Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Machine Learning-Driven Random Number Generation: A Comparative Study of WGAN-GP and RNNs for Cryptographic Security

Rana Saeed Hamdi [1], Saif Al-alak [2], Elaf Ali Abbood [3]
[1,2,3] *Department of Computer Science-College of Science for Women – University of Babylon*
* *Corresponding author's Email: scw564.rana.saeed@student.uobabylon.edu.iq*

ARTICLE INFO

Received: 15 Mar 2025

Revised: 7 Apr 2025

Accepted: 18 Apr 2025

ABSTRACT

Numerous applications require a high level of randomness, making random number generation an essential component of modern cryptographic systems. Using recurrent neural networks (RNNs) and Generative Adversarial Networks by Wasserstein with Gradient Penalty (WGAN-GP), this paper explores high-quality random number generation using machine learning methods with different complexities. The random sequences produced by the models under consideration could be used in secure cryptographic applications. In addition, while an RNN model captures temporal dependencies to convey complex sequences, a WGAN-GP utilizes the Earth-Mover's distance to improve its training stability and random number quality. The random numbers produced are exhaustively tested for their level of randomness using the NIST and Diehard test suites. In major statistical tests, higher p-values indicate that WGAN-GP performs better than RNNs in the context of randomness quality. This work applies the groundwork for further studies on secure cryptographically random number generators (RNGs) and shows how machine learning could enhance the effectiveness of RNGs.

Keywords: machine learning, GANs, random number generating, RNNs, statistical Randomness testing, p-value.

1. INTRODUCTION

A random number generator (RNG) is crucial for many modern-day applications, e.g., simulation, cryptography, gaming, and scientific research. It provides security by using high-quality, unpredictable generators to generate keys and codes for encrypting and decrypting data and messages [1] [2] [3]. The quality of the RNG is critical since the security of the cryptographic system could be compromised by using weak or predictable random numbers [4].

Two popular approaches to generating random numbers are pseudo-random number generators (PRNGs) and true random number generators (TRNGs). Whereas PRNGs generate randomness using software methods, TRNGs depend on physical phenomenon [5]. With speeds ranging from slow rates to random bit generation (RBG) at 300 Gb/s, researchers developed well-known TRNG implementations. [6] [7], as much as two Tb/s RBG [8] [9], in addition to the 250 Tb/s demonstrated rate of RBG, which was achieved by Kim et al. [10]. True random number generators are widely used in high-risk fields such as security and finance, where true unpredictability is crucial. However, compared to PRNGs, TRNGs are slower in generating a large quantity of random numbers and require specialized hardware. PRNGs, adopted for decades as general-purpose software modules, are widely used in fields like security, where information systems require massive quantities of random numbers. PRNGs effectively generate uncorrelated number sequences, which develop highly complex and random for practical purposes.

The National Institute of Standards and Technology (NIST) and diehard tests [11], validate PRNGs and evaluate the resilience and accuracy of the generated numbers. Although PRNG algorithms are common in online applications, their implementation code and seeds must be carefully hidden to prevent malicious prediction of subsequent numbers. Modern smartphone applications complicate full concealment of implementation details, even with encryption and code obfuscation, necessitating new RNG models [5].

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Machine learning technologies and deep neural networks (DNNs), which model complex mathematical relationships, are increasingly used to develop artificial intelligence [12]. The key challenge lies in creating new PRNGs with end to end learning, where machine learning tools directly mimic pre-existing PRNGs without preprocessing or post-processing [13]. With end-to-end deep learning, large neural networks could replace complex systems, enabling developers to build customized PRNGs. Over the past decade, researchers have explored deep learning models like generative adversarial networks (GANs)[14] and recurrent neural networks (RNNs)[15] for PRNGs.

This paper contributes to developing two models, Wasserstein GANs with gradient penalty (WGAN-GP) and Recurrent Neural Networks (RNNs), for generating cryptographically strong random numbers. We employ WGAN-GP to stabilize training via informative gradients and RNNs to create a sequential prediction model based on temporal dependencies. To establish how efficiently these models generate high-quality random sequences, we apply well-known sets of tests for randomness (NIST, Diehard).

2. RELATED WORK

Pseudo-random number generators (PRNGs) are important in generating random numbers and are used in watermarking, hashing, and cryptography. There are various techniques available for generating pseudo-random numbers.

Crespo et al. employed neural networks in 2024. For the first time on an original task estimation quality and security of the random number generators, both pseudo (PRNG) and natural quantum (QRNG). It was shown how well CNN and LSTM networks can distinguish ordinary PRNGs from the highest quality and security from the GSRNGs in the article, with an error of cross-entropy of 0.52 [16].

Proskurin et al. in 2024 hybrid neural network design was developed to increase the accuracy of PRNG detection by more than 95%. The design combines a working convolutional neural network and a recurrent neural network. The given method allows you to secure and ensure the reliability of the random number generator. [17].

Proskurin et al. (2023) produced a hybrid deep-learning model. May include Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs). It demonstrates that the model accurately recognizes the data patterns, outperforming the traditional methods in predicting sequences from QRNGs and PRNGs [18].

Based on the combination of chaotic systems and Generative adversarial networks, Ji et al. (2022) have proposed a method for the generation of pseudo random sequences. The GAN model has shown a high degree of unpredictability and disorder after the process of iterative training on the indicated parameters of the randomization, which makes it suitable for encryption systems [20].

Park et al (2022) aimed to advance the quality of random number generators. This was accomplished by developing a hybrid PRNG model. They used convolutional neural networks and reinforcement learning algorithms. This model was able to generate a more random sequence, with the highest correlation coefficient equal to 19% [21].

Alloun et al. (2022) designed PRNGs using the Lorenz chaotic system and artificial neural networks. It has been shown in NIST SP 800-22 tests that the generated sequence has good statistical randomness, the p-value is much higher than 0.01, and the key space and dataset generation are the best choices [22]

Pasqualini and Parton (2020) compared two RNG models: Baseline Formulation (BF) and Reinforcement Formulation (RF). The outcomes demonstrated that RF surpassed BF in producing random sequences with more significant average rewards and verifying its prowess for an intelligent compound [15].

However, Hussein and Al-alak (2021) suggested reducing the complexity of the random number generator by using lightweight algorithms in an attempt to create secret keys [23].

Mohammed and Al-alak (2018) proposed a hybrid scheme that employs an asymmetric algorithm to generate a secret key for encrypting data transmitted in a Wireless Sensor Network (WSN)[24].

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Some researchers provide random sequence numbers based on complex methods that utilize many mathematical equations. Al-alak et al. (2013) developed a protocol to improve the randomness of the Advanced Encryption Standard (AES) by generating a random sequence of secret keys [25].

Desai et al. (2012) generated pseudo-random sequences using weight matrices from a layer recurrent neural network (LRNN). These sequences passed 11 of 16 NIST statistical tests, with performance varying based on the training function used [26].

3. METHODOLOGY

3.1 Random Number Generators

- **WGAN-GP:**

GANs are a class of unsupervised techniques that generate samples by directly learning the probability distribution of data before reaching inferences about the probability distribution related to training data. Generative and discriminative networks compose the majority of GANs [14]. Using noise vectors as input, the generative network generates samples similar to real samples. The primary purpose of the discriminative network is to assess whether the input samples were generated or real.

Many GAN variants have appeared in recent years, including Wasserstein GANs (WGANs) [27], Least Squares GANs (LSGANs) [28], and Deep Convolutional GANs (DCGANs) [29]. They try to produce high-quality samples and stable training for GANs. Training the GAN is difficult due to mode collapse, vanishing gradient, and learning instability [30]. The Wasserstein GAN (WGAN) was developed as an alternative to the traditional GAN to avoid these issues.

Essentially, it demonstrates that the optimization problem is appropriate and minimizes a suitable and efficient approximation of the Earth Mover (EM) distance. The Wasserstein distance, sometimes referred to as the earth mover's distance, is used to determine the WGAN's loss function [31] [5].

In the context of a Wasserstein GAN, equations (1) and (2) expressed the dual form of Wasserstein distance.

$$- V(D, G) = W(P_r, P_g) \quad (1)$$

$$- V(D, G) = \max \{f \in \mathcal{F}\} E\_\{x \sim P_r\} [D(x; f)] - E\{z \sim P_z\} [D(G(z; \theta); f)] \quad (2)$$

Where $W(P_r, P_g)$ is the distance's Wasserstein for the distribution of data between the generated P_g and the real P_r, $\mathcal{F}$ is a set for all 1-Lipschitz functions, $D(x; f)$ is the critic function parameterized by f, $G(z; \theta)$ is the generator function which parameterized by θ, P_r is the distribution of real data, P_g is the distribution of generated data (defined essentially by $G(z; \theta)$), and P_z is a prior distribution (e.g., Gaussian or uniform noise).

WGAN uses the Gradient Penalty (GP) to overcome the drawbacks of weight clipping in traditional WGAN. As an alternative to weight clipping, a gradient Penalty (GP) applies a Lipschitz constraint on the critic. Arjovsky et al. presented WGAN-GP for the first time in 2017 [32] [27]. It resolves mode collapse, improves stability, and optimizes training hyperparameters.

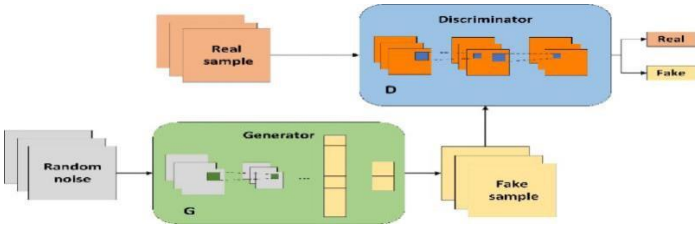

Figure 1: The architecture of the WGAN-GP model.

A generator G and a discriminator D are the two main components of WGAN-GP, as illustrated in Figure 1. By imitating the complex distribution relationship shown in real samples, the G can transform random noise into data that corresponds to amino acid sequences. Still, the data it generates is fake. To determine whether the data originates

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

from the generator or a real sample, the D could be trained to distinguish between fake and real data. The performance of the network can be continuously optimized through the collaborative game between G and D. Using equations (3) and (4), we can determine the WGAN-Objective GP's function, where (3) is expressed as critic loss function and (4) is expressed as generator loss function [33].

- Critic Loss Function:

L-critic = E - {x ~ P-r} [D - ϕ (x)] - E- {z ~P-z} [D-ϕ (G-θ (z))] + λ * E- {\hat{x} ~ P- {\ hat{x}}} [(||∇_ {\ hat{x}} D-ϕ (\ hat{x}) || _ 2 - 1) ^ 2] (3)

- Generator Loss Function:

L-generator = E- {z ~P-z} [D- ϕ (G-θ (z))] (4)

where L-generator is the loss function for the generator, L-critic is the loss function for the critic (or discriminator), and D-ϕ(x) is the generator function is parameterized by θ, the critic function is parameterized by ϕ, the gradient of the critic for input samples is ∇_{\hat{x}} D-ϕ(\hat{x}), their distribution of real data is P-r, their prior distribution (such as Gaussian or uniform noise) is P-z, and λ is the gradient penalty coefficient.

- **RNN (The Recurrent Neural Network)**

Recurrent neural networks are another form of artificial neural network where connections are made sequentially between nodes. Nodes in a variant layer will only be able to communicate with one other in one direction. The data flow in recurrent neural networks is unidirectional; hence, the network architecture comprises three layers (input, hidden, and output). Due to the sequential transfer of data from one node to another, outputs from previous nodes can be utilized as inputs for subsequent nodes. Consequently, the model can dynamically construct the past by assembling and transmitting data from previous inputs to subsequent nodes[34] [35]. Classical neural networks perform well when input and output are completely irrelevant to one another; however, that is not always the case [36]. Figure 2 illustrates a diagram for recurrent neural network. All prior data is compiled and stored in the network's hidden section, which is calculated in equation (5). A new input xt is introduced to the network at each time step t. [37].

- $h_t = f(W_x x_t + W_h h_t - 1)$ **(5)**

In conclusion, equation (6) generates the output yt at each node sequentially.

- $y_t = g(W_y)$ **(6)**

Where the selected activated function is represented by g(Wy).

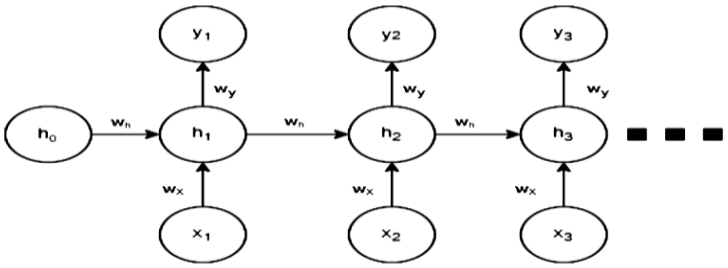

Figure 2: A typical Recurrent Neural Network.

Recurrent neural networks, also known as RNNs, are made up of three layers of the neurons that are connected via synapses. Each neuron's input, activation, and output functions can be used to define it. Weight is a value assigned to each synapse between two neurons [38]. The final result is calculated by doing the computation layer by layer. At first, the input values are fed into the input neurons. In the input layer, the output function is computed by every neuron. The other layer receives the computed values. Arriving at the output layer marks the end of the operation [37].

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.2 Training Processes:

- **Proposed Model WGAN-GP.**

Using a WGAN with a gradient penalty, the structure of the randomly generated numbers from Mersenne Twister was predicted through a machine learning model [38]. The WGAN was trained on the Mersenne Twister random numbers and subsequently used as a model to generate new random numbers. Figure 3 illustrates a proposed WGAN's architecture. A training process starts with input data and hyperparameters that define the architecture and training settings. The generator is a neural network that creates fake data from a latent vector. The generator architecture consists of dense layers (1024, 512, and latent dimension), batch Normalization (to stabilize training), LeakyReLU activations (to allow small gradients for negative inputs), and dropout (to prevent overfitting). The discriminator (also called the critic in WGAN) distinguishes real data from fake data. Its architecture consists of dense layers (1024, 512, and 1 output neuron), LeakyReLU activation, and dropout layers (to prevent overfitting)

In the process of training, a generator is provided to imitate the discriminator, while the discriminator itself has been trained to make distinctions between fake and real data. Finally, highly efficient generator and discriminator models are saved for future use.

- **Proposed Model RNNs.**

In RNN, the sequential RNN model was defined through the first Simple RNN Layer, which includes 256 units and ReLU activation that was introduced in the Elman Neural Network [39]. During training, it randomly adjusts 30% of the neurons to zero, preventing overfitting, and it returns sequences, meaning that the output will be passed to the next RNN layer. Second are the simple RNN layer (128 units with ReLU activation) and the dropout layer (0.3 probability). The dense output layer ensures that the output values are between 0 and 1 by utilizing tanh activation.

Essentially, the model generates sequences using random noise. After being rescaled to [0,255], the tanh activation produces values in [-1,1]. A flattened 1D array is returned as the representation of the generated key.

The iterative training of the model generates both fake and real data. The batch size is adjusted every 5000 iterations to improve performance and training stability over time. The final result is a 256-dimensional key, which is useful in security applications.

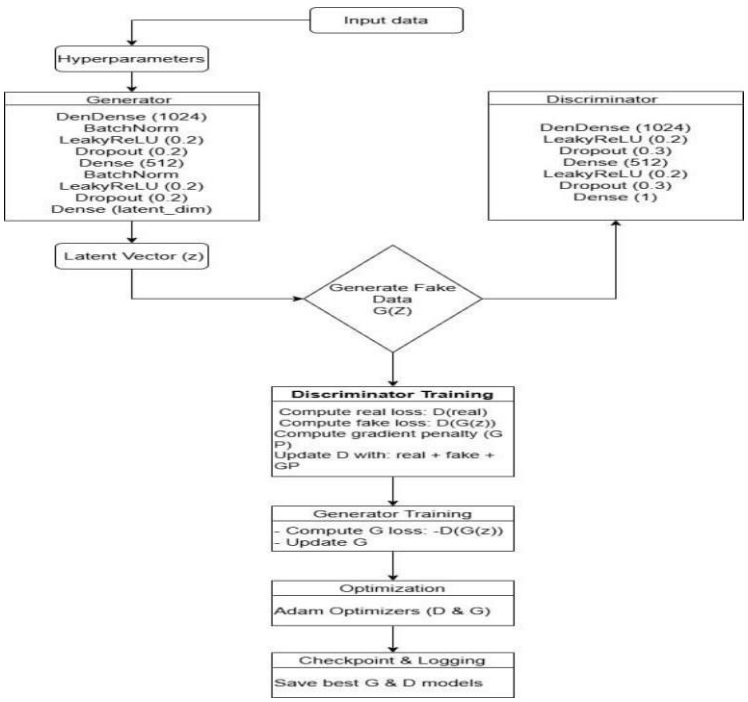

Figure 3: Proposed WGAN-GP Architecture.

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. RANDOMNESS TESTING

To assess the randomness of the numbers generated by the machine learning model, we used the Diehard and NIST random number test suites. To determine if a set of data can be classified as random, randomness tests are used to evaluate the data and analyze its distribution.

4.1 NIST Test Suite:

NIST tests consist of 15 tests (NIST). We choose the following tests: [5] [11]

- Frequency Test: Checks if an array's 1s and 0s are approximately equal in a random array, as expected.
- Block Frequency Test: An expansion of frequency testing. Also, that determines if a 1/2 probability of 1 will occur.
- Runs Test: concerned with the length of sequences, which are described as continuous sequences of identical bits, and the number of runs in each.
- Longest Run Test: Evaluate an M-bit block's longest run of 1s. This test aims to see if the evaluated sequence's longest sequence of 1s follows the expected pattern of a randomly generated sequence. Its goal is to determine whether a random sequence's predicted number of 1s and 0s of varying lengths is satisfied.

4.2 Diehard Tests:

The Diehard test is used to measure randomness. The results of diehard tests are p-values, which belong between 0 and 1. The Diehard tests are (birthday spacing, overlapping permutations, ranks of matrices, monkey tests, count the ones, parking lot test, minimum distance test, random spheres test, the squeeze test, overlapping sums test, runs test, the craps test). These p-values are divided into three kinds of areas, as follows: [24]

- **The Failure Areas**: $0 < \text{p-value} \leq 0.1$ or $0.9 \leq \text{p-value} \leq 1$.
- **The Doubt Areas**: $0.1 < \text{p-value} \leq 0.25$ or $0.75 \leq \text{p-value} < 0.9$.
- **The Safe Areas:** $0.25 < \text{p-value} < 0.75$.

5. RESULTS:

Two training techniques are included in this study: WGAN-GP and RNN. The training's objective is to generate random numbers. The results of the first WGAN-GP procedure utilizing NIST and Diehard tests to assess the generated numbers' randomness are illustrated in Tables 1 and 2, separately. Figure 1 demonstrates the randomness of the generated numbers.

Table 1: p-values of NIST statistical tests for the generated numbers by WGAN-GP.

Statistical test	p-values
Entropy	7.179382031114783
Frequency	0.250535974413638
Block Frequency	7.465128597755619e-36
Run Test p-value	0.13660789969140616
Longest Run of Ones	8.03200116771825e-11
Binary Matrix Rank	0.3524229074889868

Table 2: p-values of the diehard tests for the generated numbers by WGAN-GP.

Test Name	Fail	Doubt	Safe
BIRTHDAY SPACINGS	6	1	2
Overlapping permutations	1	1	0

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Binary rank	11	7	7
Monkey tests	20	0	0
OPSO	12	4	7
OQSO	10	5	13
DNA	10	9	10
Count the ones	2	0	0
Count the ones in specified bytes	14	7	6
The Parking lot	2	8	0
The Minimum Distance	0	0	1
Random spheres	5	4	11
The squeeze	1	0	0
Overlapping sums	9	1	0
Runs	1	3	0
Craps	1	0	1

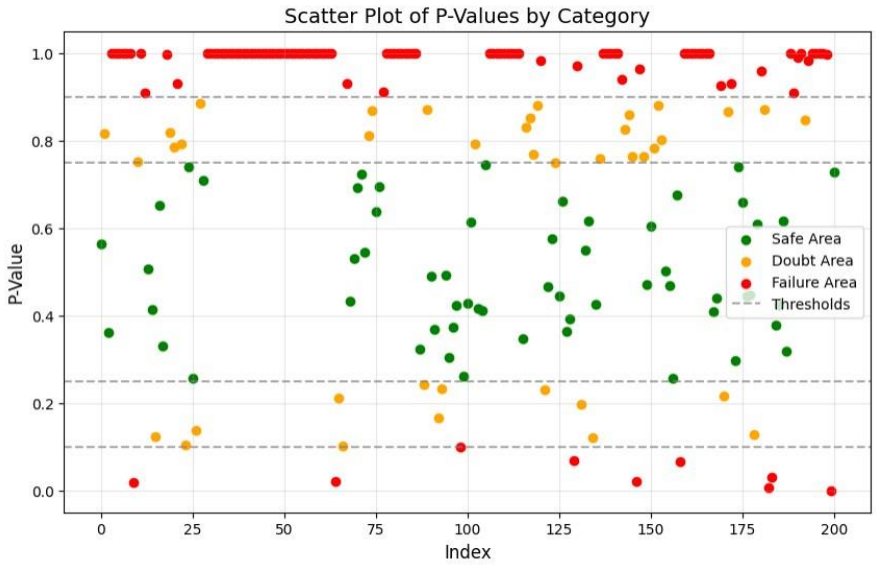

Figure 3: A randomness of the WGAN-GP generated numbers.

The results of a second procedure, RNN with NIST testing, are shown in Table 3, and Table 4 demonstrates Diehard tests. The randomness of the RNN-generated numbers is shown in Figure 2.

Table 3: NIST statistical tests of p-values for the RNN-generated numbers.

Statistical test	p-values
entropy	7.043168103758233
Frequency of ones	0.48291015625
Block Frequency	5.3142670560326 47e-34
Run Test p-value	0.0008774707000943856
Longest Run of Ones	3.797912493177547e-08
Binary Matrix Rank	0.33480176211453744

Journal of Information Systems Engineering and Management
2025, 10(36s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 4: p-values of the diehard tests for the generated numbers by RNN.

Test Name	Fail	Doubt	Safe
BIRTHDAY SPACINGS	9	0	0
Overlapping permutations	2	0	0
Binary rank	24	0	1
Monkey tests	20	0	0
OPSO	23	0	0
OQSO	26	0	2
DNA	25	3	1
Count the ones	2	0	0
Count the ones in specified bytes	27	0	0
The Parking lot	10	0	0
The Minimum Distance	1	0	0
Random spheres	20	0	0
The squeeze	1	0	0
Overlapping sums	10	0	0
Runs	4	0	0
Craps	2	0	0

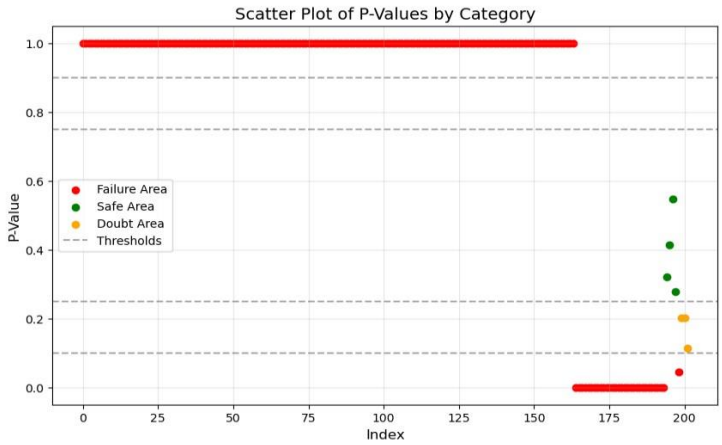

Figure 4: The randomness of the generated numbers by RNN.

The randomness of a generated numbers by RNNs and WGAN-GP is evaluated with each test and produces a p-value (statistical measure of randomness). Sequences are considered random for NIST tests if their p-values exceed the significance level of 0.01. Insufficient randomness in the generator is shown by failure in several tests. P-values in the fail region (below the threshold) reflect non-randomization in Diehard tests, but those in the pass region (for example, above the crucial threshold) show stronger evidence of randomness.

The results showed that WGAN-GP generated higher random sequences than RNNs, which was demonstrated by higher p-values in both test suites.

The Conclusions with Future Work:

This study investigated the generation of high-quality random numbers using advanced machine learning models, specifically WGAN-GP and RNNs. The primary objective was to generate models capable of generating random number sequences suitable for secure cryptographic applications. When compared to RNNs, WGAN-GP generated more unpredictable and high-quality sequences, according to an extensive evaluation that used the NIST and Diehard test suites. Through improved training stability and a greater number of sequences passing statistical randomness tests, the WGAN-GP model is a massive candidate for application in cryptography.

The results demonstrate the possibilities of machine learning, especially WGAN-GP, to improve random number generation. WGAN-GP improved GAN performance and resolved long-standing problems, including mode collapse and training instability by using the distance and gradient penalty of the Earth-Mover. WGAN-GP generated strong success in generating robust random sequences since RNNs showed their ability to predict temporal dependencies. This work opens the way for more investigation on random number generators produced by machine learning with cryptographic security. Future research may focus on improving model efficiency, optimizing designs for specific cryptography applications, and investigating other frameworks of deep learning to enhance the quality of randomness. Including machine learning in random number generation would help to ensure the dependability and safety of cryptographic systems, given the growing demand for secure encryption.

REFERENCES:

[1] L. Pasqualini and M. Parton, "Pseudo Random Number Generation through Reinforcement Learning and Recurrent Neural Networks," Nov. 19, 2020, *arXiv*: arXiv:2011.02909. doi: 10.48550/arXiv.2011.02909.
[2] E. Simion, "Entropy and Randomness: From Analogic to Quantum World," *IEEE Access*, vol. 8, pp. 74553–74561, 2020, doi: 10.1109/ACCESS.2020.2988658.
[3] Y. Lin, F. Wang, and B. Liu, "Random number generators for large-scale parallel Monte Carlo simulations on FPGA," *J. Comput. Phys.*, vol. 360, pp. 93–103, May 2018, doi: 10.1016/j.jcp.2018.01.029.
[4] S. Ergun, "Security analysis of a chaos-based random number generator for applications in cryptography," in *2015 15th International Symposium on Communications and Information Technologies (ISCIT)*, Nara: IEEE, Oct. 2015, pp. 319–322. doi: 10.1109/ISCIT.2015.7458371.
[5] K. Okada, K. Endo, K. Yasuoka, and S. Kurabayashi, "Learned Pseudo-Random Number Generator: WGAN-GP for Generating Statistically Robust Random Numbers," Jun. 27, 2022, *In Review*. doi: 10.21203/rs.3.rs-1695121/v1.
[6] A. Uchida *et al.*, "Fast physical random bit generation with chaotic semiconductor lasers," *Nat. Photonics*, vol. 2, no. 12, pp. 728–732, Dec. 2008, doi: 10.1038/nphoton.2008.227.
[7] I. Kanter, Y. Aviad, I. Reidler, E. Cohen, and M. Rosenbluh, "An optical ultrafast random bit generator," *Nat. Photonics*, vol. 4, no. 1, pp. 58–61, Jan. 2010, doi: 10.1038/nphoton.2009.235.
[8] A. Argyris, S. Deligiannidis, E. Pikasis, A. Bogris, and D. Syvridis, "Implementation of 140 Gb/s true random bit generator based on a chaotic photonic integrated circuit," *Opt. Express*, vol. 18, no. 18, p. 18763, Aug. 2010, doi: 10.1364/OE.18.018763.
[9] S. Xiang, B. Wang, Y. Wang, Y. Han, A. Wen, and Y. Hao, "2.24-Tb/s Physical Random Bit Generation With Minimal Post-Processing Based on Chaotic Semiconductor Lasers Network," *J. Light. Technol.*, vol. 37, no. 16, pp. 3987–3993, Aug. 2019, doi: 10.1109/JLT.2019.2920476.
[10] K. Kim *et al.*, "Massively parallel ultrafast random bit generation with a chip-scale laser," *Science*, vol. 371, no. 6532, pp. 948–952, Feb. 2021, doi: 10.1126/science.abc2666.
[11] A. Rukhin *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications".
[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[13] M. D. Bernardi, M. H. R. Khouzani, and P. Malacaria, "Pseudo-Random Number Generation using Generative Adversarial Networks," Sep. 30, 2018, *arXiv*: arXiv:1810.00378. doi: 10.48550/arXiv.1810.00378.
[14] I. Goodfellow *et al.*, "Generative Adversarial Nets".
[15] L. Pasqualini and M. Parton, "Pseudo Random Number Generation through Reinforcement Learning and Recurrent Neural Networks," *Algorithms*, vol. 13, no. 11, p. 307, Nov. 2020, doi: 10.3390/a13110307.
[16] J. Luis Crespo, J. González-Villa, J. Gutiérrez, and A. Valle, "Assessing the quality of random number generators through neural networks," *Mach. Learn. Sci. Technol.*, vol. 5, no. 2, p. 025072, Jun. 2024, doi: 10.1088/2632-2153/ad56fb.
[17] D. Proskurin, T. Okhrimenko, S. Gnatyuk, D. Zhaksigulova, and N. Korshun, "Hybrid RNN-CNN-based model for PRNG identification⋆".
[18] D. Proskurin, S. Gnatyuk, and T. Okhrimenko, "Predicting Pseudo-Random and Quantum Random Number Sequences using Hybrid Deep Learning Models".

[19] B. J. M. Jawad and S. Al-Alak, "DSK: The Proposed Method for Deriving Secret Keys for encrypting of the block in the networks," in *2023 1st International Conference on Advanced Engineering and Technologies (ICONNIC)*, Kediri, Indonesia: IEEE, Oct. 2023, pp. 259–262. doi: 10.1109/ICONNIC59854.2023.10467460.
[20] P. Ji, H. Ma, Q. Ma, and X. Chen, "A Novel Method to Generate Pseudo-Random Sequence based on GAN".
[21] S. Park, K. Kim, K. Kim, and C. Nam, "Dynamical Pseudo-Random Number Generator Using Reinforcement Learning," *Appl. Sci.*, vol. 12, no. 7, p. 3377, Mar. 2022, doi: 10.3390/app12073377.
[22] Y. Alloun, M. S. Azzaz, A. Kifouche, and R. Kaibou, "Pseudo Random Number Generator Based on Chaos Theory and Artificial Neural Networks," in *2022 2nd International Conference on Advanced Electrical Engineering (ICAEE)*, Constantine, Algeria: IEEE, Oct. 2022, pp. 1–6. doi: 10.1109/ICAEE53772.2022.9962090.
[23] S. N. Hussein and S. M. Al-Alak, "Secret Keys Extraction Using Light Weight Schemes for Data Ciphering," *J. Phys. Conf. Ser.*, vol. 1999, no. 1, p. 012114, Sep. 2021, doi: 10.1088/1742-6596/1999/1/012114.
[24] S. Mohammed, S. M. K. Al-Alak, and H. A. Lafta, "ECC and AES Based Hybrid Security Protocol for Wireless Sensor Networks".
[25] S. Al-Alak, Z. Zukarnain, A. Abdullah, and S. Subramiam, "Randomness improvement of AES using MKP," *Res. J. Inf. Technol.*, vol. 5, pp. 24–34, 2013.
[26] V. Desai, R. Patil, and D. Rao, "Using Layer Recurrent Neural Network to Generate Pseudo Random Number Sequences," vol. 9, no. 2, 2012.
[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," presented at the International conference on machine learning, PMLR, 2017, pp. 214–223.
[28] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," Apr. 05, 2017, *arXiv*: arXiv:1611.04076. doi: 10.48550/arXiv.1611.04076.
[29] Q. Wang *et al.*, "WGAN-Based Synthetic Minority Over-Sampling Technique: Improving Semantic Fine-Grained Classification for Lung Nodules in CT Images," *IEEE Access*, vol. 7, pp. 18450–18463, 2019, doi: 10.1109/ACCESS.2019.2896409.
[30] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "UNROLLED GENERATIVE ADVERSARIAL NETWORKS," 2017.
[31] J. Solomon, J. Solomon, R. M. Rustamov, L. Guibas, A. Butscher, and A. Butscher, "Wasserstein Propagation for Semi-Supervised Learning".
[32] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs".
[33] L. Yuan, Y. Ma, and Y. Liu, "Protein secondary structure prediction based on Wasserstein generative adversarial networks and temporal convolutional networks with convolutional block attention modules," *Math. Biosci. Eng.*, vol. 20, no. 2, pp. 2203–2218, 2022, doi: 10.3934/mbe.2023102.
[34] M. Boden, "A guide to recurrent neural networks and backpropagation".
[35] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Phys. Nonlinear Phenom.*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.
[36] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994, doi: 10.1109/72.279188.
[37] N. Alsadi, S. A. Gadsden, and J. Yawney, "Intelligent estimation: A review of theory, applications, and recent advances," *Digit. Signal Process.*, vol. 135, p. 103966, Apr. 2023, doi: 10.1016/j.dsp.2023.103966.
[38] S. Melacci, L. Sarti, M. Maggini, and M. Bianchini, "A Neural Network Approach to Similarity Learning," in *Artificial Neural Networks in Pattern Recognition*, vol. 5064, L. Prevost, S. Marinai, and F. Schwenker, Eds., in Lecture Notes in Computer Science, vol. 5064. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 133–136. doi: 10.1007/978-3-540-69939-2_13.
[39] V. V. Desai, V. B. Deshmukh, and D. H. Rao, "Pseudo random number generator using Elman neural network," in *2011 IEEE Recent Advances in Intelligent Computational Systems*, Trivandrum, India: IEEE, Sep. 2011, pp. 251–254. doi: 10.1109/RAICS.2011.6069312.