Assessing the quality of Random Number Generators through Neural Networks

José Luis Crespo[1], Javier González-Villa[1], Jaime Gutiérrez[1], and Angel Valle[2]

[1] Departamento de Matematica Aplicada y Ciencias de la Computación, Universidad de Cantabria, Santander, Spain

[2] Instituto de Física de Cantabria, Universidad de Cantabria-CSIC, Santander, Spain

E-mail: luis.crespo@unican.es, javier.gonzalezvilla@unican.es, jaime.gutierrez@unican.es, valle@ifca.unican.es

April 2024

Abstract. In this paper we address the use of Neural Networks (NN) for the assessment of the quality and hence safety of several Random Number Generators (RNGs), focusing both on the vulnerability of classical Pseudo Random Number Generators (PRNGs), such as Linear Congruential Generators (LCGs) and the RC4 algorithm, and extending our analysis to non-conventional data sources, such as Quantum Random Number Generators (QRNGs) based on Vertical-Cavity Surface-Emitting Laser (VCSEL). Among the results found, we identified a sort of classification of generators under different degrees of susceptibility, underlining the fundamental role of design decisions in enhancing the safety of PRNGs. The influence of network architecture design and associated hyper-parameters variations was also explored, highlighting the effectiveness of longer sequence lengths and convolutional neural networks in enhancing the discrimination of PRNGs against other RNGs. Moreover, in the prediction domain, the proposed model is able to deftly distinguish the raw data of our QRNG from truly random ones, exhibiting a cross-entropy error of 0.52 on the test data-set used. All these findings reveal the potential of NNs to enhance the security of RNGs, while highlighting the robustness of certain QRNGs, in particular the VCSEL-based variants, for high-quality random number generation applications.

1. Introduction

Random number generators (RNGs) are widely used in many applications including cryptographycally secured communications, industrial testing, Monte Carlo simulations, massive data processing, lotteries, quantitative finance, fundamental physics tests, etc. [1, 2, 3, 4]. The security analysis of RNGs is a vital issue in classical and quantum cryptographic systems in which secure unpredictable keys are necessary. For instance, an attack against radio-frequency identification of ID cards in Taiwan highlighted the importance of good quality random number generation because a poorly implemented RNG allowed the factoring of 184 RSA keys out of 2 million Taiwanese ID cards [5].

There are two types of RNGs. The first one, called Pseudorandom Number Generators (PRNGs), are based on algorithms that deterministically expand a small number of bits truly randomly generated (called the random seed) to a larger sequence. The second type, called True Random Number Generators (TRNGs), produce the random sequences by measuring some physical phenomenon that is expected to be random. The generated numbers are further processed for compensating for possible biases in the measurement process.

Quantum physics can be exploited to generate true random numbers that are completely unpredictable due to the inherent randomness to quantum mechanics. The TRNGs that use quantum sources to produce random numbers are known as Quantum Random Number Generators (QRNGs). Most existing QRNGs are based on quantum optics because of the availability of high-quality optical components and the possibility of chip-size integration [2]. One of the most successful strategies for quantum random number generation is that based on gain-switching of semiconductor lasers [6, 7, 8, 9]. In this technique the current applied to a semiconductor laser is modulated in a periodic way from below to above its threshold for obtaining gain-switching operation. The pulses emitted by the laser have random phases due to the effect of spontaneous emission noise. Laser phase noise is a source of quantum randomness resulting from spontaneous emission [10, 11]. These pulses are mixed with their delayed versions using an interferometer and the obtained pulses have random amplitudes. The detection and postprocessing of these amplitudes provide the random numbers [7, 8]. Advantages of these QRNGs include fast operation (with a flexible speed up to tens of Gbps), robustness and the integration in photonic integrated circuits [12]. These QRNGs are widely used in current real-world Quantum Key Distribution (QKD) systems [9]. QRNGs based on gain-switching of an special type of semiconductor laser, the vertical-cavity surface-emitting laser (VCSEL), have also been recently demonstrated [13, 14, 15]. QRNGs based on VCSELs have the extra advantages of low fabrication cost and simplicity (coherent detection is not required since the interferometric element is removed) [13, 14, 15]. This QRNG is the specific TRNG that will be considered in this paper.

The standard approach to experimentally evaluate the fitness of RNGs consists of running statistical tests of the generated numbers to detect bias, correlations or other signs of nonrandomness. Examples of the most popular tests are NIST-STS [16] and DIEHARD [17]. Passing these tests is a necessary but not sufficient condition for assuring the correct operation of a RNG. For instance simple PRNGs like the linear congruential generator (LCG) pass NIST tests [18] but are not suitable for cryptographic purposes [19, 20]. There are also several theoretical measures of pseudorandomness [21]. However they are quite difficult to test in practice because of their high computational complexity.

An alternative approach to evaluate the randomness of a given RNG is the use of neural networks (NN). NN are well known machine learning (ML) tools that have led to important advances in recent years in many fields, such as image and video

object segmentation, medical imaging, face recognition, time series prediction, signal identification, image classification, object detection or human action recognition (see for instance [22, 23]). Recently, several research studies [18, 24, 25, 26, 27, 28, 29, 30] have suggested the use of neural networks for evaluating the quality of RNGs. Machine learning can automate the testing of RNGs to detect biases, anomalies and patterns that might compromise the integrity of the generated random sequences in applications like security, gaming and cryptography where randomness is essential. In this way ML-based models can provide an efficient supplement for evaluating the quality and security of RNGs.

Two main directions have been followed to address this problem. First, NN have been used for predicting the output of a RNG. Simple dense feedforward NN were used to predict the next bits in LCG [24] while they were not able to predict the next bits in the standard Python PRNG [26]. More complicated architectures have also been used. For instance, several types of temporal pattern attention (TPA)-based deep-learning (DL) models were used to predict the output data of both, a LCG and a chaotic semiconductor laser [18]. The same model was not able to predict the output of a TRNG based on the optical heterodyning of two chaotic semiconductor lasers [18], showing in this way that this TRNG has strong resistance against the predictive model [18].

In the second direction, the NN is not trying to predict the following bit, but to tell the RNG apart from a given *golden standard* RNG (GSRNG) [25]. A GSRNG is a RNG that generates ideal random numbers. Authenticating a RNG as a GSRNG poses another problem that is not analyzed in this paper. However we note that some PRNGs like the Blum-Blum-Shub generator (BBS) [31] can be considered as GSRNG because it has been demonstrated that its security is reduced to the computational difficulty of factoring [31]. Following these ideas a NN was trained to tell a PRNG apart from a GSRNG [25]. The method was successfully applied for searching statistical biases in two PRNGs using multilayer Long Short-Term Memory (LSTM) neural networks [25]. The method automatically discovered unknown types of statistical biases using LSTMs to detect slight differences between the target PRNG's output and ideal random numbers.

In this paper we follow this approach in order to extend the previous evaluation to TRNGs. The particular case of TRNGs that we consider is the VCSEL-based QRNG [13, 14]. Similarly to [25], we consider a well-trusted PRNG, HMAC-DRBG on NIST SP 800-90A [32], as our GSRNG. We train LSTM to try to tell our QRNG apart from the GSRNG. A similar analysis is also performed for a variety of RNGs, including LCGs, the binary codification of a large video, and linear Congruential Generator on Elliptic Curves (EC-LCG). We also extend our analysis for considering convolutional neural network (CNN) models. We show that the ability to distinguish between sequences coming from the target RNG and from the GSRNG is significantly better for CNN than for LSTM. Sequences of numbers obtained from our QRNG and the GSRNG are indistinguishable even with the biggest LSTM or CNN that we have considered.

The paper is organized as follows. We start with a very short outline of the random number generators in Section 2, and neural networks in Section 3. In Section 4 we briefly

review the topics related and present our method upon the work of [25]. In Section 5 we show the results of our tests. Finally, in Section 6 we discuss our results and present our conclusions.

2. Random number generators

In this section we describe the different types of analysed RNGs. First and second subsections are devoted to the TRNG and PRNGs considered in this work, respectively.

2.1. VCSEL-based Quantum Random Number Generator

A complete description of the QRNG based on gain-switching VCSELs is presented in [13, 14]. VCSELs are characterised by the possibility of emission in two orthogonal linearly polarised modes [33]. This means that the lasing electrical field can oscillate along two orthogonal directions in the plane perpendicular to the laser beam. In [13, 14] the VCSEL is gain-switched, that is the current is periodically modulated in such a way that the linearly polarised mode that is preferably excited in each period is random. Random excitation of the VCSEL polarisations can be considered as a quantum entropy source because it is triggered by the spontaneous emission events that are quantum mechanical in nature [10].

We show in Fig. 1, with blue and red lines, the temporal waveform of the signals corresponding to both linearly polarised modes (x and y) obtained in the experiment, $V_x(t)$ and $V_y(t)$. The experimental setup is shown in Fig. 1 of [14]. These signals are proportional to the power of the $x-$ and $y-$linearly polarised modes, respectively. The VCSEL switch-off in all pulses (22 consecutive pulses are shown) in such a way that there is a random excitation of both linearly polarised modes in each modulation period.

Random bits are obtained by comparing the x and y signals when the sum of both signals is maximum, $V_x(t_{max})$ and $V_y(t_{max})$, respectively. If $V_x(t_{max})$ is larger than $V_y(t_{max})$ we assign a "0" bit, otherwise we assign a "1" bit, as illustrated in Fig. 1. These bits constitute the raw output of our QRNG, that as in all TRNGs, shows deviations from the mathematical ideal of statistically independent and uniformly distributed bits [3, 34, 35]. In fact, the raw bit sequence has not passed the NIST test suite [14]. This problem is addressed by applying an additional post-processing step to decrease bias in the bit stream (defined as $e = p(0) - 1/2$, where $p(0)$ is the probability of obtaining a "0" bit) and to increase the bit entropy. We consider the post-processed bit string based on the following result: [36] Let G be a linear corrector mapping n bits to k bits. Then the bias of any non zero linear combination of the output bits is less or equal than $2^{d-1}e^d$, where d is the minimal distance of the linear code constructed by the generator matrix G.

We have used the efficient $[n, k, d]$-BCH codes defined over the finite field $GF(2)$ and where $n + 1$ is a power of 2.

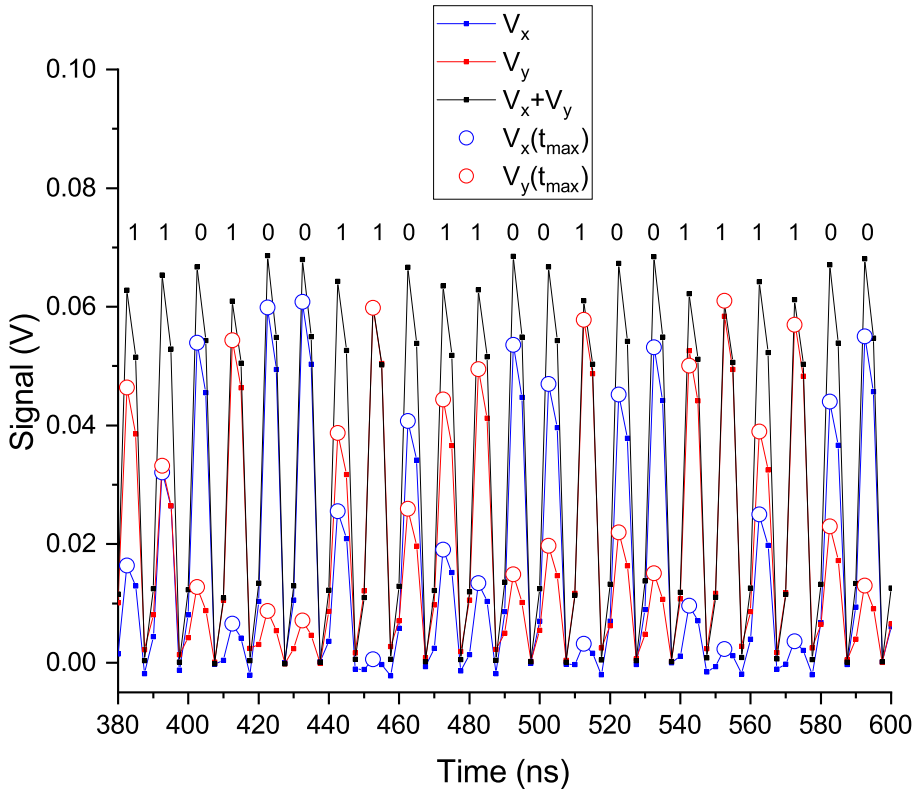


Figure 1: Experimental time traces of the signals corresponding to the x-polarisation (blue line), y-polarization (red line), and total power (black line). The signals at the sampling time are also plotted with symbols.

For the raw input bits $(x_{n-1}, \ldots, x_0)$, the output $(y_{k-1}, \ldots, y_0)$ is obtained as:

$$\begin{pmatrix} g_{n-k} & \ldots\ldots\ldots & g_0 & 0\ldots\ldots 0 \\ 0 & g_{n-k} & \ldots\ldots\ldots g_0 & 0\ldots\ldots 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0\ldots\ldots & 0 & g_{n-k} & \ldots\ldots g_0 \end{pmatrix} \begin{pmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_0 \end{pmatrix} = \begin{pmatrix} y_{k-1} \\ y_{k-2} \\ \vdots \\ y_0 \end{pmatrix}$$

where $g(x) = g_{n-k}x^k + \cdots + g_1 x + g_0$ is the cyclic generator polynomial of the $[n, k, d]$-BCH code. Here we have considered BCH code with parameters $[1023, 1003, 5]$ the generator cyclic polynomial is $x^{20} + x^{15} + x^{13} + x^{12} + x^{11} + x^9 + x^7 + x^6 + x^3 + x^2 + 1$. Using this post-processing we have obtained a sequence of 3.9595×10^9 post-processed bits that constitute the output of our QRNG [14].

2.2. Pseudorandom Number Generators

The experiment is also carried out by using two PRNG's: the Linear Gongruential Generator, and the linear Congruential Generator on Elliptic Curves.

2.2.1. Linear Congruential Generator(LCG). Given positive integers a, b and m such that $\gcd(a, m) = 1$ the *Linear Congruential Generator(LCG)* is a sequence x_n of pseudorandom numbers defined by the relation

$$x_{n+1} \equiv (ax_n + b) \bmod m, \qquad n = 0, 1, \ldots,$$

where x_0 is the *seed*. Unfortunately the LCG is not suitable for cryptographic purposes, see [19, 20]. Although the author [37] claims that NIST test suites cannot detect the linearity.
In this computational experiment we took the sequences from the rand function in the glibc library version 2-17 without any tunning such that $m = O(2^{32})$ bits, and the output of simple Python LCG code with $m = O(2^{100})$.

2.2.2. Linear Congruential Generator on Elliptic Curves(EC-LCG). For a prime p, we denote by $\mathbf{F}_p \cong \mathbf{Z}p$ the field of p elements and, we assume that it is represented by the set $\{0, 1, \ldots, p-1\}$.

Let E be an elliptic curve defined over $\mathbf{F}_p$ given by an *affine Weierstrass equation*, which for $\gcd(p, 6) = 1$ takes form $Y^2 = X^3 + aX + b$, for some $a, b \in \mathbf{F}_p$ with $4a^3 + 27b^2 \neq 0$.

We recall that the set $E(\mathbf{F}_p)$ of $\mathbf{F}_p$-rational points forms an abelian group, with the *point at infinity* $\mathcal{O}$ as the neutral element of this group (which does not have affine coordinates).

For a given point $G \in E(\mathbf{F}_p)$ the *Linear Congruential Generator on Elliptic Curves, EC-LCG* is a sequence U_n of pseudorandom numbers defined by the relation

$$U_n = U_{n-1} \oplus G = nG \oplus U_0, \quad n = 1, 2, \ldots,$$

where $\oplus$ denote the group operation in $E(\mathbf{F}_p)$ and $U_0 \in E(\mathbf{F}_p)$ is the *initial value* or *seed*. We refer to G as the *composer* of the EC-LCG.

The EC-LCG provides a very attractive alternative to linear and non-linear congruential generators with many applications to cryptography and it has been extensively studied in the literature, see [38, 39, 40, 41].

We have generated a .txt file of 2^{20} bits running the following SageMath code:

```
f = open('/Users/PRNG/Desktop/EC_LG.txt', 'a')
size_prime = 512
p=next_prime(ZZ.random_element(2**size_prime))
a=ZZ.random_element(p)
b=ZZ.random_element(p)
if (4*a**3+27*b**2)%p != 0:
    C =EllipticCurve(GF(p),[a,b])
G=C.random_element()
U0=C.random_element()
for i in range(500):
    V=U0+i*G
    f.write(bin(V[0])[2:]+bin(V[1])[2:])
f.close()
```

3. Neural Networks

In the current landscape of random sequence analysis, neural networks have introduced a paradigm shift, offering new insights and approaches to improve understanding and processing. Random sequence analysis poses a unique set of challenges, where discerning patterns and dependencies can be much more difficult compared to classical structured data. In this section, we briefly summarise most promising architectures for extracting information from random sequences, CNN and LSTM, which have brought new possibilities for predicting and categorising the seemingly random nature of sequences generated via TRNGs and PRNGs.

3.1. LSTM networks

A long-term memory network (LSTM) [42] is a specialised recurrent neural network designed to model sequential data [43]. Unlike standard recurrent networks, LSTMs have a single memory cell equipped with gating mechanisms. Within this cell, the input sequence, whether derived from external sources or the preceding layer's output, undergoes intricate filtering via distinct gates, which encompass the integration of the previous cell state. This integration with the previous state imparts dynamic and memory-retentive capabilities to this model, see Figure 2. This allows them to capture and retain long-range dependencies and temporal context within sequences, making them well suited for a variety of applications involving sequential data including random sequence prediction and classification. Their ability to capture long-range dependencies and context allows them to identify hidden patterns and temporal relationships in seemingly chaotic data.

LSTMs are also adept at dealing with sequences of varying length, which matches the unpredictability and variability of random sequences. Their adaptive capabilities

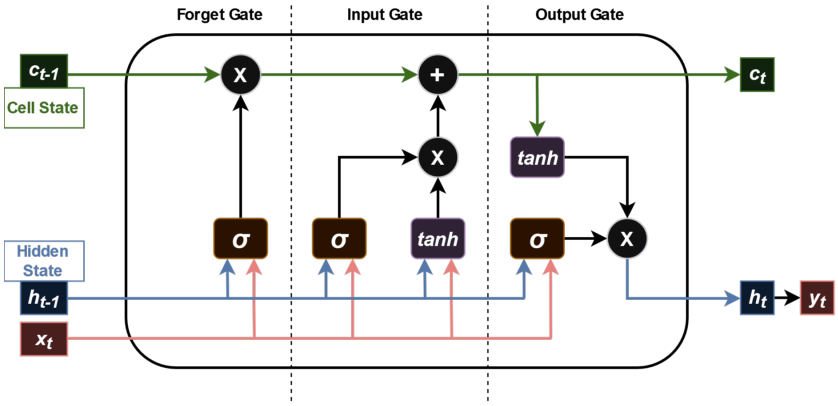


Figure 2: LSTM cell: the lower row shows standard processors, whereas the upper rows are parameter-free calculations. Note the feedback connection from third to second layer.

allow them to process and predict sequences without prior knowledge of their specific characteristics, making them ideal for the downstream application framework presented in a posterior section. In scenarios where traditional models may struggle to provide accurate predictions or classifications due to the inherent irregularities and complexities of RNGs, LSTMs have proven to be indispensable tools showing their ability to encode temporal dependencies and detect subtle patterns.

3.2. Convolutional networks

Convolutional networks, following diverse convolutional schemes (see Figure 3), are a sophisticated evolution of traditional multilayer perceptrons, which are able to refine and filter the process of information transformation within neural networks. In a standard multilayer perceptron, neurons—essentially processing units—receive the outputs from the preceding layer. These outputs undergo filtration through a nonlinear function, with the adjustable parameters manifesting as the input weights assigned to individual neurons in the layer.

When confronted with input signals representing temporal measurements of a variable, such as a sequence of consecutive random numbers, a convolutional approach proves advantageous. This involves a convoluted processing strategy, wherein a nonlinear convolution of the input array with a smaller weight array is computed. In this scenario, each neuron becomes a filter, engaging in a finite convolution of the input signal through a nonlinear gate, ultimately producing another output sample. While various step sizes for the convolution are theoretically plausible, the width of the convolution kernel typically emerges as the key design parameter.

Expanding our perspective to scenarios where the input comprises multiple sequences, the convolutional layer can be flexibly configured to amalgamate information across these sequences. This can be achieved either through summation or by independently processing each sequence. Such adaptability results in a spectrum of

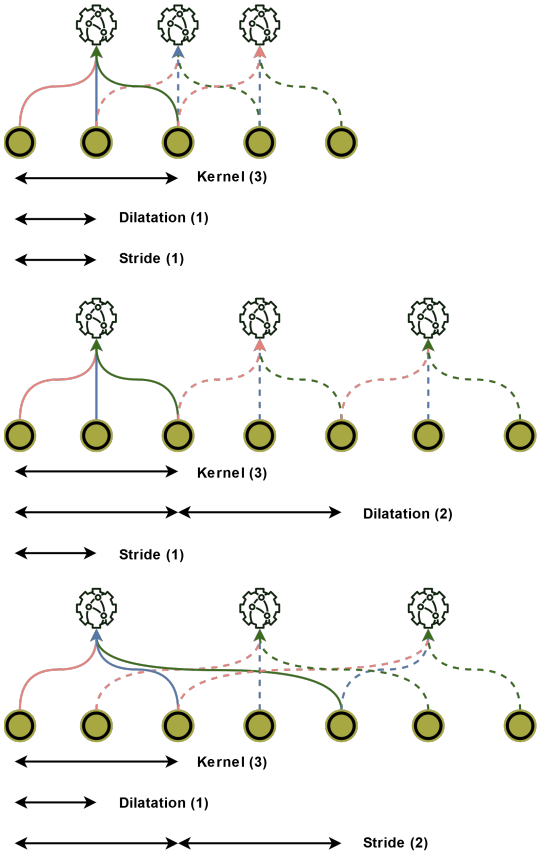


Figure 3: Convolutional processor configurations depicted in various schemes. The upper row illustrates successive values of the processor, highlighted in green. The lower row displays successive input values (single variable for simplicity), highlighted in yellow.

configurations for convolutional neural networks, empowering the creation of customised architectures that effectively address specific contextual and structural intricacies inherent in the input data.

4. Proposed framework implementation

In our efforts to discern patterns and vulnerabilities in random sequences, we have embarked on two methodological avenues: classification and prediction. These approaches are designed to elucidate the effectiveness of neural networks in distinguishing between different types of random number generators (RNGs) and predicting their outcomes.

Initially, we apply a classification methodology inspired by previous research described in [25]. This method consists of training a neural network to differentiate between a PRNG and a truly random sequence, exemplified by the sequences generated by HMAC-DBRG. The evaluation criteria include assessing the network's ability to distinguish between these categories of sequences. Our implementation, based on the architecture described in [25], features a single hidden layer of long-term memory

(LSTM) composed of 80 cells. We maintain a consistent configuration with 2^{15} sequences consisting of 256 elements, each of which represents one byte of the random sequence. The training spans 100 iterations, with a learning rate of 0.01 and a minibatch size of 250. An essential evaluation criterion, proposed by the authors of [22], consists in scrutinising the deviation of 0.5 in the average output for each type of sequence (GSRNG and the tested PRNG). Furthermore, in line with our main goal of discerning between RNGs, especially in cases of varying quality, we have focused our attention on convolutional networks. Taking advantage of their inherent interpretability and ability to detect sequential patterns and features, we use bits extracted from 256-byte sequences as inputs to our neural network. Rigorous training is performed to effectively distinguish between HMAC-DBRG sequences and other sequences, putting an special effort in trying to distinguish between true random and raw laser sequences (Raw QRNG). Our methodology is developed sequentially. Initially, we perform pruning and reduction processes on the network, in order to preserve a significant part of its discriminative capacity. Subsequently, we analyse the weights to identify the inputs that influence the response of the network.

Although derivatives have been examined, their inherent value in identifying relevant connections is limited. Despite the fact that the mean values are often tiny (averaging less than 0.005 over the initial thousand sampling points), their standard deviation exceeds the mean, with a maximum of 0.04, suggesting possible significance. Our overall goal remains to discover the weight connections that run through the network from the inputs to the outputs. To validate the importance of these connections, we further debug the network and evaluate its efficiency in distinguishing between RNGs.

Furthermore, we have ventured into predictive modeling to anticipate the RNG's future outcomes based on its past performance. Inspired by the effective recognition of raw laser sequences using classification networks, we've dedicated our attention to predictive modelling within this specific domain. Recognising the raw QRNG's identification by classification networks, we narrowed our focus to it. Faced with unsuccessful attempts at analysing individual bits, we shifted our approach to average byte values. Our central prediction task involved determining whether the average bit value in the next 127 bytes, given the preceding 256 bytes, would surpass or fall below 0.5. This revised goal was more feasible in our specific context, employing a type 4 convolutional network (as illustrated in Figure 4).

5. Results

5.1. Classification

Our first step was to reproduce the results in [25]. Next we have proceeded to apply the same framework to more PRNG's, including those coming from laser output. Finally, we have tested the influence of some hyper-parameter variations.

Different runs may yield different results. The ones shown are representative

samples. In some cases where we found strong deviations, we show more than one result.

5.1.1. Reproducing [25] In [25], the proposed framework was applied successfully to detect weaknesses in LCG and the second byte of RC4, with different training sizes.

Concerning the second byte in RC4, the pseudo-random sequence comes from applying a key sequence to the RC4 algorithm. In [25] the key source is not specified. We have used a random sequence of 64 bytes values coming from HMAC-DBRG.

Concerning LCG, we took sequences from the rand function in the glibc library version 2.17, without any tuning.

We haven't tested all training sizes appearing in [25], but only some. As can be seen in table 1 our implementation yields comparable results. In perfect classification, the average output (AO) should be 1 for the tested PRNG and 0 for the GSRNG.

Table 1: Comparison between some of our results and those reported in [25].Top table: Three results trying to mimic some of those reported in [25]. We show variability by providing the output of two different runs with the same training size for LCG. Bottom table: Results taken from [25], compatible with ours in top table.

PRNG tested	Training size	AO. PRNG	AO. GSRNG
RC4 2nd byte	2^{13}	0.53	0.42
LCG	2^{18}	0.63	0.19
LCG	2^{18}	0.87	0.47

PRNG tested	Training size	AO. PRNG	AO. GSRNG
RC4 2nd byte	2^{13}	0.51	0.38
LCG	2^{18}	0.92	0.10

5.1.2. Testing other PRNG's We then have proceeded to apply the proposed framework to several other PRNG's, including a elliptic curve generator, and the proposed QRNG, raw and postprocessed. We even considered a large enough video file as a source of random (meaning unpredictable) sequences; for this purpose we arbitrarily chose episode 7 of the continuing education course [44]

We show the results in table 2, including confusion matrices.

We can see that the VCSEL QRNG passes the test. Not so for the raw QRNG. The elliptic curve generator is also successful. We can also see that adding a periodic parameter reset to the LCG is enough to make it pass the test. The video file didn't pass the test, but its performance wasn't that bad; it may rank as good as a naive LCG.

Table 2: Results applying the proposed framework to other PRNGs.

PRNG tested	Training size	AO. PRNG	AO. GSRNG	Confusion matrix
VCSEL QRNG	2^{18}	0.47	0.47	$\begin{pmatrix} 16324 & 0 \\ 16444 & 0 \end{pmatrix}$
VCSEL QRNG	2^{19}	0.51	0.51	$\begin{pmatrix} 72 & 16255 \\ 81 & 16360 \end{pmatrix}$
Raw QRNG	2^{18}	0.73	0.55	$\begin{pmatrix} 6543 & 9934 \\ 3235 & 13056 \end{pmatrix}$
EC-LCG	2^{18}	0.49	0.49	$\begin{pmatrix} 15650 & 845 \\ 15428 & 845 \end{pmatrix}$
Video	2^{18}	0.58	0.33	$\begin{pmatrix} 14475 & 1822 \\ 7831 & 8640 \end{pmatrix}$
LCG (32 bits)	2^{18}	0.51	0.42	$\begin{pmatrix} 11625 & 4769 \\ 7910 & 8464 \end{pmatrix}$
LCG (100 bits)	2^{18}	0.50	0.51	$\begin{pmatrix} 8827 & 7458 \\ 7729 & 8754 \end{pmatrix}$

5.1.3. Network design and parameters influence Table 3 shows the effects of several design decision changes, namely: number of processors, network type, layers, sequence length, bytes per element.

LCG refers to the naive rand LCG implementation. Convolutional neural networks are designed as depicted in figure 4.

The design decisions that turned effective were: increasing sequence length and using convolutional networks. VCSEL QRNG was indistinguishable even with these more capable settings.

The design decisions that turned effective were: increasing sequence length and using convolutional networks. VCSEL QRNG was indistinguishable even with the biggest networks that we have tried (twice the size of the biggest ones reported here). It remains an open question whether a massively larger network would yield better results.

We also tested how far we could go in simplifying the convolutional networks capable of detecting the raw laser sequence.

In this process, we directed our attention towards discerning chains of substantial weights spanning from input to output nodes within the network. Our aim was to evaluate the significance of these chains in determining network functionality. To validate our hypothesis, we pruned redundant components of the network, subsequently evaluating its efficacy in distinguishing between different Random Number Generators (RNGs). The comparison between the original network (see Figure 4a) and the pruned version is depicted in Figure 4c. The confusion matrices for the original and pruned

Table 3: Performance evaluation of the neural network model with sequential application of hyperparameter or architecture settings.

Design decision	Tested PRNG	Training size	AO. PRNG	AO. GSRNG	Confusion matrix
Baseline [25]	LCG	2^{15}	0.16	-0.11	$\begin{pmatrix} 16229 & 0 \\ 16504 & 35 \end{pmatrix}$
Increasing processors from 80 up to 150	LCG	2^{15}	0.21	-0.06	$\begin{pmatrix} 16480 & 1 \\ 16241 & 46 \end{pmatrix}$
Two layers with 40 and 10 processors	LCG	2^{15}	0.36	0.41	$\begin{pmatrix} 14734 & 1742 \\ 12770 & 3522 \end{pmatrix}$
Increasing sequence length from 2^8 to 2^9	LCG	2^{15}	0.61	0.23	$\begin{pmatrix} 13689 & 2832 \\ 3706 & 12541 \end{pmatrix}$
CNN-3 (see fig. 4c)	Raw QRNG	2^{18}	0.40	0.61	$\begin{pmatrix} 14217 & 2272 \\ 8107 & 8172 \end{pmatrix}$
Increasing sequence length from 2^8 to 2^9	VCSEL QRNG	2^{18}	0.51	0.51	$\begin{pmatrix} 177 & 16146 \\ 147 & 16298 \end{pmatrix}$
CNN-1	LCG	2^{18}	1	0	$\begin{pmatrix} 16492 & 35 \\ 0 & 16233 \end{pmatrix}$
CNN-1	VCSEL QRNG	2^{19}	0.5	0.5	$\begin{pmatrix} 8218 & 8133 \\ 8165 & 8252 \end{pmatrix}$
CNN-2 and sequence length=512	VCSEL QRNG	2^{18}	0.5	0.5	$\begin{pmatrix} 9696 & 6644 \\ 9853 & 6575 \end{pmatrix}$
CNN-2 and 2-byte elements	VCSEL QRNG	2^{18}	0.5	0.5	$\begin{pmatrix} 8275 & 8140 \\ 8292 & 8061 \end{pmatrix}$

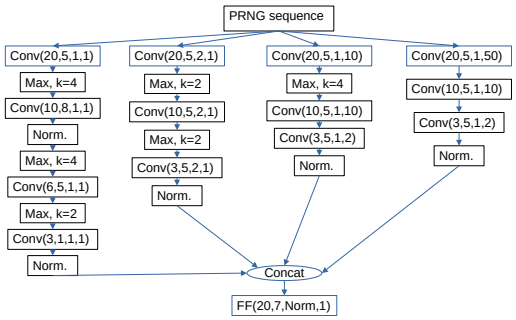
networks, generated from a test set comprising 2^{15} instances, are presented Table 4.

Table 4: Comparison between original confusion matrix (top) and pruned network confusion matrix (bottom).

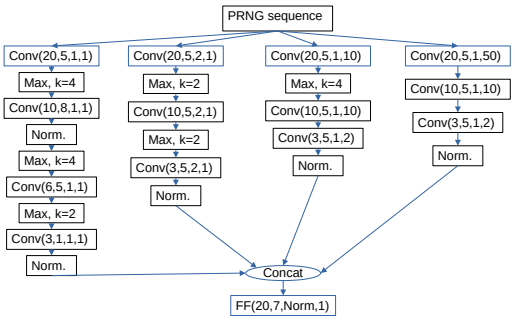
	HMAC-DRBG	Raw QRNG
HMAC-DRBG	14149	2219
Raw QRNG	7699	8701

	HMAC-DRBG	Raw QRNG
HMAC-DRBG	14217	2272
Raw QRNG	8107	8172

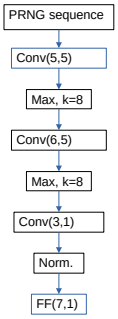
In the final pruned network, an analysis was conducted to identify the primary patterns (see Table 5) influencing network response. Notably, with the initial kernel's width set at 5 for input connections, specific patterns emerged as key triggers for network discharge, indicating the raw laser's role. Among these patterns (see Table 5), those with lesser influence due to lower weights were identified and denoted as "_". Pattern 5 was recognised as encompassing pattern 3, leading to a focus on the more specific elements, see Table 5. Further examination aimed to distinguish unique features within these patterns across the sample data. Figure 5 illustrates distinct relative frequencies



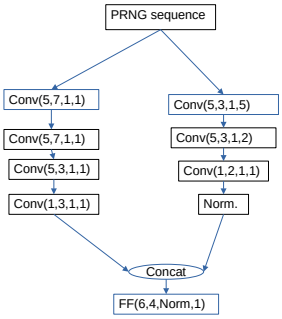
(a) Convolutional network scheme 1



(b) Convolutional network scheme 2



(c) Convolutional network scheme 3



(d) Convolutional network scheme 4

Figure 4: Convolutional models description: Conv stands for convolutional layer and the four parameters are: number of processors/channels, kernel width, stride and dilation. Max stands for reduction layers using maximum, with the given kernel width. Norm stands for normalization layer. FF stands for group of standard feedforward layers, with given number of processors per layer. Leaky rectified linear units are used everywhere but in the output layer, where we have sigmoid activation.

for the pattern 01011 among various sequences, indicating notable variations. Similarly, Figure 6 highlights significant differences in relative frequency for the 10000 pattern. While other patterns with diverse distributions were present but not selected by the network, this analysis underscores the identification of influential patterns that contributed to network performance enhancements.

5.2. Prediction

Since the raw QRNG was recognized by the classification networks, we focused on it. Since working with individual bits did not succeed, we turned to byte average values. Our final prediction problem was, given the average value of bits in previous 256 bytes,

Table 5: Primary Patterns Influencing Network Response in the Final Pruned Network.

Pattern No.	Patterns Kernel 5	Specific Patterns
1	10000	10000
2	101 _ 0	101 _ 0
3	01011	01011
4	11 _ _ _	11 _ _ _
5	0 _ 1 _ _	
6	110 _ 1	110 _ 1
7	1 _ 0 _ _	1 _ 0 _ _

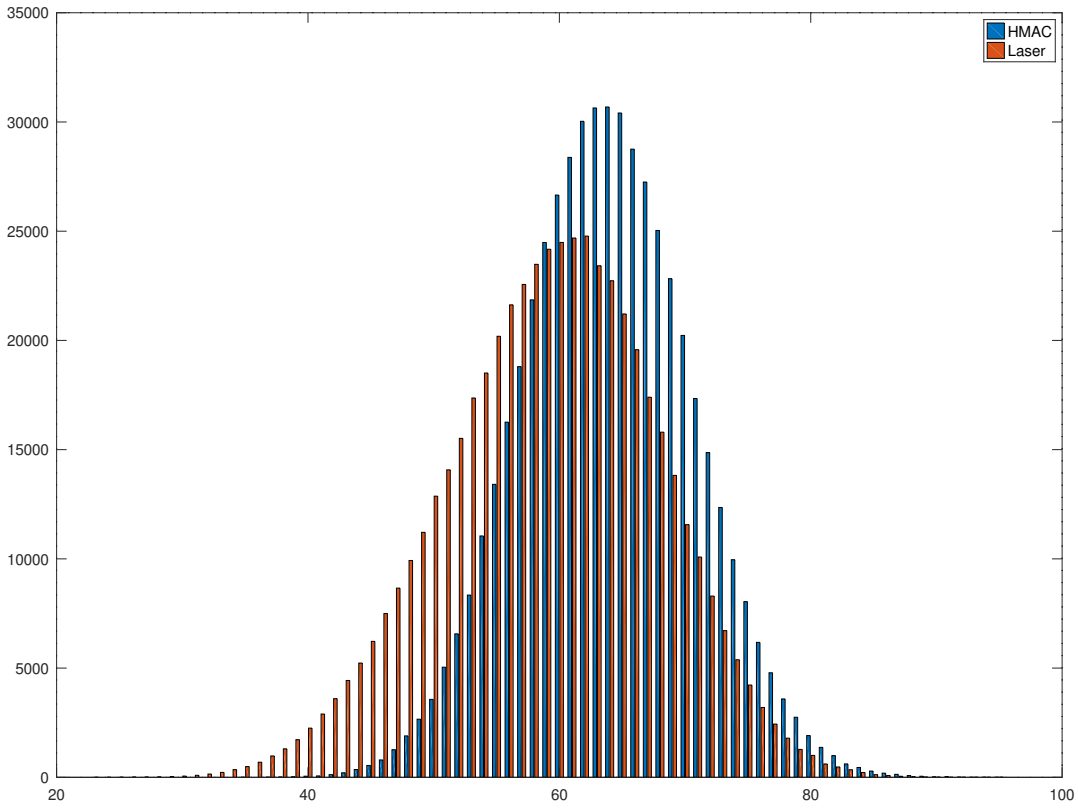
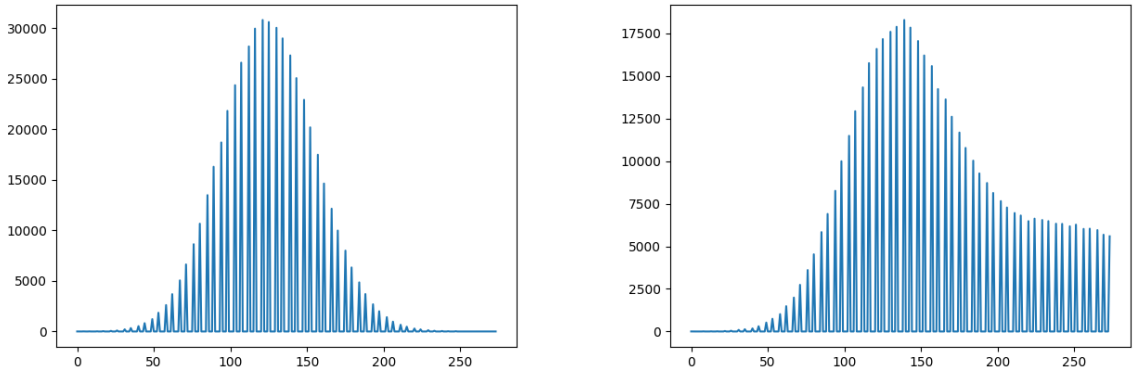


Figure 5: Comparative Analysis of the Frequency Distribution of the 01011 pattern in HMAC-DRBG and Raw Laser Sequences. In the histogram, the abscissa is the number of the 01011 pattern occurrences and the ordinate is the number of sequences of length 2^{11} bits in which there have been that number of occurrences. The distribution in HMAC-DRBG conforms to a normal distribution, while the raw laser sequence exhibits a biased distribution

predict whether the average bit value in the following 127 bytes would be larger or smaller than 0,5 This weaker result, was easier to achieve with the prediction network. We used a convolutional network type 4 (see fig.4d) that was able to give a cross-entropy error of 0.52 on a 2^{15} test set. Other error measurements are listed here (referred to the



(a) HMAC-DRBG (b) Raw QRNG

Figure 6: Comparative Analysis of the Frequency Distribution of the 10000 pattern for HMAC-DRBG and Raw QRNG. In both histograms, the abscissa is the number of 10000 pattern occurrences and the ordinate is the number of sequences of length 2^{12} bits in which there have been that number of occurrences. The evident bias in the raw laser distribution is prominently displayed.

1 label): specificity = 0.56, precision = 0.7, sensitivity = 0.89 and predictive value = 0.61.

6. Discussion and Conclusions

The study delves into the quality assessment of RNGs using neural networks, focusing on TRNGs and PRNGs. It explores the application of Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) to analyse random sequences, demonstrating their effectiveness in capturing patterns and dependencies in RNG outputs. The research leverages a quantum entropy source based on VCSEL polarisation excitation to generate random bits, addressing deviations from ideal randomness through post-processing techniques. It evaluates various PRNGs, including the Linear Congruential Generator on Elliptic Curves (EC-LCG), highlighting its potential for cryptographic applications. Neural networks, particularly LSTM and CNN architectures, play a pivotal role in discerning patterns within random sequences and predicting RNG outcomes effectively. In the realm of random sequence analysis, neural networks offer new insights and approaches to understanding and processing complex data. LSTMs excel in modelling sequential data by capturing long-range dependencies and temporal context, making them ideal for predicting and categorising random sequences. On the other hand, convolutional networks refine information transformation within neural networks, enabling effective pattern detection in random sequences. The study's proposed framework implements classification and prediction methodologies to distinguish between different RNG types and predict their outcomes. By training neural

networks to differentiate between PRNGs and truly random sequences, the research showcases the effectiveness of LSTM and CNN in processing random data. The results demonstrate the success of the application of neural networks to discern the quality of RNGs and predict possible patterns in the random sequences, always having in mind the limitations imposed by the framework and experimental setup proposed in the design phase.

Acknowledgements

This work was supported by Ministerio de Ciencia e Innovación. PID2021-12345OB-C22 MCIN /AEI /10.13039/ 501100011033/FEDER,UE. J. G. is partially supported by grant PID2019-110633GB-I00 funded by MCIN/AEI/10.13039/ 501100011033. We also acknowledge Advanced Computing and e-Science group at the Institute of Physics of Cantabria, IFCA (CSIC-Universidad de Cantabria).

References

[1] Stipčević M and Koç Ç K 2014 True random number generators *Open Problems in Mathematics and Computational Science* (Springer) pp 275–315
[2] Ma X, Yuan X, Cao Z, Qi B and Zhang Z 2016 *npj Quantum Information* **2** 1–9
[3] Herrero-Collantes M and Garcia-Escartin J C 2017 *Reviews of Modern Physics* **89** 015004
[4] Alkhazragi O, Lu H, Yan W, Almaymoni N, Park T Y, Wang Y, Ng T K and Ooi B S 2023 *Annalen der Physik* 2300289
[5] Hurley-Smith D and Hernandez-Castro J 2020 *ACM Transactions on Privacy and Security (TOPS)* **23** 1–25
[6] Jofre M, Curty M, Steinlechner F, Anzolin G, Torres J, Mitchell M and Pruneri V 2011 *Optics express* **19** 20665–20672
[7] Abellán C, Amaya W, Jofre M, Curty M, Acín A, Capmany J, Pruneri V and Mitchell M 2014 *Optics express* **22** 1645–1654
[8] Yuan Z, Lucamarini M, Dynes J, Fröhlich B, Plews A and Shields A 2014 *Applied Physics Letters* **104** 261112
[9] Paraïso T K, Woodward R I, Marangon D G, Lovic V, Yuan Z and Shields A J 2021 *Advanced Quantum Technologies* **4** 2100062
[10] Loudon R 2000 *The quantum theory of light* (OUP Oxford)
[11] Lovic V, Marangon D G, Lucamarini M, Yuan Z and Shields A J 2021 *Physical Review Applied* **16** 054012
[12] Abellan C, Amaya W, Domenech D, Muñoz P, Capmany J, Longhi S, Mitchell M W and Pruneri V 2016 *Optica* **3** 989–994
[13] Quirce A and Valle A 2022 *Optics Express* **30** 10513–10527
[14] Valle-Miñón M, Quirce A, Valle A and Gutiérrez J 2022 *Optics Continuum* **1** 2156–2166
[15] Quirce A, Valle A, Valle-Miñón M and Gutiérrez J 2024 *JOSA B* **41** 240–250
[16] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A *et al.* 2010 *NIST Special Publication* **800** 22
[17] Marsaglia G 1996 *http://stat. fsu. edu/geo*
[18] Li C, Zhang J, Sang L, Gong L, Wang L, Wang A and Wang Y 2020 *Entropy* **22** 1134
[19] Boyar J 1989 *Journal of the ACM (JACM)* **36** 129–141
[20] Knuth D 1985 *IEEE Transactions on Information Theory* **31** 49–52
[21] Goldreich O 2010 *A primer on pseudorandom generators* vol 55 (American Mathematical Soc.)
[22] Cong S and Zhou Y 2023 *Artificial Intelligence Review* **56** 1905–1969
[23] Li Z, Liu F, Yang W, Peng S and Zhou J 2021 *IEEE transactions on neural networks and learning systems*
[24] Amigo G, Dong L and Ii R J M 2021 Forecasting pseudo random numbers using deep learning *2021*

15th International Conference on Signal Processing and Communication Systems (ICSPCS) (IEEE) pp 1–7

[25] Kimura H, Isobe T and Ohigashi T 2019 Neural-network-based pseudo-random number generator evaluation tool for stream ciphers *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)* (IEEE) pp 333–338

[26] Maksutov A A, Goryushkin P N, Gerasimov A A and Orlov A A 2018 Prng assessment tests based on neural networks *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* (IEEE) pp 339–341

[27] Wen Y and Yu W 2019 *Electronics Letters* **55** 515–517

[28] Yu W and Wen Y 2019 *Electronics Letters* **55** 1080–1082

[29] Truong N D, Haw J Y, Assad S M, Lam P K and Kavehei O 2018 *IEEE Transactions on Information Forensics and Security* **14** 403–414

[30] Nagy I and Suciu A 2021 Randomness testing with neural networks *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)* (IEEE) pp 431–436

[31] Blum L, Blum M and Shub M 1986 *SIAM Journal on computing* **15** 364–383

[32] NIST 2015 Recommendation for random number generation using deterministic random bit generators

[33] Michalzik R 2012 Vcsel fundamentals *VCSELs: fundamentals, technology and applications of vertical-cavity surface-emitting lasers* (Springer) pp 19–75

[34] Dichtl M 2007 Bad and good ways of post-processing biased physical random numbers *International Workshop on Fast Software Encryption* (Springer) pp 137–152

[35] Kwok S H, Ee Y L, Chew G, Zheng K, Khoo K and Tan C H 2011 A comparison of post-processing techniques for biased random number generators *IFIP International Workshop on Information Security Theory and Practices* (Springer) pp 175–190

[36] Lacharme P 2008 Post-processing functions for a biased physical random number generator *International Workshop on Fast Software Encryption* (Springer) pp 334–342

[37] Hirose S 2004 *Investigation Reports on Cryptographic Techniques*

[38] Hallgren S 1994 *Linear congruential generators over elliptic curves* (Carnegie-Mellon University. Department of Computer Science)

[39] Beelen P and Doumen J 2002 Pseudorandom sequences from elliptic curves *Finite Fields with Applications to Coding Theory, Cryptography and Related Areas: Proceedings of the Sixth International Conference on Finite Fields and Applications, held at Oaxaca, México, May 21–25, 2001* (Springer) pp 37–52

[40] Shparlinski I E 2008 Pseudorandom points on elliptic curves over finite fields *Algebraic Geometry And Its Applications: Dedicated to Gilles Lachaud on His 60th Birthday* (World Scientific) pp 116–134

[41] Gutierrez J 2022 *Cryptography and Communications* **14** 505–525

[42] Hochreiter S and Schmidhuber J 1997 *Neural computation* **9** 1735–1780

[43] Lipton Z C, Berkowitz J and Elkan C 2015 *arXiv preprint arXiv:1506.00019*

[44] Talbot M 2012 *A Romp Through Ethics for Complete Beginners* (University of Oxford) URL `https://www.courses.com/university-of-oxford/a-romp-through-ethics-for-complete-beginners/1`