PassGAN: Password cracking using machine learning. https://www.darkreading.com/analytics/passgan-password-cracking-using-machine-learning/d/d-id/1329964, 2017.

[62] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of wasserstein GANs: A consistency term and its dual effect. *International Conference on Learning Representations (ICLR)*, 2018.

[63] Matt Weir. Probabilistic password cracker. https://sites.google.com/site/reusablesec/Home/password-cracking-tools/probablistic_cracker, 2009.

[64] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In *30th IEEE Symposium on Security and Privacy*, pages 391–405. IEEE, 2009.

[65] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016.

[66] Zhiming Zhou, Han Cai, Shu Rong, Yuxuan Song, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. Activation maximization generative adversarial nets. *International Conference on Learning Representations (ICLR)*, 2018.