

PassGAN: A Deep Learning Approach for Password Guessing 논문 리뷰

임세진

<https://youtu.be/kDNoRm76X2M>

Intro

- Password Guessing

: 대량의 후보 PW 사전을 만들어 이 사전을 기반으로 대입해서 PW를 알아내는 기법

- 유출된 PW Database를 통해 많은 user들이 추측하기 쉬운 암호를 사용 (ex. iLoVeyOu, p@ssw0rd)
- Password Guessing Tool은 이러한 암호들을 후보 암호로 사용해서 보다 효율적인 공격 시도
- 기존의 SOTA 암호 분석 도구 (HashCat, John the Ripper)에 규칙을 더 추가해서 PW를 생성하려고 하면 관련 전문 지식이 필요 (확장성 제한)

➔ GAN으로 학습 데이터(유출된 비밀번호)의 분포를 훈련시키면 PW의 구조 및 사전 지식 없이도 Password Guessing을 생성함

Purpose

- PassGAN 모델을 통해 수동 암호 분석이 아닌 GAN으로 실제 비밀번호(유출된)로부터 비밀번호의 분포를 자율적으로 학습해서 동일한 분포를 갖는 고품질 Password Guessing을 생성
- 신경망을 사용하는 것의 이점
 - PW가 생성되는 공간(범위)이 특정한 하위 집합으로 제한되지 않고 광범위하게 생성 가능

Related Work

<Tool>

- JTR (John The Ripper)

사전 공격(Dictionary Attack)을 통해 PW의 강도를 테스트하고 해시화된 PW를 Brute force 공격하여 크래킹

- HashCat

PW Crack Tool. PW를 추측하여 해시화한 후 공격하고자하는 PW의 해시와 비교하여 추측

(사전 공격, 규칙 기반 공격 등 다양한 무차별 대입 공격이 있음)

- RNN 기반 방식 (FLA) : 주 목적은 암호 강도 추정 || PassGAN은 Password Guessing 목적

Related Work

<Attack>

- Rule-based Attack (규칙 기반 공격)

모든 공격 방법 중 가장 복잡함. 단어의 수정, 잘라내기, 확장하기, 건너뛰기 등을 통해 공격하는 방식

- Markov 모델 (자연어 처리에 사용되는 알고리즘)

한 단어를 n-gram화하여 어떤 단어가 나올 확률을 조건부 확률로 예측하는 방법 (PW 전체에 n-gram 방식을 적용)

← '기억하기 쉬운 PW는 자연어의 특성과 비슷할 것'에서 출발

수동적으로 정의된 PW 규칙 (PW의 어느 부분이 문자와 숫자로 구성되는지) 사용 → PCFG (Probabilistic Context-Free Grammars)로 기술 개선

- PCFG (자연어 처리에 사용되는 알고리즘)

PW의 구조를 파악하여 각 자리에 특정 문자가 올 확률을 계산하는 방식

Contribution

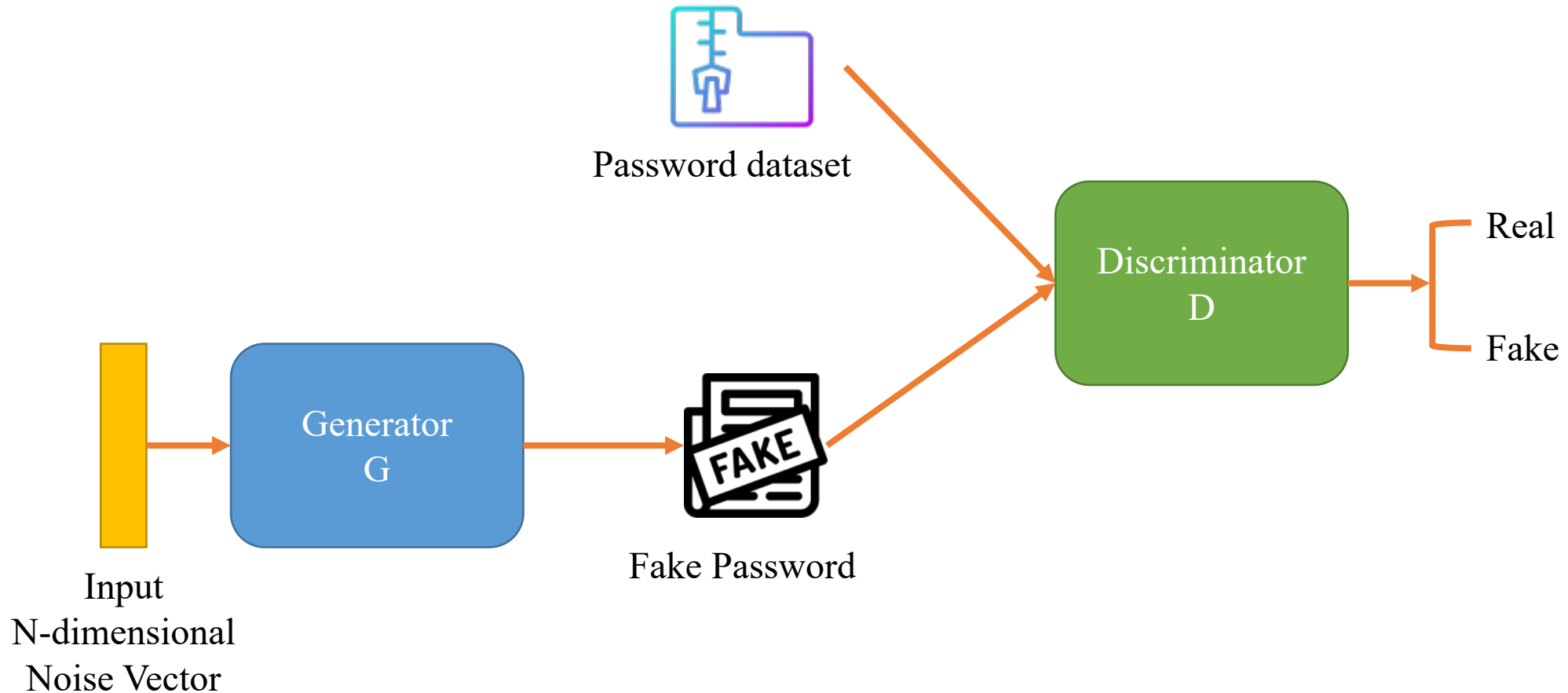
- 최초로 GAN을 사용해서 Password Guessing을 시도
- 유명 데이터셋으로 PassGAN을 평가
 - 규칙 기반 Tool, SOTA Tool을 능가함 (PW 구조에 대한 사전 지식 요구 X)
- Testing set과 일치하지 않는 PassGAN의 output의 대부분이 사람이 생성한 비밀번호처럼 보였음
 - Dataset에는 없었지만 실제 사용자 PW와 잠재적으로 일치할 수 있음
- HashCat 단독 사용 << PassGAN + HashCat 출력을 결합 (더 많은 PW 일치)

love42743	ilovey2b93	paolo9630	italyit
sadgross	usa2598	s13trumpy	trumpart3
ttybaby5	dark1106	vamperiosa	~dracula
saddracula	luvengland	albania.	bananabake
paleyoung	@crepess	emily1015	enemy20
goku476	coolarse18	iscoolin	serious003
nyc1234	thepotus12	greatrun	babybad528
santazone	apple8487	1loveyoung	bitchin706
toshibaod	tweet1997b	103tears	1holys01

Contribution

- 기존 Tool과 달리 PassGAN은 거의 무제한으로 Password Guessing을 출력할 수 있음
- PassGAN output \propto 일치하는 PW 수
 - ➔ PW가 생성되는 공간(범위)이 특정한 하위 집합으로 제한되지 않고 광범위하게 생성 가능
- 동일한 Dataset으로 관련 Tool과 성능 비교했을 때 PassGAN이 가장 많은 Password Guessing 달성
(다른 Tool보다 PW를 많이 생성했을 때)

모델 구조

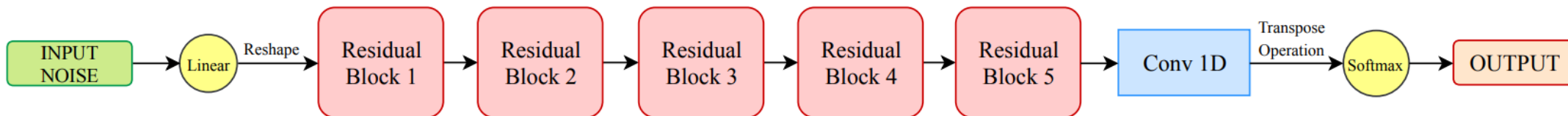


D : 진짜 PW와 G가 만든 PW를 완벽하게 구별해내는 것이 목적

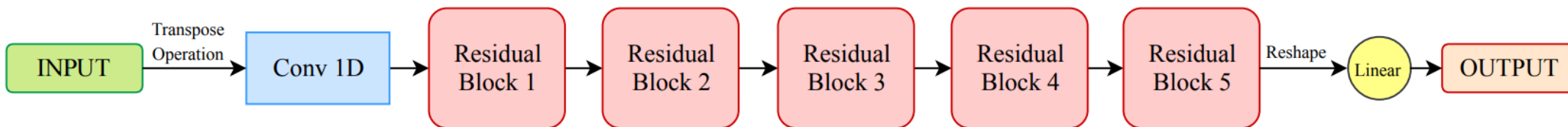
G : 그럴듯한 가짜 PW를 생성하여 D가 진짜인지 가짜인지 구별하지 못하게 하는 것이 목적

모델 구조

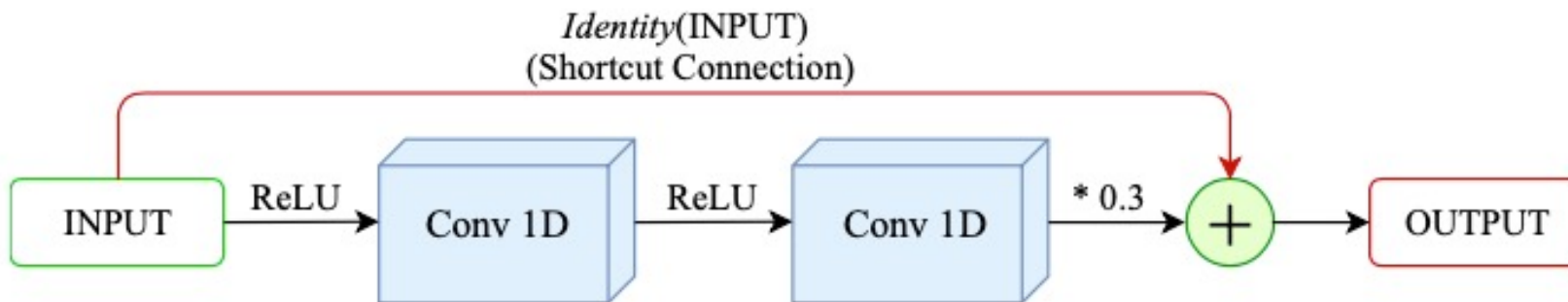
학습이 완료되면 G를 사용하여 Password Guessing 생성



(a) Generator Architecture, G



(b) Discriminator Architecture, D



<Residual Block>

Dataset

- 대규모 PW dataset

- ✓ Training : RockYou dataset의 일부

- ✓ Test : 2가지 수행

- (1) RockYou dataset (train이랑 test 데이터셋 중복 제거. 10자 이내로 필터링하여 사용)

- (2) LinkedIn (10자 이내로 필터링하여 사용)

|12345612345123456789passwordiloveyouprincess'
ngerslouisiorange789456999999shorty11111nathaz

<RockYou dataset 예시>

실험결과

(1) 43.6 % 일치 (1,350,178개 / 3,094,199개)

(2) 24.2 % 일치 (10,478,322개 / 43,354,871개)

----- train 데이터셋과 중복된 PW 제거 후 test -----

(1) 34.6 % 일치 (676,439개 / 1,978,367개)

(2) 34.2 % 일치 (8,878,284개 / 40,593,536개)

실험결과

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%)

PassGAN으로 5백억개의 PW 생성 → 겹치지 않는 PW가 70억개 → 34% 확률로 PW를 맞힘

실험결과

Approach	(1) Unique Passwords	(2) Matches	(3) Number of passwords required for PassGAN to outperform (2)	(4) PassGAN Matches
JTR Spyderlab	10^9	461,395 (23.32%)	$1.4 \cdot 10^9$	461,398 (23.32%)
Markov Model 3-gram	$4.9 \cdot 10^8$	532,961 (26.93%)	$2.47 \cdot 10^9$	532,962 (26.93%)
HashCat gen2	10^9	597,899 (30.22%)	$4.8 \cdot 10^9$	625,245 (31.60%)
HashCat Best64	$3.6 \cdot 10^8$	630,068 (31.84%)	$5.06 \cdot 10^9$	630,335 (31.86%)
PCFG	10^9	486,416 (24.59%)	$2.1 \cdot 10^9$	511,453 (25.85%)
FLA $p = 10^{-10}$	$7.4 \cdot 10^8$	652,585 (32.99%)	$6 \cdot 10^9$	653,978 (33.06%)

<다른 Password Guessing Tool과의 성능 비교 (RockYou testing set)>

같은 성능을 내기 위해 PassGAN은 다른 알고리즘보다 10배 이내의 PW 생성 요구

➔ But 추측을 위해 사전 지식 X ➔ PassGAN이 실제 PW의 분포를 잘 학습했다

실험결과

(a) RockYou Training Set

Password	Number of Occurrences in Training Set	Frequency in Training Set
123456	232,844	0.9833%
12345	63,135	0.2666%
123456789	61,531	0.2598%
password	47,507	0.2006%
iloveyou	40,037	0.1691%
princess	26,669	0.1126%
1234567	17,399	0.0735%
rockyou	16,765	0.0708%
12345678	16,536	0.0698%
abc123	13,243	0.0559%
nicole	12,992	0.0549%
daniel	12,337	0.0521%
babygirl	12,130	0.0512%
monkey	11,726	0.0495%
lovely	11,533	0.0487%
jessica	11,262	0.0476%
654321	11,181	0.0472%
michael	11,174	0.0472%
ashley	10,741	0.0454%
qwerty	10,730	0.0453%
iloveu	10,587	0.0447%
111111	10,529	0.0445%
000000	10,412	0.0440%
michelle	10,210	0.0431%
tigger	9,381	0.0396%
sunshine	9,252	0.0391%
chocolate	9,012	0.0381%
password1	8,916	0.0377%
soccer	8,752	0.0370%
anthony	8,564	0.0362%
friends	8,557	0.0361%
butterfly	8,427	0.0356%
angel	8,425	0.0356%
purple	8,381	0.0354%
jordan	8,123	0.0343%
liverpool	7,846	0.0331%
loveme	7,818	0.0330%
justin	7,769	0.0328%
fuckyou	7,702	0.0325%
football	7,559	0.0319%
123123	7,545	0.0319%
secret	7,458	0.0315%
andrea	7,395	0.0312%
carlos	7,281	0.0307%
jennifer	7,229	0.0305%
joshua	7,186	0.0303%
bubbles	7,031	0.0297%
1234567890	6,953	0.0294%
hannah	6,911	0.0292%
superman	6,855	0.0289%

(b) FLA

Password	Rank in Training Set	Frequency in Training Set	Probability assigned by FLA
123456	1	0.9833%	2.81E-3
12345	2	0.2666%	1.06E-3
123457	3,224	0.0016%	2.87E-4
1234566	5,769	0.0010%	1.85E-4
1234565	9,692	0.0006%	1.11E-4
1234567	7	0.0735%	1.00E-4
12345669	848,078	0.0000%	9.84E-5
123458	7,359	0.0008%	9.54E-5
12345679	7,818	0.0007%	9.07E-5
123459	8,155	0.0007%	7.33E-5
lover	457	0.0079%	6.73E-5
love	384	0.0089%	6.09E-5
223456	69,163	0.0001%	5.14E-5
22345	118,098	0.0001%	4.61E-5
1234564	293,340	0.0000%	3.81E-5
123454	23,725	0.0003%	3.56E-5
1234569	5,305	0.0010%	3.54E-5
lovin	39,712	0.0002%	3.21E-5
loven	57,862	0.0001%	3.09E-5
iloveyou	5	0.1691%	3.05E-5
1234568	6,083	0.0009%	2.99E-5
223455	1,699,287	0.0000%	2.68E-5
12345668	1,741,520	0.0000%	2.64E-5
1234561	12,143	0.0005%	2.59E-5
123455	5,402	0.0010%	2.58E-5
ilover	45,951	0.0002%	2.52E-5
love15	2,074	0.0025%	2.39E-5
12345660	—	—	2.39E-5
1234560	3,477	0.0015%	2.34E-5
123456789	3	0.2598%	2.27E-5
love11	1,735	0.0029%	2.21E-5
12345667	252,961	0.0000%	2.11E-5
12345678	9	0.0698%	2.11E-5
223457	8,929,184	0.0000%	2.09E-5
love12	565	0.0067%	2.05E-5
lovo	—	—	2.03E-5
12345666	46,540	0.0002%	1.98E-5
123456689	—	—	1.97E-5
1234562	92,917	0.0001%	1.92E-5
12345699	197,906	0.0000%	1.90E-5
123451	9,950	0.0006%	1.89E-5
123450	7,186	0.0008%	1.88E-5
loves	779	0.0054%	1.83E-5
1234576	61,296	0.0001%	1.80E-5
love13	1,251	0.0038%	1.78E-5
lovele	154,468	0.0001%	1.78E-5
lovine	4,497,922	0.0000%	1.75E-5
lovi	4,498,263	0.0000%	1.74E-5
iloven	323,339	0.0000%	1.59E-5
lovina	62,446	0.0001%	1.53E-5

(c) PassGAN

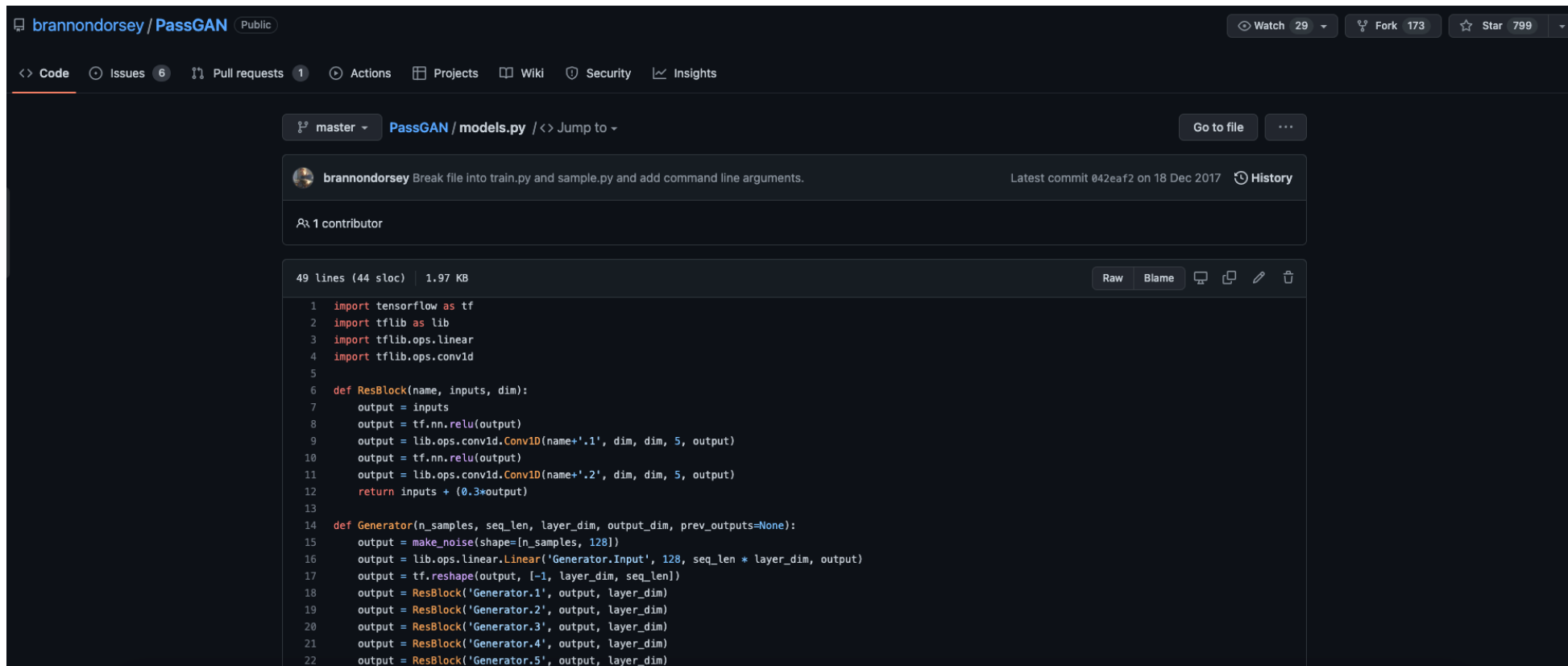
Password	Rank in Training Set	Frequency in Training Set	Frequency in PassGAN's Output
123456	1	0.9833%	1.0096%
123456789	3	0.25985%	0.222%
12345	2	0.26662%	0.2162%
iloveyou	5	0.16908%	0.1006%
1234567	7	0.07348%	0.0755%
angel	33	0.03558%	0.0638%
12345678	9	0.06983%	0.0508%
iloveu	21	0.04471%	0.0485%
angela	109	0.01921%	0.0338%
daniel	12	0.0521%	0.033%
sweetie	90	0.02171%	0.0257%
angels	57	0.02787%	0.0245%
maria	210	0.01342%	0.0159%
loveyou	52	0.0287%	0.0154%
andrew	55	0.02815%	0.0131%
123256	301,429	0.00003%	0.013%
iluv!u	—	—	0.0127%
dangel	38,800	0.00018%	0.0123%
micheel	1,442	0.00335%	0.0119%
marie	483	0.00755%	0.0118%
andres	223	0.01274%	0.0106%
lovely	15	0.0487%	0.0103%
123458	7,352	0.00076%	0.0099%
sweet	329	0.00999%	0.0097%
prince	243	0.01217%	0.0092%
ilove	2,177	0.00234%	0.0089%
hello	61	0.02648%	0.0086%
angell	184	0.01459%	0.0085%
iluveu	58,131	0.00013%	0.0083%
723456	337,321	0.00003%	0.0082%
loveu	852	0.00505%	0.0082%
lovers	70	0.0253%	0.0082%
iluv!you	—	—	0.0082%
bella	732	0.00562%	0.0081%
andrea	43	0.03123%	0.0081%
iluveyou	183,386	0.00004%	0.0079%
kella	180,219	0.00004%	0.0076%
michelle	24	0.04312%	0.0074%
mariana	228	0.01265%	0.0074%
marian	681	0.00593%	0.0073%
daniela	95	0.02064%	0.0072%
dancer	122	0.01799%	0.0072%
lovery	46,470	0.00016%	0.0071%
dancel	42,692	0.00017%	0.007%
23456	3,976	0.00134%	0.007%
lg3456	—	—	0.007%
loveme	37	0.03302%	0.007%
jessie	213	0.01329%	0.0069%
buster	145	0.01619%	0.0068%
anger	172,425	0.00005%	0.0067%

FLA보다 Training set과 더 유사

앞으로 진행 사항

논문에서 모델 동작방식에 대한 자세한 설명이 부족

1. Github에 있는 PassGAN 코드 돌려보면서 Input, output, 동작 방식 등 상세히 살펴보며 이해
2. GAN 모델 pytorch로 구현



The screenshot shows the GitHub repository for `brannondorsey / PassGAN`. The file `models.py` is open, showing the following code:

```
1 import tensorflow as tf
2 import tflib as lib
3 import tflib.ops.linear
4 import tflib.ops.conv1d
5
6 def ResBlock(name, inputs, dim):
7     output = inputs
8     output = tf.nn.relu(output)
9     output = lib.ops.conv1d.Conv1D(name+'.1', dim, dim, 5, output)
10    output = tf.nn.relu(output)
11    output = lib.ops.conv1d.Conv1D(name+'.2', dim, dim, 5, output)
12    return inputs + (0.3*output)
13
14 def Generator(n_samples, seq_len, layer_dim, output_dim, prev_outputs=None):
15     output = make_noise(shape=[n_samples, 128])
16     output = lib.ops.linear.Linear('Generator.Input', 128, seq_len * layer_dim, output)
17     output = tf.reshape(output, [-1, layer_dim, seq_len])
18     output = ResBlock('Generator.1', output, layer_dim)
19     output = ResBlock('Generator.2', output, layer_dim)
20     output = ResBlock('Generator.3', output, layer_dim)
21     output = ResBlock('Generator.4', output, layer_dim)
22     output = ResBlock('Generator.5', output, layer_dim)
```

감사합니다