



# On the effects of biased quantum random numbers on the initialization of artificial neural networks

Raoul Heese<sup>1</sup>  · Moritz Wolter<sup>2</sup> · Sascha Mücke<sup>3</sup> · Lukas Franken<sup>4</sup> ·  
Nico Piatkowski<sup>4</sup> 

Received: 31 May 2021 / Revised: 11 August 2023 / Accepted: 23 November 2023 /

Published online: 16 January 2024

© The Author(s) 2024

## Abstract

Recent advances in practical quantum computing have led to a variety of cloud-based quantum computing platforms that allow researchers to evaluate their algorithms on noisy intermediate-scale quantum devices. A common property of quantum computers is that they can exhibit instances of true randomness as opposed to pseudo-randomness obtained from classical systems. Investigating the effects of such true quantum randomness in the context of machine learning is appealing, and recent results vaguely suggest that benefits can indeed be achieved from the use of quantum random numbers. To shed some more light on this topic, we empirically study the effects of hardware-biased quantum random numbers on the initialization of artificial neural network weights in numerical experiments. We find no statistically significant difference in comparison with unbiased quantum random numbers as well as biased and unbiased random numbers from a classical pseudo-random number generator. The quantum random numbers for our experiments are obtained from real quantum hardware.

**Keywords** Quantum computing · Random number generation · Neural networks · Machine learning

---

Editor: Barbara Hammer.

---

✉ Raoul Heese  
raoul.heese@gmail.com

<sup>1</sup> Fraunhofer Center for Machine Learning and Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

<sup>2</sup> Fraunhofer Center for Machine Learning and Fraunhofer Institute for Algorithms and Scientific Computing SCAI, Schloss Birlinghoven, Konrad-Adenauer-Straße, 53757 Sankt Augustin, Germany

<sup>3</sup> Artificial Intelligence Group, TU Dortmund University, Otto-Hahn-Straße 12, 44227 Dortmund, Germany

<sup>4</sup> Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven, Konrad-Adenauer-Straße, 53757 Sankt Augustin, Germany

## 1 Introduction

The intrinsic non-deterministic nature of quantum mechanics (Kofler & Zeilinger, 2010) makes random number generation a native application of quantum computers. It has been exemplarily studied in Bird et al. (2020) how such quantum random numbers can affect stochastic machine learning algorithms. For this purpose, electron-based superposition states have been prepared and measured on quantum hardware to create random 32-bit integers. These numbers have subsequently been used to initialize the weights in neural network models and to determine random splits in decision trees and random forests. Bird et al. have observed that quantum random numbers can lead to superior results for certain numerical experiments in comparison with classically.<sup>1</sup> generated pseudo-random numbers.

However, the authors have not further explained this behavior. In particular, they have not discussed the statistical properties of the generated quantum numbers. Due to technical imperfections and physical phenomena like decoherence and dissipation, measurement results from a quantum computer might in fact significantly deviate from idealized theoretical predictions (Tamura & Shikano, 2020; Shikano et al., 2020; Tamura & Shikano, 2021). This raises the question of whether it is not the superiority of the quantum random number generator to sample perfectly random from the uniform distribution that leads to the observed effect, but instead its ability to sample bit strings from a very particular distribution that is imposed by the quantum hardware.

We therefore revisit this topic in the present manuscript and generate biased random numbers using real quantum hardware, where the specifics of the bias are determined by the natural imperfections of the hardware itself. The bias is therefore not under our control and even beyond our full understanding. With this approach, we aim to better comprehend the effects observed by Bird et al. for an analogous setup and explore the resulting implications. Summarized, our main goal is to further study the results of that work and to analyze the effects of quantum and classical random numbers with and without biases on neural network initialization. Our analysis is mainly based on numerical experiments and statistical tests.

The structure of the remaining paper is as follows. In Sect. 2, we briefly summarize the background of the main ingredients of our work, namely quantum computing and random number generation. Subsequently, we present the setup of our quantum random number generator and discuss the statistics of its results in Sect. 3. In Sect. 4, we study the effects of the generated quantum random numbers on artificial neural network weight initialization using numerical experiments. Finally, we close with a conclusion.

## 2 Background

In the following, we provide a brief introduction to quantum computing and random number generation without claiming to be exhaustive. For more in-depth explanations, we refer to the cited literature.

<sup>1</sup> We use the term “classical” in the sense of the physics community to distinguish deterministically behaving entities from the realm of classical physics from those governed by the non-deterministic rules of quantum physics (Norsen, 2017).

## 2.1 Quantum computing

Quantum mechanics is a physical theory that describes objects at the scale of atoms and subatomic particles, e. g., electrons and photons (Norsen, 2017). An important interdisciplinary subfield is quantum information science, which considers the interplay of information science with quantum effects and includes the research direction of quantum computing (Nielsen & Chuang, 2011).

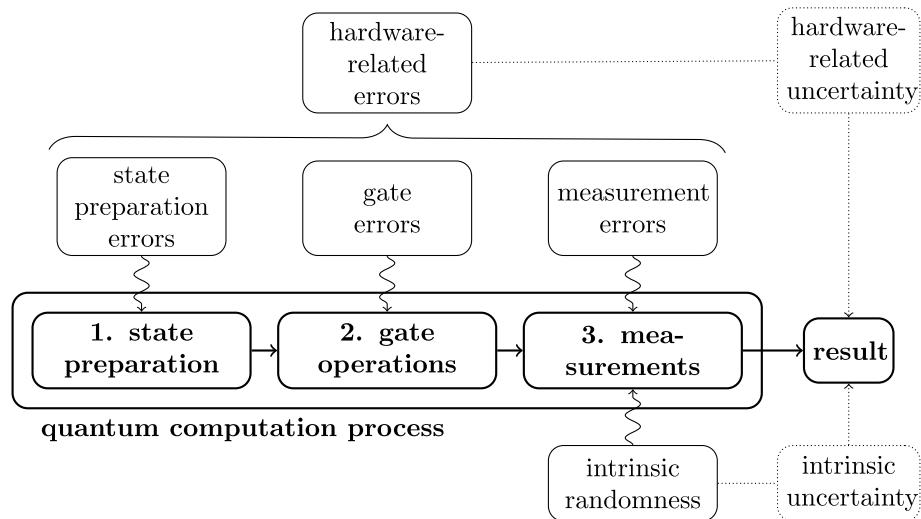
### 2.1.1 Quantum devices

A quantum computer is a processor which utilizes quantum mechanical phenomena to process information (Benioff, 1980; Grumblin & Horowitz, 2019). Theoretical studies show that quantum computers are able to solve certain computational problems significantly faster than classical computers, for example, in the fields of cryptography (Pirandola et al., 2020) and quantum simulations (Georgescu et al., 2014). Recently, different hardware solutions for quantum computers have been realized and are steadily improved. For example, superconducting devices (Huang et al., 2020) and ion traps (Bruzewicz et al., 2019) have been successfully used to perform quantum computations. However, various technical challenges are still unresolved so that the current state of technology, which is subject to substantial limitations, is also phrased as *noisy intermediate-scale quantum* (NISQ) computing (Preskill, 2018). Nevertheless, quantum supremacy on NISQ devices has already been verified experimentally for a specialized task of randomized sampling (Boixo et al., 2018; Wu et al., 2021).

There are different theoretical models to describe quantum computers, typically used for specific hardware or in different contexts. We only consider the quantum circuit model, in which a computation is considered as a sequence of *quantum gates* and the quantum computer can consequently be seen as a *quantum circuit* (Nielsen & Chuang, 2011). In contrast to a classical computer, which operates on electronic bits with a well-defined binary state of either 0 or 1, a quantum circuit works with *qubits*. A qubit is described by a quantum mechanical state, which can represent a binary 0 or 1 in analogy to a classical bit. In addition, however, it can also represent any *superposition* of these two values. Such a quantum superposition is a fundamental principle of quantum mechanics and cannot be explained with classical physical models. Moreover, two or more qubits can be *entangled* with each other. Entanglement is also a fundamental principle of quantum mechanics and leads to non-classical correlations (Bell & Aspect, 2004).

In order to illustrate the aforementioned fundamental quantum principles and to connect them with well-known notions from the field of machine learning, one can consider the following intuitive (but physically inaccurate) simplifications: Superposition states can be understood as probability distributions over a finite state space, while entanglement amounts to high-order dependencies between univariate random variables. This intuition particularly emphasizes the close relationship between quantum mechanics and probability theory.

Any quantum computation can be considered as a three-step process, which is sketched in Fig. 1. First, an initial quantum state of the qubits is prepared, usually a low-energy ground state. Second, a sequence of quantum gates deterministically transforms the initial state into a final quantum state. Third, a measurement is performed on the qubits to determine an outcome. When a qubit is measured, the result of the



**Fig. 1** Sketch of the three-step quantum computation process consisting of an initial state preparation, a sequence of gate operations and a final measurement, which yields the result of the computation. Also shown are the errors associated with each step in the computation process: the state preparation errors, the gate errors, and the measurement errors, respectively. They are all hardware-related errors, which can in principle be reduced (or even eliminated) by technological advances. These errors can cause a hardware-related uncertainty (statistical and systematic) of the computation result. On the other hand, the intrinsic randomness of quantum mechanics emerging at the time of the measurement causes an intrinsic uncertainty of the computation result, which is an integral part of quantum computing and can be exploited to construct QRNGs

measurement is always either 0 or 1, but the observation is non-deterministic with a probability depending on the quantum state of the qubit at the time of the measurement.

In this sense, a quantum computation includes an intrinsic element of randomness. This randomness is in particular not a consequence of lack of knowledge about the quantum system, but an integral part of quantum mechanics itself. In contrast to classical mechanics, where complete knowledge about the initial state of a system allows to infer all later (and earlier) states, complete knowledge about a quantum mechanical state does not generally allow the prediction of a single measurement outcome, but only its probability as determined by Born's rule (Norsen, 2017). The non-deterministic nature of quantum mechanics relies on the assumption that there are no so-called *hidden variables* whose knowledge would lead to a deterministic behavior (Norsen, 2017). Various theoretic and experimental evidences, for example based on Bell's theorem (Bell & Aspect, 2004) or the Kochen-Specker theorem (Kochen & Specker, 1975), strongly suggest that there are no such hidden variables. However, a conclusive answer to the question of quantum non-determinism is still in scientific discourse. For a more detailed discussion about this topic, we refer to Bera et al. (2017) and references therein. Since our work concerns the practical application of random numbers in machine learning algorithms and a theoretical provability of their randomness from first principles is beyond the scope of this paper, we presume in the following that quantum mechanics is indeed intrinsically non-deterministic for all purposes considered.

NISQ devices, as their name suggests, are typically only capable of computing noisy results. A fundamental reason is that the quantum computer, despite all technical efforts, is not perfectly isolated and interacts (weakly) with its environment. In particular, there are two major effects of the environment that can contribute to computational errors, namely dissipation and decoherence in the sense of dephasing (Zurek, 2007; Vacchini, 2016). Dissipation describes the decay of qubit states of higher energy due to an energy exchange with the environment. Decoherence, on the other hand, represents a loss of quantum superpositions as a consequence of environmental interactions. Typically, decoherence is more dominating than dissipation. Beyond these typical effects, other (possibly unknown) influences can occur, which can lead to additional uncertainties.

To compensate the resulting computational errors to a certain extend, error correction can be used (Roffe, 2019). However, it is generally not possible to completely eliminate statistical (also called *aleatoric*) or systematic (also called *epistemic*) uncertainties, which might originate from quantum and classical effects, respectively. Therefore, quantum algorithms must be designed sufficiently robust for practical applications on NISQ hardware.

In Fig. 1, we briefly outline different error sources in the quantum computation process. Specifically, each computation step is affected by certain hardware-related errors, which are referred to as state preparation errors, gate errors, and measurement errors, respectively (Nachman & Geller, 2021). All of them are a consequence of the imperfect physical hardware and they are non-negligible for NISQ devices (Leymann & Barzen, 2020). The resulting hardware-related uncertainty might be both statistical and systematic. In addition, the final measurement step is also affected by the intrinsic randomness of quantum mechanics. The measurement ultimately yields a computation result that contains two layers of uncertainty (Heese & Freyberger, 2014): First, the uncertainty caused by the hardware-related errors, and second, the uncertainty caused by the intrinsic randomness. While technological advances (like better hardware and improved algorithm design) can in principle reduce (or even eliminate) hardware-related errors and thus the hardware-related uncertainty, the intrinsic uncertainty is an integral part of quantum computing. It is this intrinsic uncertainty which can be exploited to construct QRNGs.

### 2.1.2 Quantum machine learning

In a machine learning context, we may identify a quantum circuit with a parameterizable probability distribution over all possible measurement outcomes, where each measurement of the circuit draws a sample from this distribution. The interface between quantum mechanics and machine learning can be attributed to the field of *quantum machine learning* (Biamonte et al., 2017). A typical use case is the processing of classical data using algorithms that are fully or partially computed with quantum circuits, which is also called *quantum-enhanced machine learning* (Dunjko et al., 2016).

The noisy nature of NISQ devices presents a challenge for machine learning applications. On the other hand, the probabilistic nature of quantum computing can be related to the statistical background of machine learning algorithms, for which the understanding and modeling of uncertainty is crucial. A review about different types of uncertainty in machine learning and how to typically deal with them can for example be found in Hüllermeier and Waegeman (2021).

## 2.2 Random number generation

For many machine learning methods, random numbers are a crucial ingredient and therefore random number generators (RNGs) are an important tool. Examples include sampling from generative models like generative adversarial networks, variational autoencoders or Markov random fields, parameter estimation via stochastic optimization methods, as well as randomized regularization and validation techniques, randomly splitting for cross-validation, drawing of random mini-batches, and computing a stochastic gradient, to name a few. Randomness also plays an important role in non-deterministic optimization algorithms or the initialization of (trainable) neural network parameters (Glorot & Bengio, 2010; He et al., 2015).

At its core, a RNG performs random coin tosses in the sense that it samples from a uniform distribution over a binary state space (or, more generally, a discrete state space of arbitrary size). Given a sequence of randomly generated bits, corresponding integer or floating-point values can be constructed straightforwardly.

### 2.2.1 Classical RNGs

In the classical world, there are two main types of random number generators. Pseudo-random number generators (PRNGs) represent a class of algorithms to generate a sequence of apparently random (but in fact deterministic) numbers from a given *seed* (James & Monta, 2020). In other words, the seed fully determines the order of the bits in the generated sequence, but the statistical properties of the sequence (e. g., mean and variance) are independent of the seed (as determined by the underlying algorithm). We remark that PRNGs can also be constructed based on machine learning algorithms (Pasqualini & Parton, 2020).

The more advanced true random number generators (TRNGs) are hardware devices that receive a signal from a complex physical process, which is unpredictable for all practical purposes, to extract random numbers (Yu et al., 2019). A multitude of physical effects can be used as sources of entropy for TRNGs, with only some of them directly linked to quantum phenomena. For example, metastability in latches can be exploited in specialized electrical circuits (CMOS devices) to yield random bits (Tokunaga et al., 2008; Holleman et al., 2008). Usually, such setups are built to calibrate themselves to account for hardware-inherent bias effects. Multiple of these self-calibrating entropy sources can be combined to further increase the cryptographic quality (Mathew et al., 2015). Other approaches make use of ring oscillators to source randomness from timing jitter (Kim et al., 2017), or exploit random telegraph noise to produce bit streams (Puglisi et al., 2018; Brown et al., 2020).

For TRNGs, the lack of knowledge about the observed physical system induces randomness, but it cannot be guaranteed in principle that the dynamics of the underlying physical system are unpredictable (if quantum effects are not sufficiently involved). Likewise, the statistical properties of the generated random sequence are not in principle guaranteed to be constant over time since they are subject to the hidden process.

Independent of their source, random numbers have to fulfill two properties: First, they have to be truly random (i. e., the next random bit in the sequence must not be predictable from the previous bits) and second, they have to be unbiased (i. e., the statistics of the random bit sequence must correspond to the statistics of the underlying uniform distribution). In other words, they have to be secure and reliable. A “good” RNG has to produce numbers that fulfill both requirements. In practice, it is difficult to rigorously proof the

quality of RNGs. For a bit sequence of finite length, there is no formal method to decide its randomness with certainty. On the other hand, an infinite bit sequence cannot be tested in finite time (Khrennikov, 2015). Therefore, statistical test are typically used to check specific properties of RNGs with a certain confidence.

Typically, statistical tests are organized in the form of test suites (e. g., the *NIST Statistical Test Suite* described in Rukhin et al., 2010) to provide a comprehensive statistical screening. A predictive analysis based on machine learning methods can also be used for a quality assessment (Li et al., 2020). It remains a challenge to certify classical RNGs in terms of the aforementioned criteria (Balasch et al., 2018) to, e. g., ensure cryptographical security.

When implementing learning and related algorithms, PRNGs are typically used. Despite the broad application of randomness in machine learning, the apparent lack of research regarding the particular choice of RNGs suggests that it is usually not crucial in practice. This assumption has been experimentally verified, e. g., in Rajashekharan and Shummuva Velayutham (2016) for differential evolution and is most certainly due to the fact that modern PRNGs seem to be sufficiently secure and reliable for most practical purposes. The influence of different seeds for a PRNG on various deep learning algorithms for computer vision has been studied empirically in Picard (2021) with the result that it is often possible to find seeds that lead to a much better or much worse performance than the average. This highlights the fact that numerical experiments with non-deterministic algorithms have to be conducted carefully to account for the variance of random numbers. However, the specific implications of varying degrees of security and reliability of RNGs on machine learning applications generally remain unresolved, i. e., it generally remains unclear whether a certain machine learning algorithm may suffer or benefit from the artifacts of an imperfect RNG. In the present work, we approach this still rather open field of research by specifically considering the randomness in artificial neural network initialization.

## 2.2.2 Quantum RNGs

As previously stated, quantum computers (or, more generally, quantum systems) have an intrinsic ability to produce truly random outcomes in a way that cannot be predicted or emulated by any classical device (Calude et al., 2010). Therefore, it seems natural to utilize them as a source of random numbers in the sense of a quantum random number generator (QRNG). Such QRNGs (Herrero-Collantes & Garcia-Escartin, 2017) have already been realized with different quantum systems, for example using nuclear decay (Park et al., 2020) or optical devices (Leone et al., 2020).

Summarized, the main difference between randomness from classical systems and randomness from quantum systems is that a classical system is fully deterministic and therefore all randomness can only result from a lack of knowledge about the system, whereas a quantum system is non-deterministic and therefore – even with perfect knowledge – an intrinsic randomness may be involved. In this sense, the origin of randomness is different for quantum and classical RNGs. However, it is in principle not possible to mathematically distinguish the randomness of a classical system from the randomness of a quantum system (Khrennikov, 2015).

A simple QRNG can be straightforwardly realized using a quantum circuit. For this purpose, each of its qubits has to be brought into a superposition of 0 and 1 such that both outcomes are equally probable to be measured. This operation can for example be performed by applying a single Hadamard gate on each qubit (Nielsen & Chuang, 2011). Each

measurement of the circuit consequently generates a sequence of i.i.d. random bits, one for each qubit.

However, when computing this simple QRNG circuit on a NISQ device, it can be expected that the results will deviate from the theoretic expectations due to statistical and systematic uncertainties such that the QRNG is likely to produce biased outcomes. This means that it is in fact not guaranteed that the measurement outcomes obey the theoretically predicted probability distribution of a fair coin toss. It is not even guaranteed that the measurement outcomes are truly random in the sense that bits are generated entirely independent. As a consequence (and based on the fact that quantum non-determinism is not ultimately resolved), it cannot be generally taken for granted that random numbers from such a QRNG are naturally “better” than random numbers from PRNGs, both with respect to security and reliability. For this reason, technically more refined solutions are necessary to realize trustworthy QRNGs on NISQ devices. Moreover, QRNGs have to be certified similar to classical RNGs. For example, to enable a theoretically secure QRNG, the Bell inequality (Pironio et al., 2010) or the Kochen-Specker theorem can be utilized (Abbott et al., 2014, 2015; Kulikov et al., 2017). For an experimental verification of random bit sequences from a QRNG, entanglement-based public tests of randomness can be used without violating the secrecy of the generated sequences (Jacak et al., 2020).

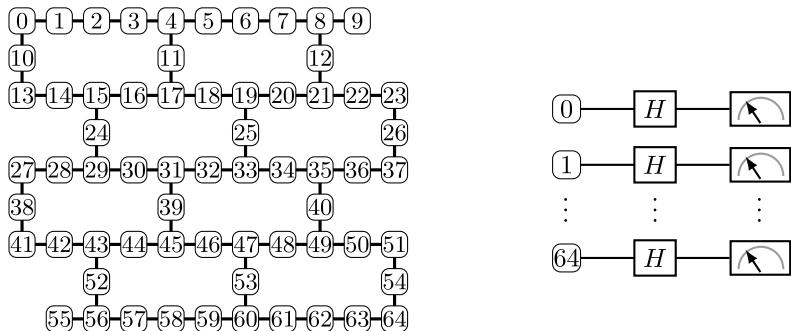
Currently, there exist various commercial and non-commercial QRNGs, which can be used to create quantum random numbers on demand, for example ANU (2021). Although there still seem to be some practical challenges (Martínez et al., 2018; Petrov et al., 2020), theoretical and technological advances in the field will most certainly lead to a steady improvement of QRNGs.

### 3 Biased QRNG

Motivated by the work in Bird et al. (2020), we take a different approach than usual in this manuscript. Instead of aiming for a RNG with as little bias as possible, we discuss whether the typical bias in a naively implemented, gate-based QRNG can actually be beneficial for certain machine learning applications. In other words, we consider the bias that is “naturally” imposed by the quantum hardware itself (i. e., by the hardware-related errors outlined in Fig. 1). In addition to a bias, we also accept that the randomness of the results is not necessarily guaranteed in the sense that the QRNG can (to some degree) produce correlations or predictable patterns from systematic quantum hardware errors. Since the imperfections of the quantum hardware are beyond our control (i. e., they can in particular not be switched off at will), a RNG realized in this way contains unknown and uncontrollable elements. Therefore, we have to analyze its outcomes statistically to capture the effects of these elements on the generated random numbers. In the present section, we first describe our experimental setup for such a naively implemented QRNG and subsequently discuss the statistics of the resulting “hardware-biased” quantum random numbers.

#### 3.1 Setup

To realize a hardware-biased QRNG (B-QRNG), we utilize a physical quantum computer, which we access remotely via Qiskit (Abraham et al., 2019) using the cloud-based quantum computing service provided by IBM Quantum (IBM, 2021). With this service, users can send online requests for quantum experiments using a high-level quantum circuit model of



(a) Topology diagram of the *ibmq\_manhattan* backend with 65 qubits. Qubits are shown as boxes with their respective hardware index  $n$ . Pairs of qubits that support two-qubit gate operations (which are not used in our setup) are connected with lines.

(b) Circuit diagram. A single Hadamard gate (denoted by  $H$ ) is applied to each of the 65 qubits from (a) and a subsequent measurement is performed. The idealized measurement result of each qubit is either 0 or 1 with an equal probability of 50%.

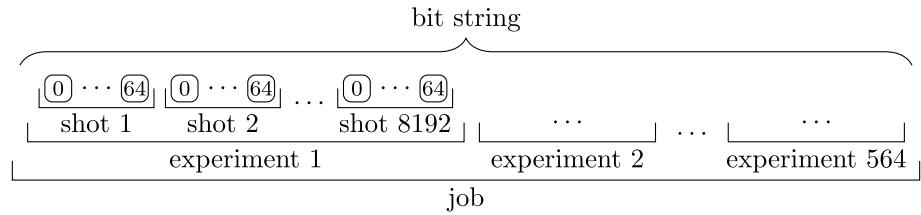
**Fig. 2** Main components of our B-QRNG setup: **a** topology diagram of the backend and **b** circuit diagram

computation, which are then executed sequentially (LaRose, 2019). The respective quantum hardware, also called *backend*, operates on superconducting transmon qubits.

For our application, we specifically use the *ibmq\_manhattan* backend (version 1.11.1), which is one of the IBM quantum *Hummingbird r2* processors with  $N \equiv 65$  qubits. A sketch of the backend topology diagram can be found in Fig. 2a. It indicates the *hardware index* of each qubit and the pairs of qubits that support two-qubit gate operations between them. IBM also provides an estimate for the relaxation time  $T_1$  and the dephasing time  $T_2$  for each qubit at the time of operation. The mean and standard deviation of these times over all qubits read  $T_1 \approx 59.11 + -15.25 \mu\text{s}$  and  $T_2 \approx 74.71 + -31.22 \mu\text{s}$ , respectively.

Initially, all qubits in this backend are prepared in the ground state. Our B-QRNG circuit, which is sketched in Fig. 2b, consists of one Hadamard gate applied to each qubit such that it is brought into a balanced superposition of ground state and excited state. A subsequent measurement on each qubit should therefore ideally (i. e., in the error-free case) reveal an outcome of either 0 (corresponding to the ground state) or 1 (corresponding to the excited state) with equal probability. However, since we run the circuit on real quantum hardware, we can expect to obtain random numbers which deviate from these idealized outcomes due to hardware-related errors. An analogous setup with a different backend is considered in Tamura and Shikano (2020); Shikano et al. (2020); Tamura and Shikano (2021).

We sort the qubit measurements according to their respective hardware index in an ascending order so that each run of the backend yields a well-defined bit string of length  $N$ . Such a single run is called a *shot* in Qiskit. We perform sequences of  $S \equiv 8192$  shots (which is the upper limit according to the backend access restrictions imposed by IBM) for which we concatenate the resulting bit strings in the order in which they are executed. Such a sequence of shots is called *experiment* in Qiskit. We repeat this experiment  $R \equiv 564$  times (900 experiments is the upper limit set by IBM) and again concatenate the resulting bit strings in the order of execution. A sequence of experiments is denoted as a *job* in Qiskit and can be submitted directly to the backend. It is run in one pass without interruption from other jobs.



**Fig. 3** Bit string composition from our B-QRNG. A single job is submitted to the backend, it consists of 564 experiments. In each experiment, 8192 shots are performed. In each shot, each of the 65 qubits yields a single bit. The resulting bit string consequently contains 300318720 bits

Our submitted job ran from March 5, 2021 10:45 AM GMT to March 5, 2021 11:58 AM GMT. The final result of the job is a bit string of length  $M \equiv NSR = 300318720$  as sketched in Fig. 3. The choice of  $R$  is determined by the condition  $M \gtrsim 3e8$ , which we have estimated as sufficient for our numerical experiments. We split the bit string into chunks of length  $C \equiv 32$  to obtain  $L \equiv M/C = 9384960$  random 32-bit integers, which we use for the following machine learning experiments.

## 3.2 Statistics

Before we utilize our generated random numbers for learning algorithms, we first briefly discuss their statistics. The measurement results from the  $n$ th qubit can be considered as a Bernoulli random variable (Forbes et al., 2011), where  $n \in \{0, \dots, 64\}$  represents the hardware index as outlined in Fig. 2. Such a variable has a probability mass function

$$f(b;p) \equiv p^b (1-p)^{1-b} \quad (1)$$

depending on the value of the bit  $b \in \mathbb{B}$  and the success probability  $p \in [0, 1]$  of observing an outcome  $b = 1$ .

### 3.2.1 Bias

We denote the measured bit string from our B-QRNG as a vector  $\mathbf{B} \in \mathbb{B}^M$ . The extracted bit string exclusively resulting from measurements of the  $n$ th qubit is given by the vector

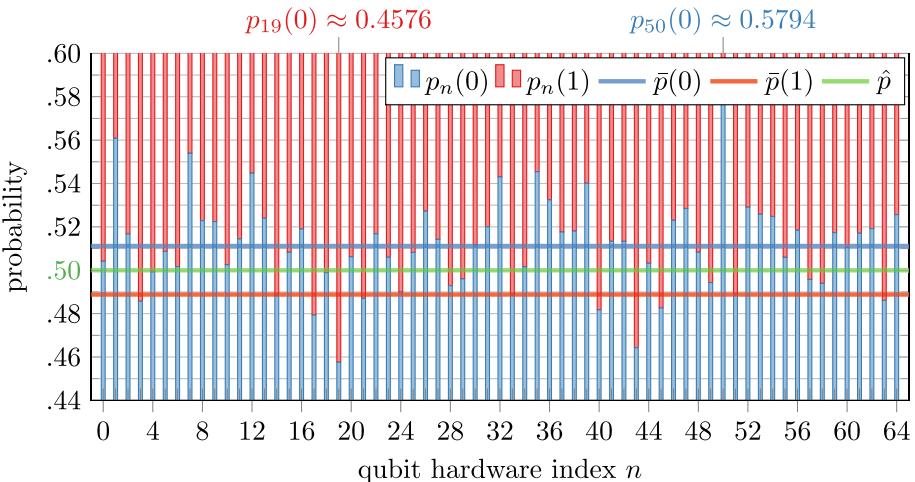
$$\mathbf{b}_n \equiv (B_{n+1}, B_{n+1+N}, \dots, B_{n+1+M-N}) \quad (2)$$

with  $\mathbf{b}_n \in \mathbb{B}^{M/N}$ . Based on its population, the corresponding expected probability  $p_n(0)$  of obtaining the bit  $b$  for the  $n$ th qubit is given by

$$p_n(b) = \frac{N \sum_{i=1}^{M/N} \mathbb{1}_n(i, b)}{M} \quad (3)$$

with the indicator function

$$\mathbb{1}_n(i, b) \equiv \begin{cases} 1 & \text{if } B_{n+(i-1)N+1} = b \\ 0 & \text{otherwise} \end{cases} \quad (4)$$



**Fig. 4** Measured bit distribution for each qubit from the B-QRNG on *ibmq\_manhattan*. We show the expected probability  $p_n(0)$  of obtaining a zero bit from the measured bit string for the  $n$ th qubit, Eq. (3), and (stacked on top) its complement  $p_n(1) = 1 - p_n(0)$ . Also shown are the corresponding expected probabilities with respect to all measured bits  $\bar{p}(0) \approx 0.51$  and  $\bar{p}(1) = 1 - \bar{p}(0) \approx 0.49$ , respectively, Eq. (6). Apparently, all bit distributions deviate differently from the uniform probability  $\bar{p}$ , Eq. (5), which we assume to be a consequence of the imperfect hardware. The distributions with the highest ( $n = 50$ ) and lowest ( $n = 19$ ) expected probabilities of obtaining a zero bit are marked on top

such that  $p_n(0) + p_n(1) = 1$ . From an idealized prediction of the measurement results of qubits in a balanced superposition, we would assume that all expected probabilities  $p_0(b), \dots, p_N(b)$  correspond to the uniform probability

$$\bar{p} \equiv \bar{p}(b) \equiv \frac{1}{2} \quad (5)$$

with uncertainties coming only from the finite number of samples.

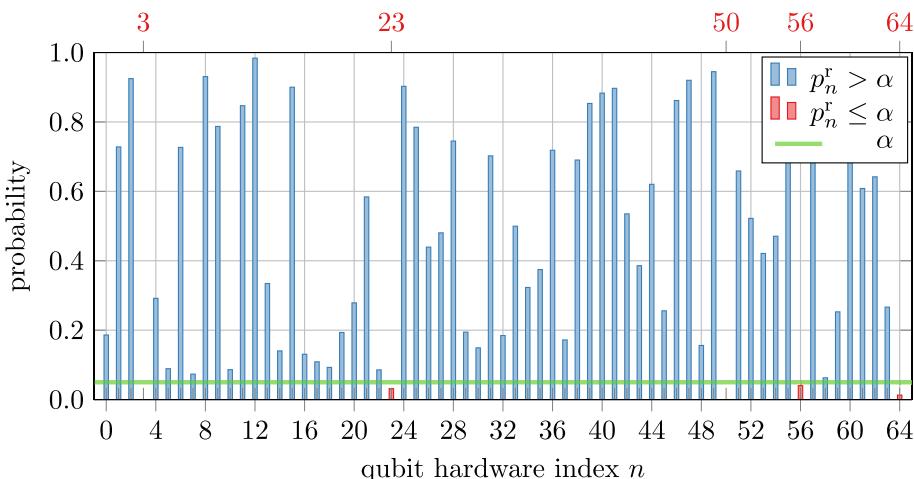
We show the estimated probabilities in Fig. 4. It is apparent that all bit probabilities deviate significantly from their idealized value  $\bar{p}$ , Eq. (5). In particular, we find an expected probability and standard deviation with respect to all measured bits of

$$\bar{p}(0) \approx 0.5112 \pm 0.0215. \quad (6)$$

We assume that this is a consequence of the imperfect hardware with its decoherence and dissipation effects. In particular, the fact that  $\bar{p}(0) > \bar{p}(1)$  is most likely a consequence of dissipation since a bit of 0 corresponds to an observation of a qubit ground state, whereas a bit of 1 is associated with an excited state.

From a  $\chi^2$  test (Pearson, 1900) on the measured bit distribution, the null hypothesis of a uniform zero bit occurrence can be rejected as expected with a confidence level of 1.0000. To further quantify the deviation of the measured probabilities from a uniform distribution, we utilize the discrete Hellinger distance (Hellinger, 1909)

$$H(q_1, q_2) \equiv \frac{1}{\sqrt{2}} \sqrt{\sum_{i \in Q} \left( \sqrt{q_1(i)} - \sqrt{q_2(i)} \right)^2}, \quad (7)$$



**Fig. 5** Results of Wald-Wolfowitz runs test on the bit strings of all qubits, where  $p_n^r$  denotes the resulting p-value of the bit string of the  $n$ th qubit  $\mathbf{b}_n$ , Eq. (2). We show p-values in different colors depending on whether or not they exceed  $\alpha = 0.05$ . In case of  $p_n^r \leq \alpha$ , the corresponding hardware indices are additionally denoted on top of the plot and indicate the qubits that fail the test of randomness

which can be used to measure similarities between two discrete probability distributions  $q_1 \equiv q_1(i)$  and  $q_2 \equiv q_2(i)$  defined on the same probability space  $Q$ . By iterating over all qubits we find the mean and standard deviation

$$\langle H(p_n, \tilde{p}) \rangle \approx 0.0133 \pm 0.0110. \quad (8)$$

The mean value quantifies the average deviation of the measured qubit distributions from the idealized uniform distribution and confirms our qualitative observations. The non-negligible standard deviation results from the fluctuations in-between the individual qubit outcomes.

### 3.2.2 Randomness

Although quantum events intrinsically exhibit a truly random behavior, the output of our B-QRNG is the result of a complex physical experiment behind a technically sophisticated pipeline that appears as a black box to us and it can therefore not be assumed with certainty that its outcomes are indeed statistically independent. To examine this issue in more detail, we briefly study the randomness of the resulting bit string in the following.

For this purpose, we make use of the Wald-Wolfowitz runs test (Wald & Wolfowitz, 1940), which can be used to test the null hypothesis that elements of a binary sequence are mutually independent. We perform a corresponding test on the measured bit string from the  $n$ th qubit  $\mathbf{b}_n$ , Eq. (2), and denote the resulting p-value as  $p_n^r$ . The null hypothesis has to be rejected if this probability does not exceed the significance level, which we choose as  $\alpha = 0.05$ .

The test results are shown in Fig. 5. We find that the bit strings from almost all qubits pass the test and can therefore be considered random in the sense of the test criteria. However, the bit strings from five qubits fail the test, which implies non-randomness. We also

perform a test on the total bit string  $\mathbf{B}$ , which yields the p-value  $p^r \approx 0.0000 < \alpha$  such that the test also fails for the entire sequence of random numbers.

Summarized, we find that the reliability of the generated quantum random numbers is questionable. A typical binary random sequence from a PRNG of the same length as  $\mathbf{B}$  can be expected to pass the Wald-Wolfowitz runs test. However, within the scope of this work, the reason for this observation cannot be further investigated and we accept it as an integral part of our naive approach to the B-QRNG. Further work regarding the properties of our setup (applied to a different quantum hardware) can be found in Tamura and Shikano (2020); Shikano et al. (2020); Tamura and Shikano (2021), which contain similar observations. A lack of reliability is not surprising considering the fact that we have not aimed for a certified random number generation and our setup is motivated by a strongly idealized model of quantum gate computers, as already mentioned above.

### 3.2.3 Integers

Next, we analyze the resulting random 32-bit integers. To obtain these, we convert  $\mathbf{B}$  into a vector of integers  $\mathbf{B} \mapsto \mathbf{I} \in \{0, \dots, 2^C - 1\}^L$  by consecutively grouping its elements into bit strings of length  $C$  and converting them to non-negative integers according to

$$I_j \equiv \sum_{i=0}^{C-1} B_{C(j-1)+i+1} 2^i \quad (9)$$

with  $j \in \{1, \dots, L\}$ . For a bit string of Bernoulli random variables  $\mathbf{B}$  with a fair success probability  $p = \tilde{p}$ , Eqs. (1) and (5), the sequence of random integers in  $\mathbf{I}$  would be uniformly distributed. However, as we have seen before, this assumption does not hold true for the results from our B-QRNG. So the question arises as to what the distribution of random integers looks like for our unfair set of Bernoulli variables.

For this purpose, we rescale the elements of  $\mathbf{I}$  by a division by

$$\xi \equiv 2^C - 1 \quad (10)$$

such that  $\mathbf{I}/\xi \in [0, 1]^L$  and group the range  $[0, 1]$  into  $K \equiv 250$  equally sized bins. Thus, the population of the  $k$ th bin is given by

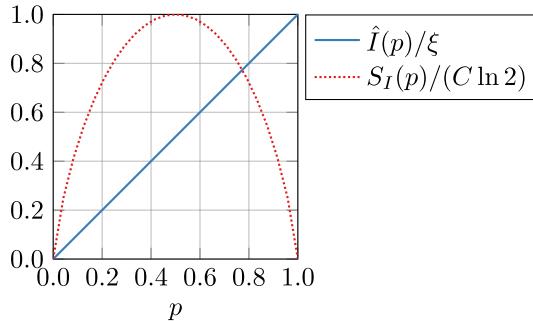
$$c_k \equiv \sum_{i=1}^L \mathbb{1}(I_i, k) \quad (11)$$

with the indicator function

$$\mathbb{1}(i, k) \equiv \begin{cases} 1 & \text{if } k < K \wedge \frac{k-1}{K} \leq \frac{i}{\xi} < \frac{k}{K} \\ 1 & \text{if } k = K \wedge \frac{K-1}{K} \leq \frac{i}{\xi} \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

for  $k \in \{1, \dots, K\}$ .

Additionally, we consider a simplified theoretical description of the bin population by modeling the bit string as the result of a Bernoulli process with a single success probability  $p$ , Eq. (1). That is, the bits represent i.i.d. Bernoulli random variables. The integer  $j \in \{0, \dots, \xi\}$  corresponding to a bit string  $\tau(j) \in \mathbb{B}^C$  is determined in analogy to Eq. (9)



**Fig. 6** Expected value  $\hat{I}(p)$ , Eq. (14), and entropy  $S_I(p)$ , Eq. (15), for a random integer from the domain  $\{0, \dots, \xi\}$  resulting from a string of  $C$  random bits from a Bernoulli process with success probability  $p$ , Eq. (1). The expected value is proportional to  $p$ , whereas the entropy attains its maximum value at  $p = 0.5$ . We apply rescaling factors to constrain both quantities to the same scale

such that  $\sum_{i=0}^{C-1} \tau_{i+1}(j)2^i = j$ . The probability mass function of the resulting integers can consequently be written as

$$P(j, p) \equiv \prod_{i=1}^C p^{\tau_i(j)} (1-p)^{1-\tau_i(j)}. \quad (13)$$

Its expected value is given by

$$\hat{I}(p) \equiv \sum_{i=0}^{\xi} i P(i, p) = \xi p \quad (14)$$

and the information entropy (Shannon, 1948) in *nats* by

$$S_I(p) \equiv \sum_{i=0}^{\xi} P(i, p) \ln P(i, p) = -C[p \ln p + (1-p) \ln(1-p)]. \quad (15)$$

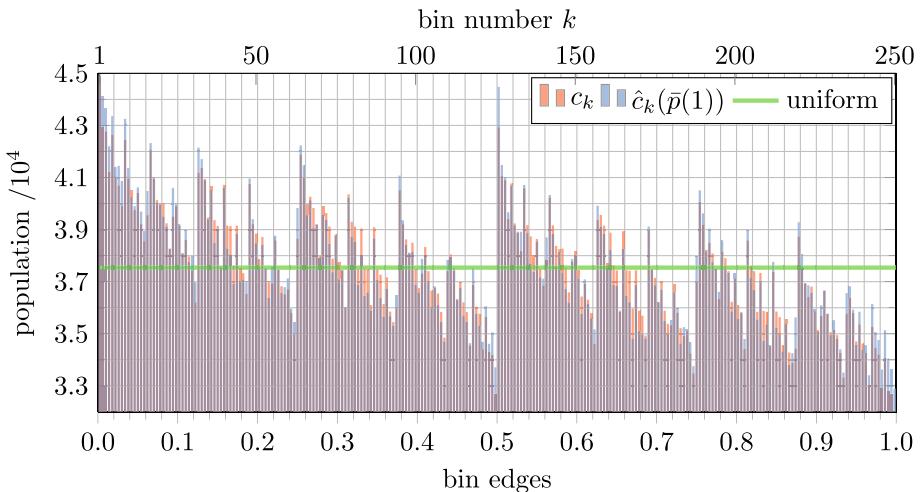
We show a plot of Eqs. (14) and (15) in Fig. 6. Finally, the predicted (possibly non-integer) population of the  $k$ th bin reads

$$\hat{c}_k(p) \equiv L \sum_{i=0}^{\xi} \mathbb{1}(i, k) P(i, p), \quad (16)$$

which we use as our simplified model of Eq. (11).

We show both the measured bin population  $c_k$ , Eq. (11), and the theoretical bin population  $\hat{c}_k(p)$ , Eq. (16), for a success probability  $p$  corresponding to the expected probability of all measured bits  $\bar{p}(1) = 1 - \bar{p}(0)$ , Eq. (6), in Fig. 7. Clearly, the generated sequence of random integers is not uniformly distributed (i. e., with a population of  $L/K$  in each bin). Instead, we find a complex arrangement of spikes and valleys in the bin populations.

Specifically, since  $\bar{p}(0) > \bar{p}(1)$ , random integers become more probable when their binary representation contains as many zeros as possible, which is reflected in the bin populations. In particular, the first bin (containing the smallest integers) has the highest population. The minor deviations between the measured and the theoretic bin populations results from the finite number of measured samples and the simplification of the theoretical



**Fig. 7** Measured distribution of 32-bit integers from the B-QRNG. The values from the generated vector of random integers  $\mathbf{I}$ , Eq. (9), are rescaled by a division by  $(2^{32} - 1)$  and sorted into 250 equally sized bins. The  $k$ th bin (with  $k \in \{1, \dots, 250\}$ ) has a population of  $c_k$  according to Eq. (11). For comparison, the corresponding theoretic bin population of the  $k$ th bin  $\hat{c}_k(\bar{p}(b))$  is shown, which is obtained from a Bernoulli process according to Eq. (16) with a success probability of  $p = \bar{p}(1) = 1 - \bar{p}(0)$ , Eq. (6). The minor deviations between the two populations results from the finite number of measured samples as well as the observation that bits from different qubits have their own success probability, cf. Fig. 4. An outline of the uniform bin population is shown as a frame of reference

model: The success probability of each bit from the B-QRNG specifically depends on the qubit it is generated from as shown in Fig. 4, whereas our theoretical model only uses one success probability for all bits corresponding to  $\bar{p}(1)$ .

We recall the Hellinger distance, Eq. (7), to quantify the deviation of the distribution of integers from the uniform distribution. Specifically, we find

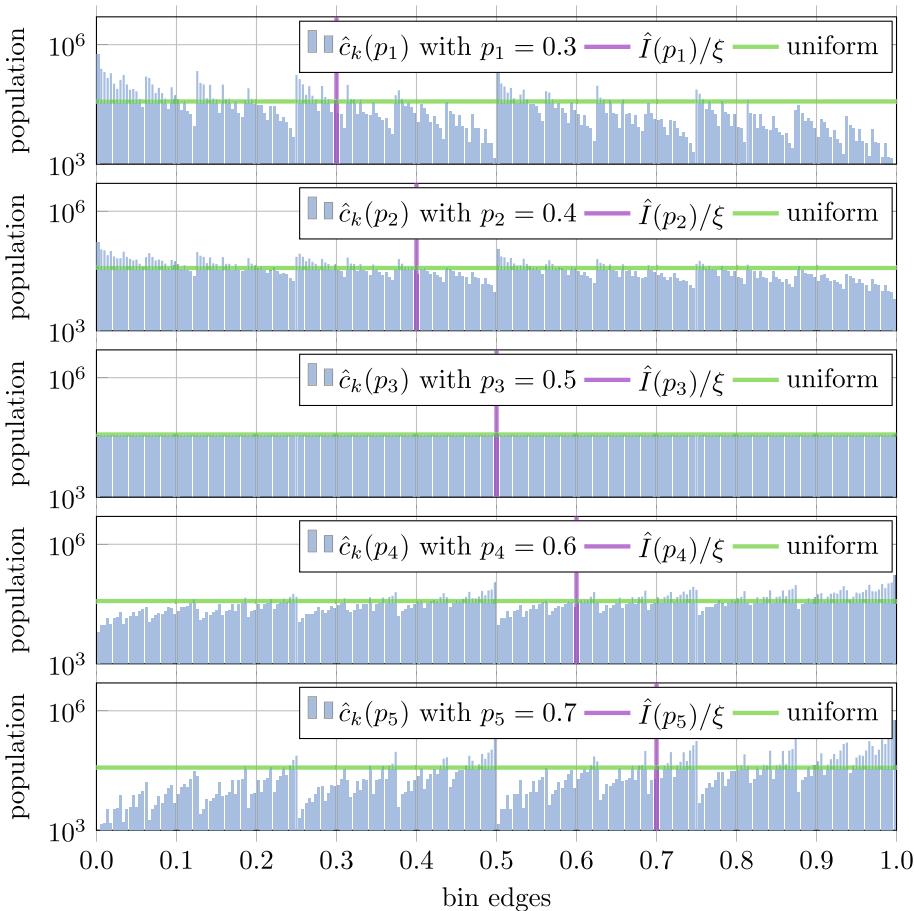
$$H(p_c, \tilde{p}_c) \approx 0.0213, \quad (17)$$

where we have made use of the measured integer distribution  $p_c \equiv p_c(k) \equiv c_k/L$  and the corresponding uniform distribution  $\tilde{p}_c \equiv \tilde{p}_c(k) \equiv 1/K$  with  $k \in \{1, \dots, K\}$ . This metric quantifies our observations from Fig. 7.

For comparative purposes, we show additional theoretical bin populations for other success probabilities in Fig. 8. As expected, the rugged pattern of the distribution becomes sharper for lower or higher values of  $p$  and the deviation from the uniform distribution increases.

## 4 Experiments

To study the effects of quantum-based network initializations, we consider two independent experiments, which are both implemented in *PyTorch* (Paszke et al., 2019): First, a convolutional neural network (CNN) and second, a recurrent neural network (RNN). The choice of these experiments is motivated by the statement from Bird et al.



**Fig. 8** Theoretical distribution of 32-bit integers in analogy to Fig. 4 for different success probabilities  $p \in \{p_1 = 0.3, p_2 = 0.4, p_3 = 0.5, p_4 = 0.6, p_5 = 0.7\}$ , Eq. (16). The population axis is scaled logarithmically. We also show the (rescaled) mean values  $\hat{I}(p)/\xi$ , Eq. (14), and the uniform distribution  $\tilde{p}_c$  as used in Eq. (17). The corresponding Hellinger distances, Eq. (7), with  $\hat{p}_c(p) \equiv \hat{p}_c(p; k) \equiv \hat{c}_k(p)/L$  and  $k \in \{1, \dots, K\}$  read  $H(\hat{p}_c(p_1), \tilde{p}_c) \approx H(\hat{p}_c(p_5), \tilde{p}_c) \approx 0.3776$ ,  $H(\hat{p}_c(p_2), \tilde{p}_c) \approx H(\hat{p}_c(p_4), \tilde{p}_c) \approx 0.1867$ , and  $H(\hat{p}_c(p_3), \tilde{p}_c) \approx 0.0000$ , respectively

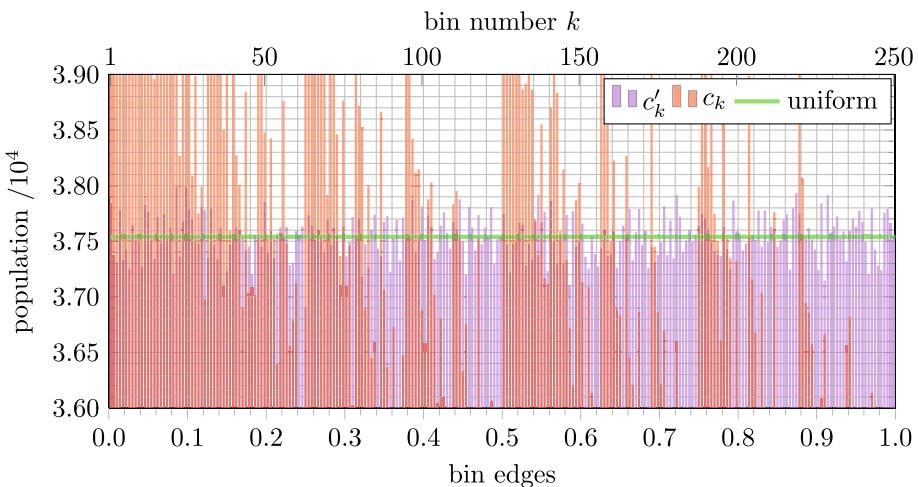
(2020) that “neural network experiments show greatly differing patterns in learning patterns and their overall results when using PRNG and QRNG methods to generate the initial weights.”

To ensure repeatability of our experiments, PyTorch is run in *deterministic mode* with fixed (i. e., hard-coded) random seeds. The main hardware component is a *Nvidia GeForce GTX 1080 Ti* graphics card. Our Python implementation of the experiments is publicly available online (Wolter, 2021).

In the present section, we first summarize the considered RNGs. Subsequently, we present the two experiments and discuss their results.

**Table 1** Overview over the four considered RNGs presented in Sect. 4.1, which are either based on a classical pseudo-random number generator or a quantum experiment (as indicated by the rows) and yield either unbiased or biased outcomes (as indicated by the columns)

	Unbiased	Biased
Classical	PRNG	B-PRNG
Quantum	QRNG	B-QRNG



**Fig. 9** Distribution of 32-bit integers from the QRNG in analogy to Fig. 7. The values from the vector of random integers  $\mathbf{I}'$  are rescaled by a division by  $(2^{32} - 1)$  and sorted into 250 equally sized bins. The population of the  $k$ th bin (with  $k \in \{1, \dots, 250\}$ ) is denoted by  $c'_k$ . For comparison, we also show the corresponding population  $c_k$ , Eq. (11), from the B-QRNG and an outline of the uniform bin population

#### 4.1 RNGs

In total, we use four different RNGs to initialize neural network weights:

1. B-QRNG: Our hardware-biased quantum random number generator introduced in Sect. 3 from which we extract the integer sequence  $\mathbf{I}$  according to Eq. (9). The data is publicly available online (Heese et al., 2023).
2. QRNG: A bias-free quantum random number generator (ANU, 2021) based on quantum-optical hardware that performs broadband measurements of the vacuum field contained in the radio-frequency sidebands of a single-mode laser to produce a continuous stream of binary random numbers (Symul et al., 2011; Haw et al., 2015). We particularly use a publicly available pre-generated sequence of random bits from this stream (ANU, 2017), extract the first  $M$  bits and convert them into the integer sequence  $\mathbf{I}' \in \{0, \dots, 2^C - 1\}^L$  according to Eq. (9). Based on the Hellinger distance  $H(p'_c, \tilde{p}_c) \approx 0.0018$ , Eq. (7), with  $p'_c \equiv p'_c(k) \equiv c'_k/L$  and  $c'_k \equiv \sum_{i=1}^L \mathbb{1}(I'_i, k)$ , Eq. (12), for  $k \in \{1, \dots, K\}$ , we find that  $\mathbf{I}'$  is indeed much closer to the uniform distribution than  $\mathbf{I}$ , Eq. (17). We visualize the corresponding integer distribution in Fig. 9.

3. PRNG: The (presumably unbiased) native pseudo-random number generator from PyTorch.
4. B-PRNG: A “pseudo hardware-biased quantum random number generator”, which generates a bit string of i.i.d. Bernoulli random variables with a success probability  $p$  corresponding to the expected probability of all measured bits  $\bar{p}(1) = 1 - \bar{p}(0)$ , Eqs. (1) and (6), using the native pseudo-random number generator from PyTorch. The bit strings are then converted into integers according to Eq. (9). Their probability mass function is given by Eq. (13).

All of these RNGs, which are summarized in Tab. 1, produce 32-bit random numbers. However, the random numbers from the B-QRNG and the QRNG are taken in order (i. e., unshuffled) from the predefined sequences  $\mathbf{I}$  and  $\mathbf{I}'$ , respectively, whereas the PRNG and the B-PRNG algorithmically generate random numbers on demand based on a given random seed.

For the sake of completeness, we also analyze the binary random numbers from the B-QRNG and the QRNG, respectively, with the NIST Statistical Test Suite for the validation of random number generators (Rukhin et al., 2010; NIST, 2010). For this purpose, the bit strings are segmented into smaller sequences and multiple statistical tests are evaluated on each sequence. Each test consists of one or more sub-tests with the null hypothesis that the sequence being tested is random. Based on the proportion of sequences for which a sub-test satisfies the null hypothesis, it is considered as passed or rejected, where a rejection indicates non-randomness. A more detailed discussion about this procedure can also be found in Sýs et al. (2015).

A summary of our results is listed in Tab. 2. It shows that the B-QRNG numbers fail a majority of statistical tests of randomness, as expected, whereas the QRNG passes all.

## 4.2 CNN

In the first experiment, we consider a *LeNet-5* inspired CNN with ReLU activation functions and without dropout (Lecun et al., 1998). The network weights are initialized as proposed by He et al. (2015), but we use a uniform distribution instead of a normal distribution, as is also common. This means that each weight  $w_i$  (with  $i = 1, 2, \dots$ ) is sampled uniformly according to

$$w_i \sim [-h_i, h_i], \quad (18)$$

where  $h_i > 0$  is chosen such that a constant output variance can be achieved over all layers. The network biases are initialized analogously.

As data we use the MNIST handwritten digit recognition problem (LeCun et al., 1998), which contains 70,000 grayscale images of handwritten digits in  $28 \times 28$  pixel format. The digits are split into a training set of 60,000 images and a testing set of 10,000 images. The network is trained using *Adadelta* (Zeiler, 2012) over  $d \equiv 14$  epochs.

In Fig. 10 we show the CNN test accuracy convergence for each epoch over 31 independent training runs using the four RNGs from Sect. 4.1. The use of a biased RNG means that the He et al. initialization is actually effectively realized based on a non-uniform distribution instead of a uniform distribution. Therefore, such an approach could potentially be considered a new type of initialization strategy (depending on the bias), which is why one might expect a different training efficiency. However, the results show

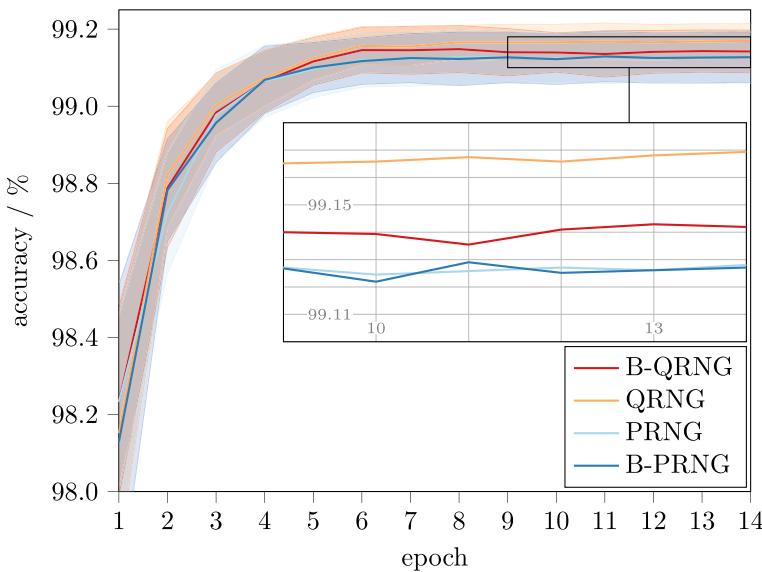
**Table 2** Summary of the results from the NIST Statistical Test Suite for the validation of random number generators (NIST, 2010) applied to the whole sequence of binary random numbers from the B-QRNG and the QRNG, respectively

Test name	✓	✗	Passed	✓	✗	Passed
Approximate entropy	0	1	0	1	0	99
Frequency within block	0	1	0	1	0	98
Cumulative sums	0	2	0, 0	2	0	98, 99
Discrete Fourier transform	0	1	60	1	0	100
Frequency	0	1	0	1	0	99
Linear complexity	1	0	100	1	0	96
Longest run of ones within block	0	1	72	1	0	100
Non-overlapping template matching	40	108	$52.4 \pm 41.5$	148	0	$99.0 \pm 1.0$
Overlapping template matching	0	1	0	1	0	100
Random excursions	0	0	–	8	0	$71.5 \pm 0.7$
Random excursions variant	0	0	–	18	0	$71.7 \pm 0.6$
Binary matrix rank	1	0	100	1	0	100
Runs	0	1	0	1	0	96
Serial	1	1	0, 100	2	0	98, 99
Maurer’s “universal statistical”	0	1	80	1	0	99
Total	43	119		188	0	
	B-QRNG			QRNG		

For each of the two bit strings, a series of statistical tests is evaluated, where each test consists of one or more sub-tests with the null hypothesis that the sequence being tested is random. The bit strings are segmented into sequences and each sub-test is run on each of these sequences. A sub-test is accepted or rejected based on a certain proportion of sequences that satisfy the null hypothesis. A rejection therefore indicates non-randomness. We choose 100 bitstreams in the program options such that each sequence contains at least  $10^6$  bits, as recommended. The required number of passed sequences of the total number of sequences for the acceptance of a sub-test is 96 of 100 for all tests except for the Random excursion tests, for which it is 68 of 72 due to a reduced number of effectively used sequences. The predefined standard parameters are used for all tests (e. g., significance level  $\alpha = 0.01$ ). We list for each statistical test the corresponding number of accepted (“✓”) and rejected (“✗”) sub-tests. The total number of acceptances and rejections are shown at the bottom in bold. In addition, the column “Passed” contains information about the number of passed sequences. For tests with only one or two sub-tests, we explicitly list the number of passed sequences. Otherwise, we present the respective means and standard deviations. A detailed description of the software and its statistical tests of randomness can be found in Rukhin et al. (2010). Summarized, the QRNG data passes all tests of randomness, whereas most of the tests fail for the B-QRNG data, which indicates non-randomness. Note that the Random excursions tests are not applicable for the B-QRNG data since they require the acceptance of the rejected Frequency test (Sýs et al., 2015)

that the choice of RNG for the network weight initialization has no major effect on the CNN test accuracy convergence. Only a closer look reveals that the mean QRNG results seem to be slightly superior to the others in the last epochs.

To quantify this observation, we utilize Welch’s (unequal variances)  $t$ -test for the null hypothesis that two independent samples have identical expected values without the assumption of equal population variance (Welch, 1947). We apply this test to two of each of the four results from different RNGs, where the resulting test accuracies from all runs in a specific epoch are treated as samples. We denote the two results to be compared as  $\mathbf{x}$  and  $\mathbf{y}$ , respectively, with  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{31 \times d}$  for 31 runs and  $d$  epochs. Consequently, for each pair of results and each epoch  $i \in \{1, \dots, d\}$ , we obtain a two-tailed p-value



**Fig. 10** CNN test accuracy convergence on the MNIST data set using four different random number generators (B-QRNG, QRNG, PRNG and B-PRNG from Sect. 4.1). Shown are mean values over 31 runs with the respective standard deviations (one sigma). The inset plot zooms in on the means of the final epochs

$p_i^t(\mathbf{x}, \mathbf{y})$ . The null hypothesis has to be rejected if such a p-value does not exceed the significance level, which we choose as  $\alpha = 0.05$ .

We are particularly interested whether the aforementioned hypothesis holds true for all epochs. To counteract the problem of multiple comparisons, we use the Holm-Bonferroni method (Holm, 1979) to adjust the p-values  $p_i^t(\mathbf{x}, \mathbf{y}) \mapsto \bar{p}_i^t(\mathbf{x}, \mathbf{y})$  for all  $i \in \{1, \dots, d\}$ . Summarized, if the condition

$$\min_{i, \mathbf{x}, \mathbf{y}} \bar{p}_i^t(\mathbf{x}, \mathbf{y}) \equiv \min_{\mathbf{x}, \mathbf{y}} \bar{p}_{\min}^t(\mathbf{x}, \mathbf{y}) > \alpha = 0.05 \quad (19)$$

is fulfilled, no overall statistically significant deviation between the results from different RNGs is present.

In addition, we also quantify the correlation of  $\mathbf{x}$  and  $\mathbf{y}$  using the Pearson correlation coefficient (Pearson, 1895)

$$\rho(\mathbf{x}, \mathbf{y}) \equiv \frac{\sum_{i=1}^c \bar{x}_i^m \bar{y}_i^m}{\sqrt{\sum_i (\bar{x}_i^m)^2 \sum_j (\bar{y}_j^m)^2}} \in [-1, 1] \quad (20)$$

of the mean values over all runs, where we make use of the abbreviations  $\bar{x}'_i \equiv x'_i - \sum_{i=1}^d x'_i / d$ ,  $x'_i \equiv \sum_{j=1}^{31} x_{ji} / 31$ ,  $\bar{y}'_i \equiv y'_i - \sum_{i=1}^d y'_i / d$ , and  $y'_i \equiv \sum_{j=1}^{31} y_{ji} / 31$ . A coefficient of 1 implies a perfect linear correlation of the means, whereas a coefficient of 0 indicates no linear correlation.

For the results from the CNN experiment, we obtain the similarity and correlation metrics listed in Tab. 3 in the rows marked with “CNN”. Summarized, we find a high mutual similarity [Eq. (19) holds true] and almost perfect mutual correlations of the

**Table 3** Minimum  $p$ -values from Welch’s  $t$ -test over all epochs  $\bar{p}_{\min}^t(\mathbf{x}, \mathbf{y})$ , Eq. (19), and Pearson correlation coefficient  $\rho(\mathbf{x}, \mathbf{y})$ , Eq. (20), of the experimental data

	<b>x</b>	<b>y</b>	$\bar{p}_{\min}^t(\mathbf{x}, \mathbf{y})$	$\rho(\mathbf{x}, \mathbf{y})$
CNN	B-QRNG	QRNG	0.3784	0.9984
	B-QRNG	PRGN	1.0000	0.9980
	B-QRNG	B-PRNG	0.9749	0.9986
	QRNG	PRGN	0.0641	0.9941
	QRNG	B-PRNG	0.0577	0.9992
	PRGN	B-PRNG	1.0000	0.9951
	B-QRNG	QRNG	1.0000	0.9997
	B-QRNG	PRGN	0.4526	0.9995
	B-QRNG	B-PRNG	1.0000	0.9998
	QRNG	PRGN	1.0000	0.9998
RNN-M	QRNG	B-PRNG	1.0000	0.9999
	PRGN	B-PRNG	0.5355	0.9996
	B-QRNG	QRNG	1.0000	0.9946
	B-QRNG	PRGN	1.0000	0.9954
	B-QRNG	B-PRNG	1.0000	0.9962
RNN-A	QRNG	PRGN	1.0000	0.9944
	QRNG	B-PRNG	1.0000	0.9951
	PRGN	B-PRNG	1.0000	0.9965
	B-QRNG	QRNG	1.0000	0.9946
	B-QRNG	PRGN	1.0000	0.9954

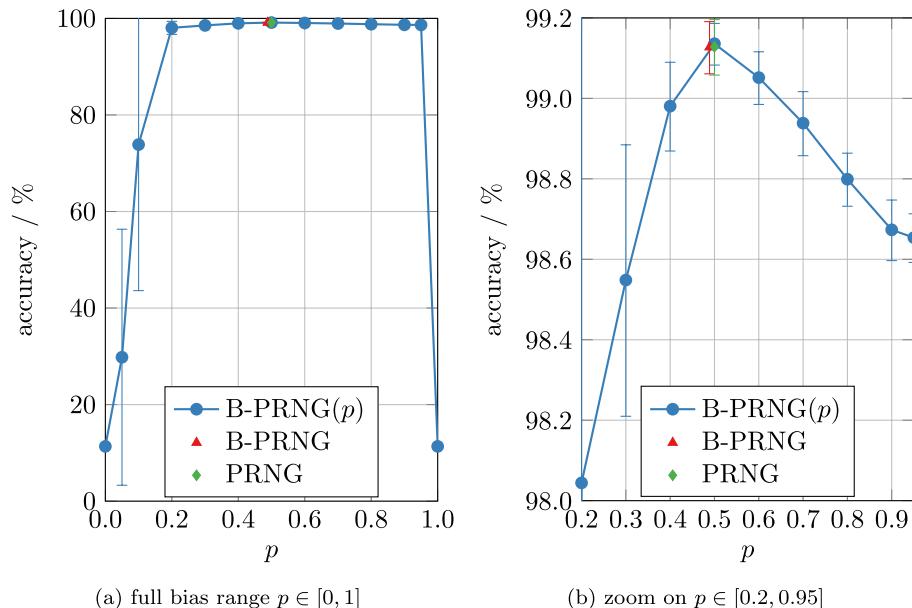
The metrics are listed for all mutual combinations of the results from the four RNGs (B-QRNG, QRNG, PRGN, and B-PRNG from Sect. 4.1) of all experiments (CNN, RNN-M, and RNN-A from Sects. 4.2 and 4.3, respectively)

results. This means that the choice of RNG for the network weight initialization has no statistically significant effect on the CNN test accuracy convergence and, in particular, the QRNG results are not superior despite the visual appearance in Fig. 10.

At this point, the question arises whether a different bias of the RNGs might have led to better training results. To answer this question, we consider additional pseudo-random number generators B-PRNG( $p$ ), which are based on a Bernoulli process with success probability  $p$ , Eq. (1), such that the originally considered B-PRNG corresponds to B-PRNG( $\bar{p}(1)$ ), Eq. (6), and the PRNG corresponds to B-PRNG(0.5). In the extreme cases of  $p = 0$  and  $p = 1$ , B-PRNG( $p$ ) is not random anymore and produces only constant values of 0 and  $2^{32} - 1$ , respectively. The probability mass function of the resulting integers is given by Eq. (13). We train the CNN again on the MNIST data set with a weight initialization based on B-PRNG( $p$ ) for different values of  $p \in [0, 1]$  and consider the test accuracy at epoch 14.

The results are shown in Fig. 11. Clearly, the mean test accuracy attains a maximum at  $p = 0.5$ , which corresponds to an unbiased pseudo-random number generator (i. e., the PRGN). For smaller and larger success probabilities, the mean test accuracy decreases. In particular, we observe a steep drop in performance for  $p < 0.2$  and  $p > 0.95$ , which indicates that a bias of the random number generator towards 0 has more severe effects than a bias towards 1. The worst performance is achieved for  $p = 0$  and  $p = 1$ , respectively.

We recall that for  $p = 0.5$ , weights are sampled uniformly around zero, Eq. (18). Thus, for  $p > 0.5$ , the weights are more probable to be positive, whereas for  $p < 0.5$ ,

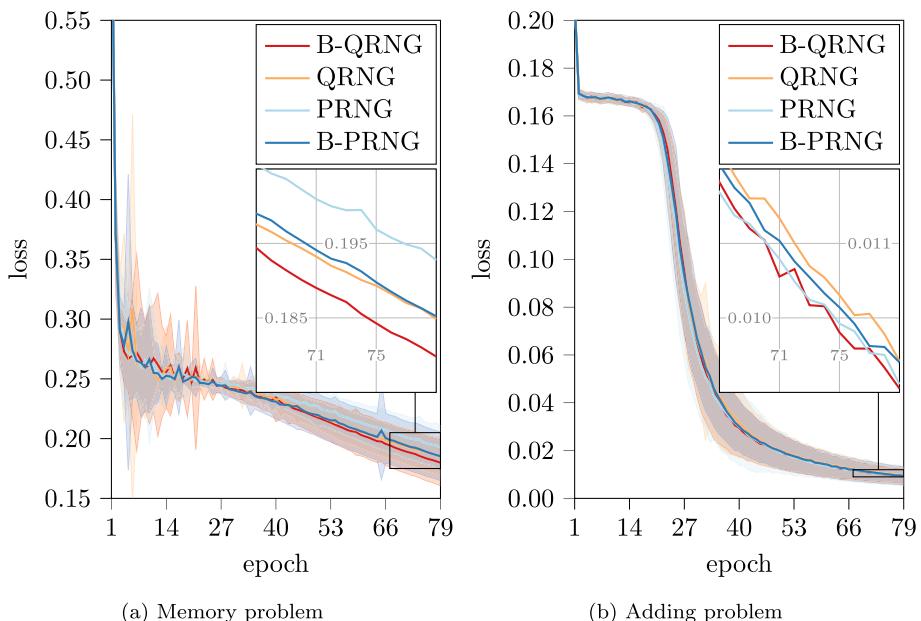


**Fig. 11** CNN test accuracy on the MNIST data at epoch 14 using different pseudo-random number generators B-PRNG( $p$ ), which are based on a Bernoulli process with success probability  $p$ , Eq. (1). We consider  $p \in \{0, .05, .1, .2, .3, .4, .5, .6, .7, .8, .9, .95, 1\}$ . Shown are mean values over 30 runs with the respective standard deviations (one sigma) as error bars for (a) the full bias range and (b) a zoom on the peak of the accuracy at  $p = 0.5$ . For comparison, we also plot the corresponding results from Fig. 10 for the B-PRNG with  $p = \bar{p}(1)$ , Eq. (6), as well as for the PRNG with  $p = 0.5$

they are more probable to be negative, cf. Eq. (14). Since our CNN contains ReLU activation functions, a shift of the weights towards negative values leads to vanishing gradients. According to our experiments, this seems to become significant for  $p < 0.2$ . On the other hand, an equivalent shift towards positive values does not drastically decrease the training performance and even for  $p = 0.95$  the test accuracy is above 98.6 %. However, for  $p = 1$  the test accuracy also drops. We think that the reason for this behavior is that the weights are in this case constant and attain the maximum value of the distribution, Eq. (18). The resulting lack of diversity, which is for example evident from the entropy, Eq. (15), is probably the cause for the bad training performance (Frankle & Carbin, 2019).

### 4.3 RNN

In the second experiment, we consider a recurrent LSTM cell with a uniform initialization in analogy to Eq. (18), which we apply on the synthetic adding and memory standard benchmarks (Hochreiter & Schmidhuber, 1997) with  $T = 64$  for the memory problem. For this purpose, we use RMSprop (Hinton, 2012) with a step size of e-3 to optimize LSTM cells (Hochreiter & Schmidhuber, 1997) with a state size of 256. For each problem, a total of 9e5 updates with training batches of size 128 is computed until the training stops. In total, there are  $\lfloor 9e5/128 \rfloor = 7031$  training steps.



**Fig. 12** RNN convergence on two benchmark data sets using four different RNGs (B-QRNG, QRNG, PRGN and B-PRNG from Sect. 4.1). Shown are mean values over 31 runs with the respective standard deviations (one sigma) in analogy to Fig. 10. The inset plot zooms in on the means of the final epochs

Since the synthetic data sets are infinitely large, overfitting is not an issue and we can consequently use the training loss as performance metric. Specifically, we consider 89 consecutive training steps as one epoch, which leads to  $d \equiv 4687/89 = 79$  epochs in total, each associated with the mean loss of the corresponding training steps.

The results are shown in Fig. 12, where we present the loss for each of the 79 epochs over 31 independent training runs for both problems. Again, we compare the results using random numbers from the four RNGs from Sect. 4.1. The use of a biased RNG effectively realizes a non-uniform initialization (depending on the bias) in comparison with the uniform initialization from a non-biased RNG. However, we find that no RNG yields a major difference in performance.

In analogy to the first experiment, we list the similarity and correlation metrics in Tab. 3 in the rows marked with ‘‘RNN-M’’ and ‘‘RNN-A’’, respectively. Again, we find a high mutual similarity [Eq. (19) holds true] and correlation. Thus, the choice of RNG also has no statistically significant effect in this second experiment. Due to the numerical effort required to train the RNNs, we cannot perform an analysis of different biases of RNGs as in the first experiment.

## 5 Conclusions

Summarized, by running a naively designed quantum random number generator on a quantum gate computer, we have generated a random bit string. Its statistical analysis has revealed a significant bias and mutual dependencies as imposed by the quantum hardware.

When converted into a sequence of integers, we have found a specially shaped distribution of values with a rich pattern. We have utilized these integers as hardware-biased quantum random numbers (B-QRNG). Motivated by the results from Bird et al. (2020), we have deliberately chosen to use these biased and correlated random numbers to study their impact on machine learning algorithms.

Specifically, we have studied their effect on the initialization of artificial neural network weights in two experiments. For comparison, we have additionally considered unbiased random numbers from another quantum random number generator (QRNG) and a classical pseudo-random number generator (PRNG) as well as random numbers from a classical pseudo-random number generator replicating the hardware bias (B-PRNG). The two experiments consider a CNN and a RNN, respectively, and show no statistically significant influence of the choice of RNG.

Despite a similar setup, we have not been able to replicate the observation from Bird et al. (2020), where it is stated that quantum random number generators and pseudo-random number generators “do inexplicably produce different results to one another when employed in machine learning.” However, we have not explicitly attempted to replicate the numerical experiments from the aforementioned work, but have instead considered two different examples that we consider typical applications of neural networks in machine learning.

Since our results are only exemplary, it may indeed be possible that there is an advantage in the usage of biased quantum random numbers for certain applications. Based on our studies, we expect, however, that in such cases it will in fact not be the “true randomness” of the quantum random numbers, but rather the opposite – their hardware-induced bias, including possible correlations – that will cause an effect. But is quantum hardware really necessary to produce such results? It seems that classical pseudo-random number generators are also able to mimic these effects. Even more, because the reliability and security of PRNGs can be ensured with less effort and a greater confidence than that of gate-based QRNGs on NISQ devices. Therefore, we think that for typical machine learning applications the usage of (high-quality) pseudo-random numbers is sufficient. Accordingly, a more elaborate experimental or theoretical study of the effects of biased pseudo-random numbers (with particular patterns) on certain machine learning applications could be a suitable research topic, e. g., to better understand the claims from Bird et al. (2020).

Repeatability is generally difficult to achieve for numerical calculations involving random numbers (Crane, 2018). In particular, our B-QRNG can in principle not be forced to reproduce a specific random sequence (as opposed to PRNGs). Furthermore, the statistics of the generated quantum random numbers may depend on the specific configuration of the quantum hardware at the time of operation. It might therefore be possible that a repetition of the numerical experiments with quantum random numbers obtained at a different time or from a different quantum hardware may lead to significantly different results. To ensure the greatest possible transparency, the source code for our experiments is publicly available online (Wolter, 2021) and may serve as a point of origin for further studies.

**Acknowledgements** We thank Christian Bauckhage and Bogdan Georgiev for informative discussion. Parts of this work have been funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01|S18038A and 01|S18038B), the Fraunhofer Cluster of Excellence Cognitive Internet Technologies (CCIT), the Fraunhofer Research Center Machine Learning as well as the State of North Rhine-Westphalia (Germany) as part of the Lamarr Institute for Machine Learning and Artificial Intelligence. We thank the University of Bonn for access to its *Auersberg* and *Teufelskapelle* clusters and acknowledge the use of IBM Quantum services for this work. Access to the IBM hardware has been funded by the Ministry of Science and Health of the State of Rhineland-Palatinate (Germany) as part

of the project AnQuC-2. For our numerical calculations, we have made particular use of Virtanen et al. (2020); Seabold and Perktold (2010); Paszke et al. (2019).

**Author contributions** All authors contributed to the research. RH particularly contributed to conception, writing, design and statistical analysis. MW particularly contributed to numerical experiments and Python code. NP particularly contributed to the machine learning and statistics background. All authors read and approved the final manuscript.

**Funding** Open Access funding enabled and organized by Projekt DEAL. Parts of this work have been funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01|S18038A and 01|S18038B), the Fraunhofer Cluster of Excellence Cognitive Internet Technologies (CCIT), the Fraunhofer Research Center Machine Learning as well as the State of North Rhine-Westphalia (Germany) as part of the Lamarr Institute for Machine Learning and Artificial Intelligence. All fundings are also listed in the acknowledgements.

**Availability of data and material** All data used for the numerical experiments is publicly accessible online via the respective references.

**Code availability** The code is publicly accessible online (Wolter, 2021).

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethical approval** Not Applicable.

**Consent for publication** All authors consent to the publication of this manuscript.

**Consent to participate** Not Applicable.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

Abbott, A. A., Calude, C. S., & Svozil, K. (2014). A quantum random number generator certified by value indefiniteness. *Mathematical Structures in Computer Science*, 24(3), e240303. <https://doi.org/10.1017/S0960129512000692>

Abbott, A. A., Calude, C. S., & Svozil, K. (2015). A variant of the Kochen-Specker theorem localising value indefiniteness. *Journal of Mathematical Physics*, 56(10), 102201. <https://doi.org/10.1063/1.4931658>

Abraham, H., & AduOffei, Agarwal R., et al. (2019) Qiskit: An open-source framework for quantum computing. <https://doi.org/10.5281/zenodo.2562110>

ANU QRNG. (2017). AARNet cloudstor: pre-generated random binary numbers. [https://cloudstor.aarnet.edu.au/plus/s/9Ik6rao7ACFyWL4/ANU\\_3May2012\\_100MB](https://cloudstor.aarnet.edu.au/plus/s/9Ik6rao7ACFyWL4/ANU_3May2012_100MB), Accessed on April 2021.

ANU QRNG. (2021). ANU QRNG quantum random numbers. <https://qrng.anu.edu.au/>, accessed on November 2021.

Balasch, J., Bernard, F., Fischer, V. et al. (2018). Design and testing methodologies for true random number generators towards industry certification. In *2018 IEEE 23rd European Test Symposium (ETS)*, pp. 1–10, <https://doi.org/10.1109/ETS.2018.8400697>

Bell, J. S., & Aspect, A. (2004). *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy* (2nd ed.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511815676>

Benioff, P. (1980). The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22, 563–591. <https://doi.org/10.1007/BF01011339>

Bera, M. N., Acín, A., Kuś, M., et al. (2017). Randomness in quantum mechanics: Philosophy, physics and technology. *Reports on Progress in Physics*, 80(12), 124001. <https://doi.org/10.1088/1361-6633/aa8731>

Biamonte, J., Wittek, P., Pancotti, N., et al. (2017). Quantum machine learning. *Nature*, 549(7671), 195–202. <https://doi.org/10.1038/nature23474>

Bird, J. J., Ekárt, A., & Faria, D. R. (2020). On the effects of pseudorandom and quantum-random number generators in soft computing. *Soft Computing*, 24(12), 9243–9256.

Boixo, S., Isakov, S. V., Smelyanskiy, V. N., et al. (2018). Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6), 595–600. <https://doi.org/10.1038/s41567-018-0124-x>

Brown, J., Zhang, J. F., Zhou, B., et al. (2020). Random-telegraph-noise-enabled true random number generator for hardware security. *Scientific Reports*, 10(1), 17210. <https://doi.org/10.1038/s41598-020-74351-y>

Bruzewicz, C. D., Chiaverini, J., McConnell, R., & Sage, J. M. (2019). Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2), 021314. <https://doi.org/10.1063/1.5088164>

Calude, C. S., Dinneen, M. J., Dumitrescu, M., & Svozil, K. (2010). Experimental evidence of quantum randomness incomputability. *Physical Review A*. <https://doi.org/10.1103/PhysRevA.82.022102>

Crane, M. (2018). Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association for Computational Linguistics*, 6, 241–252. [https://doi.org/10.1162/tacl\\_a\\_00018](https://doi.org/10.1162/tacl_a_00018)

Dunjko, V., Taylor, J. M., & Briegel, H. J. (2016). Quantum-enhanced machine learning. *Physical Review Letters*. <https://doi.org/10.1103/PhysRevLett.117.130501>

Forbes, C., Evans, M., Hastings, N., & Peacock, B. (2011). *Statistical distributions*. John Wiley & Sons.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. [arXiv:1803.03635](https://arxiv.org/abs/1803.03635)

Georgescu, I. M., Ashhab, S., & Nori, F. (2014). Quantum simulation. *Reviews of Modern Physics*, 86, 153–185. <https://doi.org/10.1103/RevModPhys.86.153>

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh, M. Titterington (Eds.) *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, PMLR, Chia Laguna Resort, Sardinia, Italy, Proceedings of Machine Learning Research, vol. 9, pp. 249–256, <http://proceedings.mlr.press/v9/glorot10a.html>

Grumblung, E., & Horowitz, M. (2019). *Quantum computing: Progress and prospects*. The National Academies Press. <https://doi.org/10.17226/25196>

Haw, J. Y., Assad, S. M., Lance, A. M., et al. (2015). Maximization of extractable randomness in a quantum random-number generator. *Physical Review Applied*, 3, 054004. <https://doi.org/10.1103/PhysRevApplied.3.054004>

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034

Heese, R., & Freyberger, M. (2014). Pointer-based simultaneous measurements of conjugate observables in a thermal environment. *Physical Review A*, 89, 052111. <https://doi.org/10.1103/PhysRevA.89.052111>

Heese, R., Wolter, M., & Mücke, S. et al. (2023). Hardware-biased quantum random numbers. <https://doi.org/10.5281/zenodo.8223863>, Accessed on August 2023.

Hellinger, E. (1909). Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik*, 1909(136), 210–271. <https://doi.org/10.1515/crll.1909.136.210>

Herrero-Collantes, M., & Garcia-Escartin, J. C. (2017). Quantum random number generators. *Reviews of Modern Physics*. <https://doi.org/10.1103/RevModPhys.89.015004>

Hinton, G. (2012). Neural networks for machine learning, lecture 6a overview of mini-batch gradient descent. <https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf>, Accessed on May 2021.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>

Holleman, J., Bridges, S., Otis, B. P., & Diorio, C. (2008). A 3  $\mu$ W CMOS true random number generator with adaptive floating-gate offset cancellation. *IEEE Journal of Solid-State Circuits*, 43(5), 1324–1336. <https://doi.org/10.1109/JSSC.2008.920327>

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2), 65–70.

Huang, H. L., Wu, D., Fan, D., & Zhu, X. (2020). Superconducting quantum computing: A review. [arXiv:2006.10433](https://arxiv.org/abs/2006.10433)

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3), 457–506. <https://doi.org/10.1007/s10994-021-05946-3>

IBM. (2021). IBM Quantum. <https://quantum-computing.ibm.com>

Jacak, J. E., Jacak, W. A., Donderowicz, W. A., & Jacak, L. (2020). Quantum random number generators with entanglement for public randomness testing. *Scientific Reports*, 10(1), 164. <https://doi.org/10.1038/s41598-019-56706-2>

James, F., & Moneta, L. (2020). Review of high-quality random number generators. *Computing and Software for Big Science*, 4(1), 2. <https://doi.org/10.1007/s41781-019-0034-3>

Khrennikov, A. (2015). Randomness: Quantum versus classical. [arXiv:1512.08852](https://arxiv.org/abs/1512.08852)

Kim, E., Lee, M., & Kim, J. J. (2017). 8Mb/s 28Mb/mJ robust true-random-number generator in 65nm CMOS based on differential ring oscillator with feedback resistors. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 144–145. <https://doi.org/10.1109/ISSCC.2017.7870302>

Kochen, S., & Specker, E. P. (1975). *The problem of hidden variables in quantum mechanics* (pp. 293–328). Springer. [https://doi.org/10.1007/978-94-010-1795-4\\_17](https://doi.org/10.1007/978-94-010-1795-4_17)

Kofler, J., & Zeilinger, A. (2010). Quantum information and randomness. *European Review*, 18(4), 469–480. <https://doi.org/10.1017/S1062798710000268>

Kulikov, A., Jerger, M., Potočnik, A., et al. (2017). Realization of a quantum random generator certified with the Kochen-Specker theorem. *Physical Review Letters*. <https://doi.org/10.1103/PhysRevLett.119.240501>

LaRose, R. (2019). Overview and comparison of gate level quantum software platforms. *Quantum*, 3, 130. <https://doi.org/10.22331/q-2019-03-25-130>

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>

LeCun, Y., Cortes, C., & Burges, CJC. (1998). The MNIST database of handwritten digits. <http://yann.lecun.com/exdb/mnist>, Accessed on May 2021.

Leone, N., Rusca, D., Azzini, S., et al. (2020). An optical chip for self-testing quantum random number generation. *APL Photonics*, 5(10), 101301. <https://doi.org/10.1063/5.0022526>

Leymann, F., & Barzen, J. (2020). The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Science and Technology*, 5(4), 044007. <https://doi.org/10.1088/2058-9565/abae7d>

Li, C., Zhang, J., Sang, L., et al. (2020). Deep learning-based security verification for a random number generator using white chaos. *Entropy*, 22(10), 1134.

Martínez, A. C., Solis, A., Díaz Hernández Rojas, R., et al. (2018). Advanced statistical testing of quantum random number generators. *Entropy*. <https://doi.org/10.3390/e20110886>

Mathew, S., Johnston, D., & Newman, P. et al. (2015).  $\mu$ RNG: A 300–950mV 323Gbps/W all-digital full-entropy true random number generator in 14nm FinFET CMOS. In *ESSCIRC Conference 2015 - 41st European Solid-State Circuits Conference (ESSCIRC)*, pp. 116–119. <https://doi.org/10.1109/ESSCIRC.2015.7313842>

Nachman, B., & Geller, M. R. (2021). Categorizing readout error correlations on near term quantum computers. [arXiv:2104.04607](https://arxiv.org/abs/2104.04607)

Nielsen, M. A., & Chuang, I. L. (2011). *Quantum computation and quantum information: 10th anniversary edition* (10th ed.). Cambridge University Press.

NIST. (2010). Statistical Test Suite. <https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software>, Accessed on May 2021.

Norsen, T. (2017). *Foundations of quantum mechanics*. Springer.

Park, K., Park, S., Choi, B. G., et al. (2020). A lightweight true random number generator using beta radiation for IoT applications. *ETRI Journal*, 42(6), 951–964. <https://doi.org/10.4218/etrij.2020-0119>

Pasqualini, L., Parton, M. (2020). Pseudo random number generation: a reinforcement learning approach. *Procedia Computer Science* 170:1122–1127, <https://doi.org/10.1016/j.procs.2020.03.057>, <https://www.sciencedirect.com/science/article/pii/S1877050920304944>, the 11th International Conference on Ambient Systems, Networks and Technologies (ANT)/The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40)/Affiliated Workshops.

Paszke, A., Gross, S., & Massa, F. et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In: H. Wallach, H. Larochelle, A. Beygelzimer et al. (Eds.) *Advances in Neural Information Processing Systems 32*, (pp. 8024–8035) Curran Associates, Inc.

Pearson, K. (1895). Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58, 240–242.

Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302), 157–175. <https://doi.org/10.1080/1478644009463897>

Petrov, M., Radchenko, I., & Steiger, D. et al. (2020). Independent security analysis of a commercial quantum random number generator. [arXiv:2004.04996](https://arxiv.org/abs/2004.04996)

Picard, D. (2021). Torch.manual\_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. [arXiv:2109.08203](https://arxiv.org/abs/2109.08203)

Pirandola, S., Andersen, U. L., Banchi, L., et al. (2020). Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4), 1012. <https://doi.org/10.1364/aop.361502>

Pironio, S., Acín, A., Massar, S., et al. (2010). Random numbers certified by Bell's theorem. *Nature*, 464(7291), 1021–1024. <https://doi.org/10.1038/nature09008>

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, 2, 79. <https://doi.org/10.22331/q-2018-08-06-79>

Puglisi, F. M., Zagni, N., Larcher, L., & Pavan, P. (2018). Random telegraph noise in resistive random access memories: Compact modeling and advanced circuit design. *IEEE Transactions on Electron Devices*, 65(7), 2964–2972. <https://doi.org/10.1109/TED.2018.2833208>

Rajashekharan, L., & Shunmuga Velayutham, C. (2016). Is differential evolution sensitive to pseudo random number generator quality?—An investigation. In S. Berretti, S. M. Thampi, & P. R. Srivastava (Eds.), *Intelligent systems technologies and applications* (pp. 305–313). Springer International Publishing.

Roffe, J. (2019). Quantum error correction: An introductory guide. *Contemporary Physics*, 60(3), 226–245. <https://doi.org/10.1080/00107514.2019.1667078>

Rukhin, A., Soto, J., & Nechvatal, J. et al. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. Rep. Natl. Inst. Stand. Technol. Spec. Publ. 800-22rev1a, National Institute of Standards and Technology.

Seabold, S., Perktold, J. (2010). statsmodels: econometric and statistical modeling with Python. In *9th Python in Science Conference*.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3), 379–423. <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>

Shikano, Y., Tamura, K., & Raymond, R. (2020). Detecting temporal correlation via quantum random number generation. *Electronic Proceedings in Theoretical Computer Science*, 315, 18–25. <https://doi.org/10.4204/EPTCS.315.2>

Symul, T., Assada, S. M., & Lamb, P. K. (2011). Real time demonstration of high bitrate quantum random number generation with coherent laser light. *Applied Physics Letters*, 98, 231103. <https://doi.org/10.1063/1.3597793>

Sýs, M., Ríha, Z., Matyáš, V., et al. (2015). On the interpretation of results from the NIST Statistical Test Suite. *Romanian Journal of Information Science and Technology*, 18(1), 18–32.

Tamura, K., Shikano, Y. (2020). Quantum random number generation with the superconducting quantum computer IBM 20Q Tokyo. Cryptology ePrint Archive, Report 2020/078, <https://ia.cr/2020/078>

Tamura, K., Shikano, Y., et al. (2021). Quantum random numbers generated by a cloud superconducting quantum computer. In T. Takagi, M. Wakayama, & K. Tanaka (Eds.), *International symposium on mathematics, quantum theory, and cryptography* (pp. 17–37). Springer Singapore.

Tokunaga, C., Blaauw, D., & Mudge, T. (2008). True random number generator with a metastability-based quality control. *IEEE Journal of Solid-State Circuits*, 43(1), 78–85. <https://doi.org/10.1109/JSSC.2007.910965>

Vacchini, B. (2016). Quantum noise from reduced dynamics. *Fluctuation and Noise Letters*, 15(03), 1640003. <https://doi.org/10.1142/s0219477516400034>

Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>

Wald, A., & Wolfowitz, J. (1940). On a test whether two samples are from the same population. *The Annals of Mathematical Statistics*, 11(2), 147–162. <https://doi.org/10.1214/aoms/1177731909>

Welch, B. L. (1947). The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1–2), 28–35. <https://doi.org/10.1093/biomet/34.1-2.28>

Wolter, M. (2021). Python implementation of the experiments from this manuscript. <https://github.com/Castle-Machine-Learning/quantum-init-experiments>

Wu, Y., Bao, W. S., Cao, S., et al. (2021). Strong quantum computational advantage using a superconducting quantum processor. *Physical Review Letters*, 127, 180501. <https://doi.org/10.1103/PhysRevLett.127.180501>

Yu, F., Li, L., Tang, Q., et al. (2019). A survey on true random number generators based on chaos. *Discrete Dynamics in Nature and Society*, 2019, 2545123. <https://doi.org/10.1155/2019/2545123>

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. [arXiv:1212.5701](https://arxiv.org/abs/1212.5701)

Zurek, W. H. (2007). Decoherence and the transition from quantum to classical — revisited. In B. Duplantier, J. M. Raimond, & V. Rivasseau (Eds.) *Quantum Decoherence: Poincaré Seminar 2005*, Birkhäuser Basel, Basel, pp. 1–31, [https://doi.org/10.1007/978-3-7643-7808-0\\_1](https://doi.org/10.1007/978-3-7643-7808-0_1)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.