Project Proposal Team 7

Aims: 2/3
Nets: 2/2
Data: 4/4
Testing: 3/4

Spencer Arnold
Ryan Hines
Matthew Hawks
Jacob Anderson
Tae Kweon
Ali Najib

Grade: 11/13

Pretty good overall, but you need to flesh out the methods of investigation (potential architectures) and also provide some justification for choice of n (length) or explore length-dependence a little more broadly to convince everyone you aren't just taking it for granted that you have the right choice of n

Code Def

Acronyms	Definition
PRN	Pseudo random number
PRNG	Pseudo random number generator

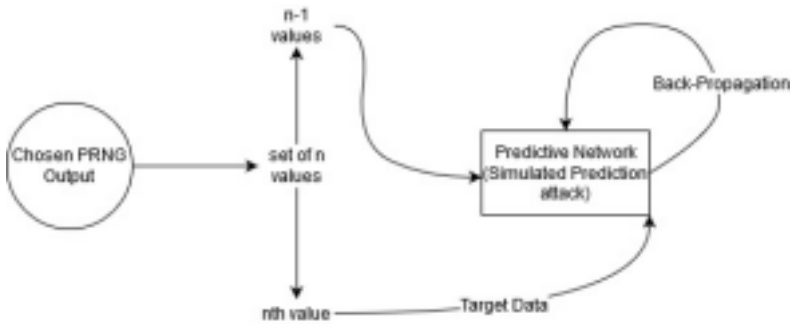
Our Aim

- To train a neural net to predict PRNs from a chosen PRNGs output. We aim to train new neural networks on additional methods of PRN generation, if we have enough time. -> well you need some benchmark...
- Our secondary aim is to continue the quest [1] (https://arxiv.org/ftp/arxiv/papers/1801/1801.01117.pdf)[2] (https://arxiv.org/pdf/1810.00378.pdf) for insights on pseudo randomness.

The Meat

Our proposal is to create predictive networks that are fed outputs of PRNGs, which will attempt to predict an nth value, based on previous n-1 values in a particular set of input data. After being trained on, ideally thousands of sets, the predictive network will form a better stochastic "understanding" of how the PRNG works underneath, thus being able to more accurately predict numbers generated from that PRNG in the future. The backpropagation will flow through the predictive network to acheive the adversarial nature of a traditional GAN setup, but in reality, we won't be using a generative net, just a PRNG algorithm, so the generative part of our setup won't be defensive. There is an eloquent description of what our predictor network will do below. -> what kinds of results are expected (reasonable accuracy?) (-1)

> The predictor network maximizes [gradient ascent] the probability of correctly predicting the nth value from the other [n-1] values.ref (https://arxiv.org/pdf/1810.00378.pdf)



We hope to do this process above with several PRNGs, starting with one of the oldest in 1946 and gradually making our way to newer methods of PRN generation.

> Middle-square method (1946) (https://en.wikipedia.org/wiki/Middle-square_method)
> Lehmer generator (1951) (https://en.wikipedia.org/wiki/Lehmer_random_number_generator)
> Linear congruential generator (1958) (https://en.wikipedia.org/wiki/Linear_congruential_generator)
> Lagged Fibonacci (1965) (https://en.wikipedia.org/wiki/Lagged_Fibonacci_generator)
> Wichmann-Hill generator (1982) (https://en.wikipedia.org/wiki/Wichmann%E2%80%93Hill)
> Park-Miller generator (1988) (https://en.wikipedia.org/wiki/Lehmer_random_number_generator)
> Maximally periodic reciprocals (1992) (https://en.wikipedia.org/wiki/Sophie_Germain_prime)
> Mersenne Twister (1998) (https://en.wikipedia.org/wiki/Mersenne_Twister)

data ok

Theoretically, we will be able to look at the data and stochastically rank each PRNG on how easy a prediction attack worked on it. This should also tell us how "good" each PRNG is at generating "random" numbers.

Our hypothesis comes in two parts:

1) We predict there will be a positive trend over time on the cryptographic strength of each subsequent PRNG, given the nature of the increasing importance of stronger PRNGs.

2) We also predict that as we get into cryptographically stronger generation methods, our prediction success rates (even with learning) will be less effective. oh, I like these: good mesh of qualitative/quantitative components

Practical Application

A GAN method similar to this could be used in the future during a real-time prediction attack scenario. If attackers are somehow able to break a strong PRNG that is used to generate numbers for cryptographic methods, a method like this could be used as a defense mechanism to prevent (or slow down) further PRN generation cycles from being compromised.

Of course, determining any correlation in pseudo random sequences would be a great help to the community to write stronger generators

You need to have a plan concerning what network architectures you will be employing, how many experiments, different sequence lengths, other potential hypotheses that might come to light (length-dependence of learning, prediction, generalization, etc.) and which architectures do a better/worse job of this. As a result, I can see where a NNet can fit in but you need more details on how they actually fit into the plans for the project (-1)