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

Raoul Heese[1*] ⓘ, Moritz Wolter[2], Sascha Mücke[3],

Lukas Franken[4], Nico Piatkowski[4] ⓘ

[1] *Fraunhofer Institute for Industrial Mathematics ITWM*
[2] *Fraunhofer Institute for Algorithms and Scientific Computing SCAI*
[3] *Artificial Intelligence Group, TU Dortmund University*
[4] *Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS*
[*] *raoul.heese@itwm.fraunhofer.de*

August 31, 2021

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 (NISQ) devices. A common property of quantum computers is that they 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.

1 Introduction

The intrinsic non-deterministic nature of quantum mechanics (Kofler and 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. observed that quantum random numbers can lead to superior results for certain numerical experiments in comparison with classically[1] 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 (Shikano et al., 2020, Tamura and 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.

---

[1] 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).

We therefore revisit this topic in the present manuscript and generate purposely biased random numbers using real quantum hardware. The structure of the remaining paper is as follows. In section 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 section 3. In section 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.

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 and Chuang, 2011).

2.1.1 Quantum devices

A quantum computer is a processor which utilizes quantum mechanical phenomena to process information (Benioff, 1980, Grumbling and 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, Boixo et al., 2018).

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 and 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 and 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. 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 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 unavoidable element of randomness. This randomness is in particular not a consequence of uncertainty or noise, but instead an intrinsic property of quantum mechanics (Bera et al., 2017). A sketch of the quantum computation process is shown in Fig. 1.

2.1.2 Quantum machine learning

In a machine learning context, we may consequently 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.

With the currently available NISQ technology, the result of a quantum computation on a physical device may deviate significantly from the expected outcome. 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.

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 and Geller, 2021). All of them are a consequence of the imperfect physical hardware and they are non-negligible for NISQ devices (Leymann and Barzen, 2020). 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: First, the uncertainty caused by the hardware-related errors, and second, the uncertainty caused by the intrinsic randomness. Furthermore, hardware-related errors can also induce additional systematic deviations of the computation result. 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 that cannot be avoided. It is this intrinsic uncertainty which can be exploited to construct QRNGs.
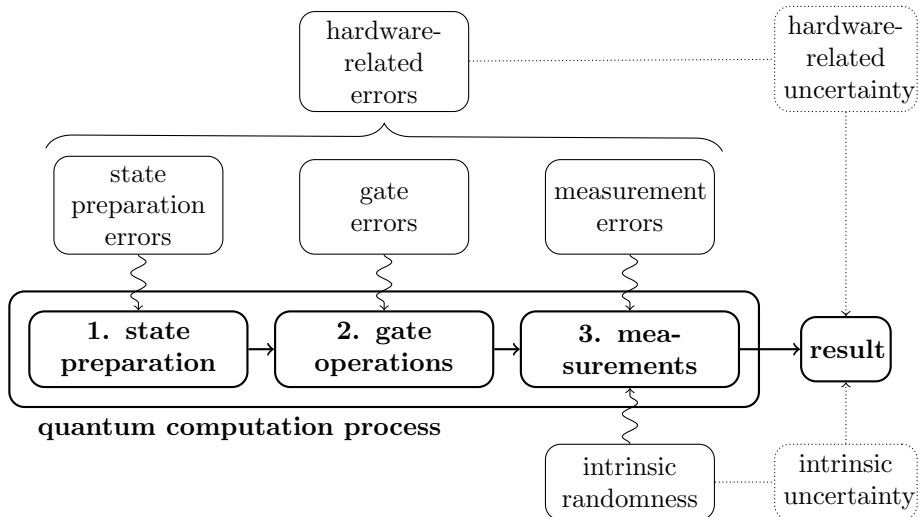
2.2 Random number generation

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 stochstic 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 and 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 and Moneta, 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

Figure 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 (and additional systematic deviations) 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 unavoidable aspect of quantum computing and can be exploited to construct QRNGs.

by the underlying algorithm). We remark that PRNGs can also be constructed based on machine learning algorithms (Pasqualini and Parton, 2020).

The more advanced true random number generators (TRNGs) are hardware devices that receive a signal from a complex physical process to extract random numbers (Yu et al., 2019), for example the telegraph noise in a transistor (Brown et al., 2020), which is unpredictable for all practical purposes. 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. Typically, statistical test 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 exemplarily verified, e.g., in Rajashekharan and Shunmuga 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. 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) (Herrero-Collantes and Garcia-Escartin, 2017). Such QRNGs have already been realized with different quantum systems, for example using nuclear decay (Park et al., 2020) or optical devices (Leone et al., 2020).

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 and Chuang, 2011). Each measurement of the circuit consequently generates a sequence of random bits, one for each qubit. However, due to imperfections of the quantum hardware, such a QRNG is likely to produce biased outcomes. For this reason, technically more refined solutions are necessary and QRNGs have to be certified similar to classical RNGs.

Currently, there exist various commercially available QRNGs, which can be used to create quantum random numbers on demand. 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.
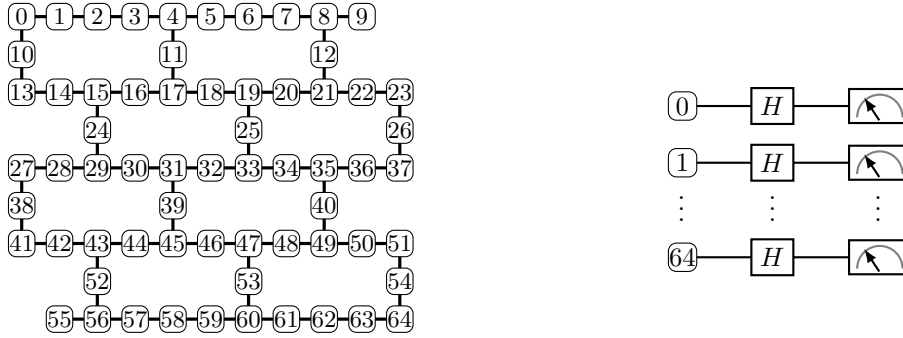
3 Biased QRNG

Motivated by the work in Bird et al. (2020), we take a different approach in this manuscript than is usually used. 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 shown in Fig. 1). In the present section, we describe our experimental setup for this purpose 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 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. 2(a). 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 $T_1 \approx (59.11 \pm 15.25)\,\text{µs}$ and $T_2 \approx (74.71 \pm 31.22)\,\text{µs}$, respectively.
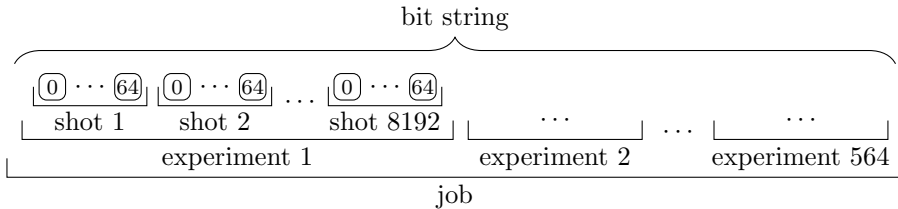
Initially, all qubits in this backend are prepared in the ground state. Our B-QRNG cicuit, which is sketched in Fig. 2(b), consists of one Hadamard gate applied to each qubit such that it is brought into a balanced superposition of ground state and excited state. A subsequent measurement on each qubit 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

(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%.

Figure 2: Main components of our B-QRNG setup: (a) topology diagram of the backend and (b) circuit diagram.

Figure 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 300 318 720 bits.

errors. An analogous setup with a different backend is considered in 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.

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 $M \equiv NSR = 300\,318\,720$ as sketched in Fig. 3. The choice of R is determined by the condition $M \gtrsim 3 \times 10^8$, which we have estimated as sufficient for our numerical experiments. We split the bit string into chunks of length $C \equiv 32$ to obtain $L \equiv M/C = 9\,384\,960$ 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 nth qubit can be considered as a Bernoulli random variable (Forbes et al., 2011), where $n \in \{0, \ldots, 64\}$ represents the hardware index as outlined in Fig. 2. Such a variable has a probability mass function

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

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 nth qubit is given by the vector

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

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 nth qubit is given by

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

with the indicator function

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

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 $p_0(b), \ldots, p_N(b)$ correspond to the uniform probability

$$\tilde{p} \equiv \tilde{p}(b) \equiv \frac{1}{2} \tag{5}$$

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 $\tilde{p}$, Eq. (5). In particular, we find an expected probability and standard deviation with respect to all measured bits of

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

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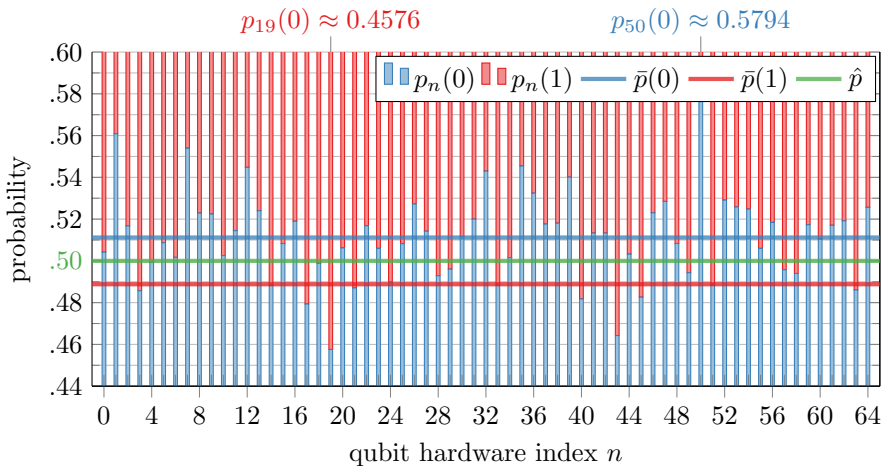
From a χ^2 test (Pearson, 1900) on the measured bit distribution, the null hypothesis of a uniform zero bit occurrence can be rejected as expected with a confidence level of 1.0000. To further quantify the deviation of the measured probabilities from a uniform distribution, we utilize the discrete Hellinger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

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

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

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

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.

Figure 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 nth 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 $\tilde{p}$, 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.

3.2.2 Randomness

Although quantum events intrinsically exhibit a truly random behavior, the output from 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 and 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 nth 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 $p^{\mathrm{r}} \approx 0.0000 < \alpha$ such that the test also fails for the entire sequence of random numbers.
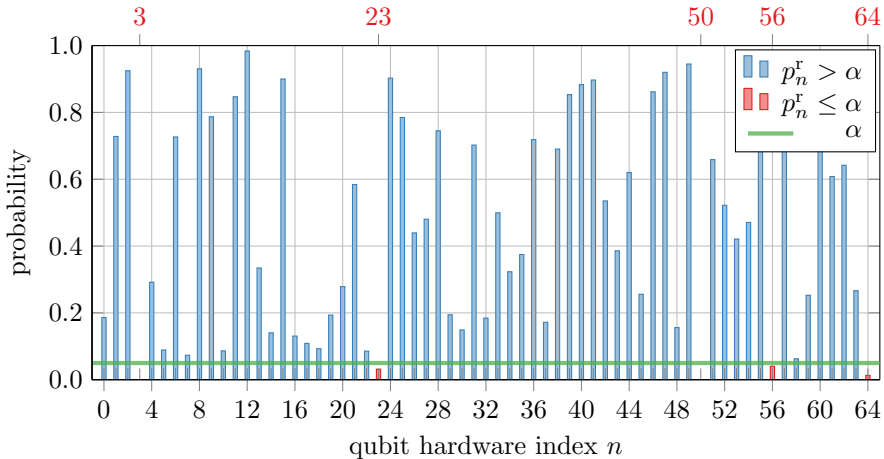
Summarized, we find that the reliability of the generated quantum random numbers is questionable. A typical binary random sequence from a PRNG of the same length as $\mathbf{B}$ can be expected to pass the Wald–Wolfowitz runs test. However, within the scope of this work, the reason for this observation cannot be further investigated and we accept it as an integral part of our naive approach to the B-QRNG. A more detailed study of the properties of our setup (applied to a different quantum hardware) can be found in Tamura and Shikano (2021), which contains similar observations.

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 $\mathbf{B} \mapsto \mathbf{I} \in \{0, \ldots, 2^C - 1\}^L$ by consecutively grouping its elements into bit strings of length C and converting them to non-negative integers according to

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

Figure 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 nth 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 $p_n^{\mathrm{r}} \leq \alpha$, the corresponding hardware indices are additionally denoted on top of the plot and indicate the qubits that fail the test of randomness.

with $j \in \{1, \ldots, L\}$. 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 $\xi \equiv 2^C - 1$ 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 kth bin is given by

$$c_k \equiv \sum_{i=1}^{L} \mathbb{1}(I_i, k) \tag{10}$$

with the indicator function

$$\mathbb{1}(i, k) \equiv \begin{cases} 1 & \text{if } k < K \ \wedge \ \frac{k-1}{K} \leq \frac{i}{\xi} < \frac{k}{K} \\ 1 & \text{if } k = K \ \wedge \ \frac{K-1}{K} \leq \frac{i}{\xi} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$
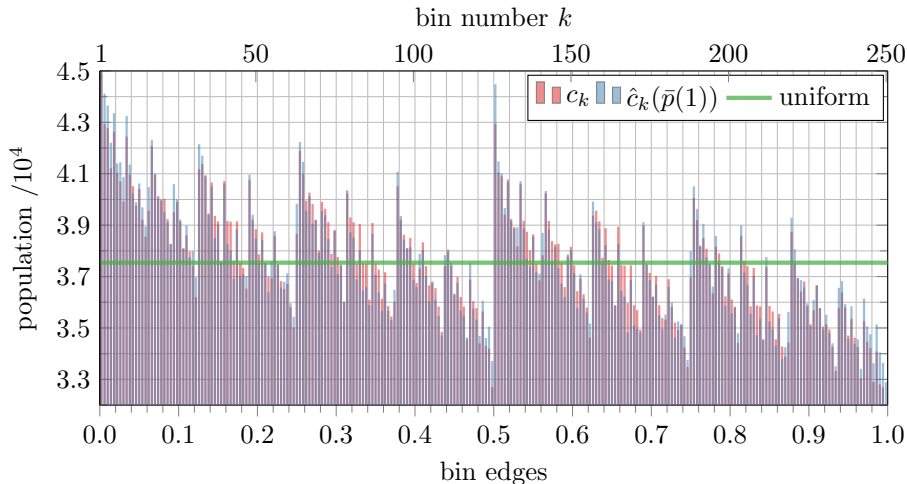
for $k \in \{1, \ldots, K\}$.

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). Based on this presumption, the predicted (possibly non-integer) population of the kth bin is given by

$$\hat{c}_k(p) \equiv L \sum_{i=0}^{\xi} \mathbb{1}(i, k) P(i, p) \tag{12}$$

with $P(j, p) \equiv \prod_{i=1}^{C} p^{\tau_i(j)} (1-p)^{1-\tau_i(j)}$. The latter expression contains the binary vector $\tau(j) \in \mathbb{B}^C$, which is the unique result of the relation $\sum_{i=0}^{C-1} \tau_{i+1}(j) 2^i = j$ for $j \in \{0, \ldots, \xi\}$.

We show both the measured bin population c_k, Eq. (10), and the theoretical bin population $\hat{c}_k(p)$, Eq. (12), 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. 6. 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,

Figure 6: Measured distribution of 32-bit integers from the B-QRNG. The values from the generated vector of random integers **I**, Eq. (9), are rescaled by a division by $(2^{32}-1)$ and sorted into 250 equally sized bins. The kth bin (with $k \in \{1, \ldots, 250\}$) has a population of c_k according to Eq. (10). For comparison, the corresponding theoretic bin population of the kth bin $\hat{c}_k(\bar{p}(b))$ is shown, which is obtained from a Bernoulli process according to Eq. (12) 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.

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 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

$$\mathrm{H}(p_c, \tilde{p}_c) \approx 0.0213, \tag{13}$$

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 $k \in \{1, \ldots, K\}$. This metric quantifies our observations from Fig. 6.
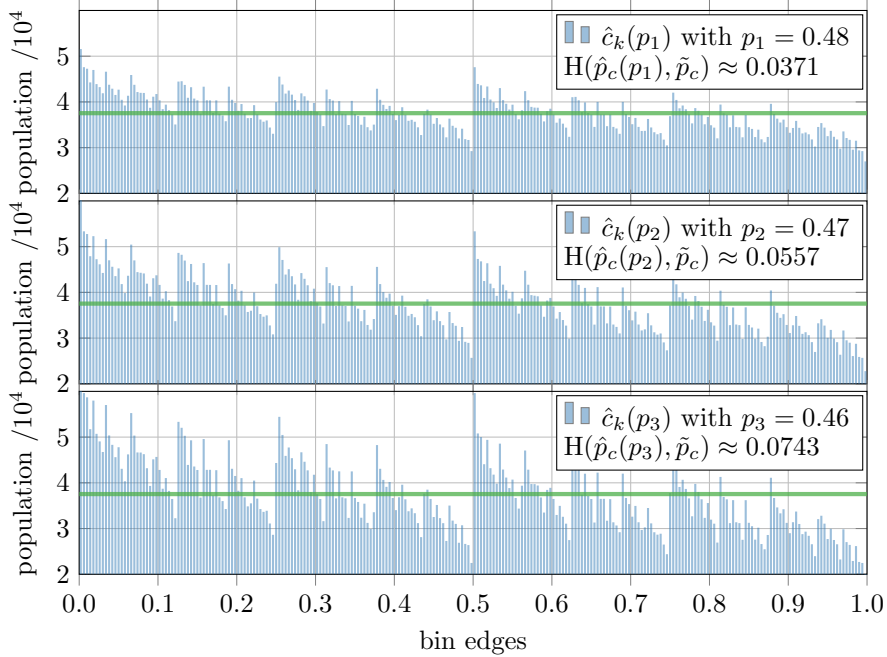
For comparative purposes, we show additional theoretical bin populations for other success probabilities in Fig. 7. As expected, the rugged pattern of the distribution becomes sharper for higher values of p and the deviation from the uniform distribution increases.

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. (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.

Figure 7: Theoretical distribution of 32-bit integers in analogy to Fig. 4 for different success probabilities $p \in \{p_1, p_2, p_3\}$, Eq. (12). We also show the corresponding Hellinger distance $\mathrm{H}(\hat{p}_c(p), \tilde{p}_c)$, Eq. (7), with $\hat{p}_c(p) \equiv \hat{p}_c(p; k) \equiv \hat{c}_k(p)/L$ and the uniform distribution $\tilde{p}_c$ as used in Eq. (13), respectively, where $k \in \{1, \ldots, K\}$.

4.1 RNGs

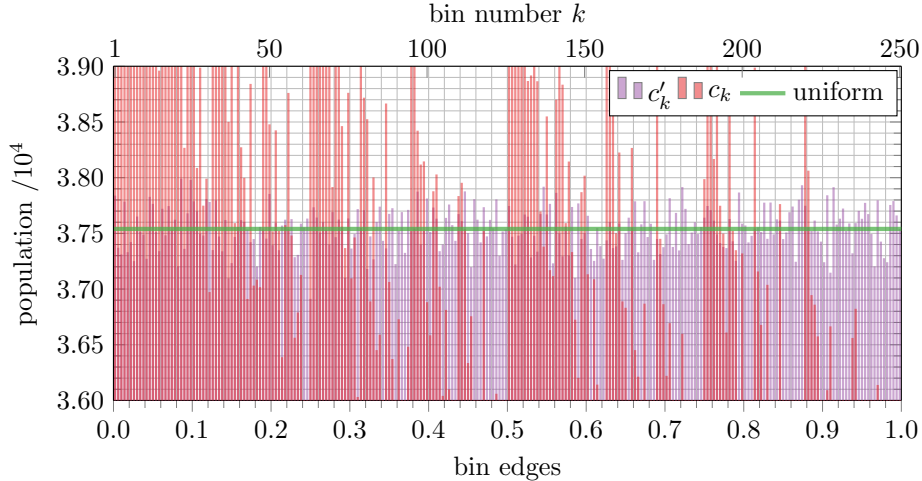In total, we use four different RNGs to initialize neural network weights:

1. B-QRNG: Our hardware-biased quantum random number generator introduced in section 3 from which we extract the integer sequence **I** according to Eq. (9).

2. QRNG: A bias-free quantum random number generator 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 QRNG, 2017), extract the first M bits and convert them into the integer sequence $\mathbf{I}' \in \{0, \ldots, 2^C - 1\}^L$ according to Eq. (9). Based on the Hellinger distance $\mathrm{H}(p'_c, \tilde{p}_c) \approx 0.0018$, 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. (11), for $k \in \{1, \ldots, K\}$, we find that $\mathbf{I}'$ is indeed much closer to the uniform distribution than **I**, Eq. (13). We visualize the corresponding integer distribution in Fig. 8.

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 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).

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 **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

Table 1: Overview over the four considered RNGs presented in section 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

Figure 8: Distribution of 32-bit integers from the QRNG in analogy to Fig. 6. 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 kth bin (with $k \in \{1, \dots, 250\}$) is denoted by c_k'. For comparison, we also show the corresponding population c_k, Eq. (10), from the B-QRNG and an outline of the uniform bin population.

number generators (Rukhin et al., 2010, NIST, 2010). A summary of the results is listed in Tab. 2 and shows that the B-QRNG numbers fails a majority of statistical tests of randomness, whereas the QRNG passes all, as expected.

4.2 CNN

In the first experiment, we consider a *LeNet-5* inspired CNN without dropout (Lecun et al., 1998). The network weights are uniformly initialized as proposed by He et al. (2015). As data we use the MNIST handwritten digit recognition problem (LeCun et al., 1998), which contains $70\,000$ grayscale images of handwritten digits in 28×28 pixel format. The digits are split into a training set of $60\,000$ images and a training set of $10\,000$ images. The network is trained using *Adadelta* (Zeiler, 2012) over $d \equiv 14$ epochs.

In Fig. 9 we show the CNN test accuracy convergence for each epoch over 31 independent training runs using the four RNGs from section 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 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 $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{31 \times d}$

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. Specifically, we have considered 10 bit streams containing 30 031 872 bits each. A detailed description of the software and its statistical tests of randomness can be found in Rukhin et al. (2010). Several iterations of each test are performed and we list the corresponding number of acceptances ("pass") and rejections ("reject") of the null hypothesis that the sequence is random. For all tests, the predefined standard parameters are used. We also list the total number of acceptances and rejections in bold.

Test name	pass	reject	pass	reject
Approximate entropy	0	1	1	0
Frequency within block	0	1	1	0
Cumulative sums	0	2	2	0
Discrete Fourier transform	0	1	1	0
Frequency	0	1	1	0
Linear complexity	1	0	1	0
Longest run of ones within block	0	1	1	0
Non-overlapping template matching	15	133	148	0
Overlapping template matching	0	1	1	0
Random excursions	0	0	8	0
Random excursions variant	0	0	18	0
Binary matrix rank	1	0	1	0
Runs	0	1	1	0
Serial	1	1	2	0
Maurer's "universal statistical"	0	1	1	0
Total	**18**	**144**	**188**	**0**
	B-QRNG		QRNG	

for 31 runs and d epochs. Consequently, for each pair of results and each epoch $i \in \{1, \ldots, d\}$, we obtain a two-tailed p-value $p_i^t(\mathbf{x}, \mathbf{y})$. 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 $i \in \{1, \ldots, d\}$. Summarized, if the condition
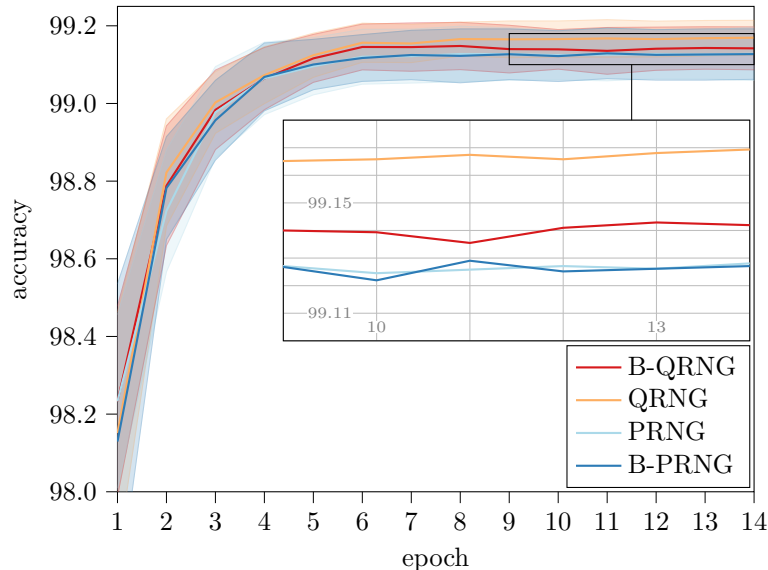
$$\min_{i,\mathbf{x},\mathbf{y}} \bar{p}_i^t(\mathbf{x}, \mathbf{y}) \equiv \min_{\mathbf{x},\mathbf{y}} \bar{p}_{\min}^t(\mathbf{x}, \mathbf{y}) \overset{!}{>} \alpha = 0.05 \tag{14}$$

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)

$$\rho(\mathbf{x}, \mathbf{y}) \equiv \frac{\sum_{i=1}^c \bar{x}_i^m \bar{y}_i^m}{\sqrt{\sum_i (\bar{x}_i^m)^2 \sum_j (\bar{y}_j^m)^2}} \in [-1, 1] \tag{15}$$

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.

Figure 9: CNN test accuracy convergence on the MNIST data set using four different random number generators (B-QRNG, QRGN, PRGN and B-PRNG from section 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.

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. (14) holds true) and almost perfect mutual correlations of the 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. 9.

4.3 RNN

In the second experiment, we consider a recurrent LSTM cell with a uniform initialization on the synthetic adding and memory standard benchmarks (Hochreiter and Schmidhuber, 1997) with $T = 64$ for the memory problem. For this purpose, we use *RMSprop* (Hinton, 2012) with a step size of 10^{-3} to optimize LSTM cells (Hochreiter and Schmidhuber, 1997) with a state size of 256. For each problem, a total of 9×10^5 updates with training batches of size 128 is computed until the training stops. In total, there are $\lfloor 9 \times 10^5/128 \rfloor = 7031$ training steps.

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. 10, 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 section 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. (14) holds true) and correlation. Thus, the choice of RNG also has no statistically significant effect in this second experiment.

Table 3: Minimum p-values from Welch's t-test over all epochs $\bar{p}^t_{\min}(\mathbf{x}, \mathbf{y})$, Eq. (14), and Pearson correlation coefficient $\rho(\mathbf{x}, \mathbf{y})$, Eq. (15), of the experimental data. The metrics are listed for all mutual combinations of the results from the four RNGs (B-QRNG, QRGN, PRGN, and B-PRNG from section 4.1) of all experiments (CNN, RNN-M, and RNN-A from sections 4.2 and 4.3, respectively).

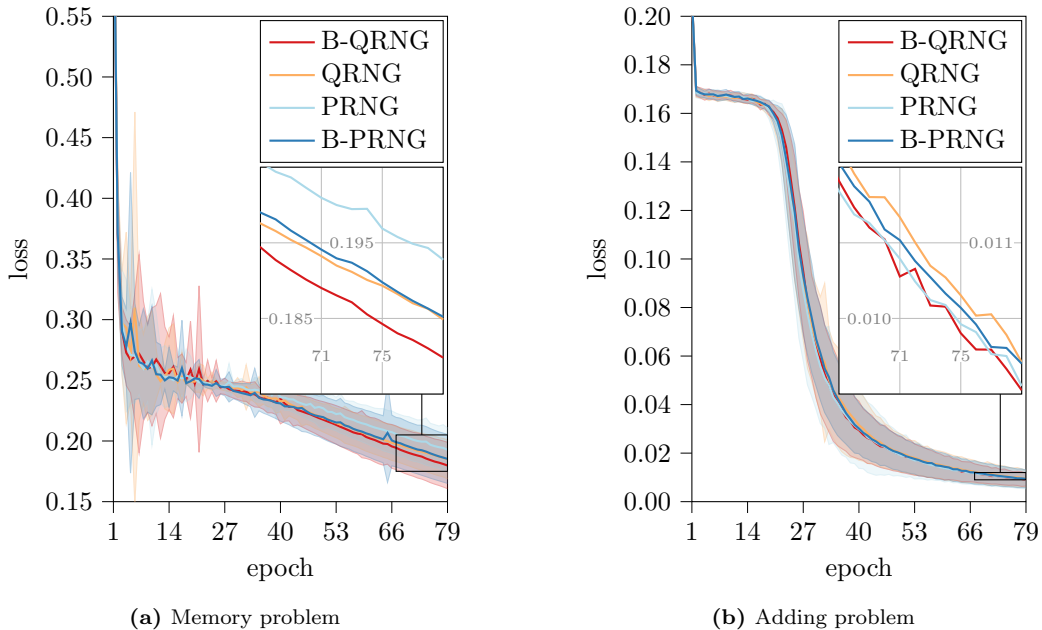|	$\mathbf{x}$	$\mathbf{y}$	$\bar{p}^t_{\min}(\mathbf{x}, \mathbf{y})$	$\rho(\mathbf{x}, \mathbf{y})$
CNN	B-QRNG	QRGN	0.3784	0.9984
	B-QRNG	PRGN	1.0000	0.9980
	B-QRNG	B-PRNG	0.9749	0.9986
	QRGN	PRGN	0.0641	0.9941
	QRGN	B-PRNG	0.0577	0.9992
	PRGN	B-PRNG	1.0000	0.9951
RNN-M	B-QRNG	QRGN	1.0000	0.9997
	B-QRNG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

5 Conclusions

Summarized, by running a naively designed quantum random number generator on real quantum hardware 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).

In two experiments we have studied their effect on the initialization of artificial neural network weights. 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 not be the "true randomness" of the quantum random

(a) Memory problem   **(b)** Adding problem

Figure 10: RNN convergence on two benchmark data sets using four different RNGs (B-QRNG, QRGN, PRGN and B-PRNG from section 4.1). Shown are mean values over 31 runs with the respective standard deviations (one sigma) in analogy to Fig. 9. The inset plot zooms in on the means of the final epochs.

numbers, but rather their bias 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 a corresponding bias. 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

References

Abraham H, AduOffei, Agarwal R et al. (2019) Qiskit: An open-source framework for quantum computing. DOI 10.5281/zenodo.2562110

ANU QRNG (2017) AARNnet cloudstor: pre-generated random binary numbers. `https://cloudstor.aarnet.edu.au/plus/s/9Ik6roa7ACFyWL4/ANU_3May2012_100MB`, accessed on April 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, DOI 10.1109/ETS.2018.8400697

Bell JS and Aspect A (2004) Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy, 2nd edn. Cambridge University Press, DOI 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, DOI 10.1007/BF01011339

Bera MN, Acín A, Kuś M et al. (2017) Randomness in quantum mechanics: philosophy, physics and technology. Reports on Progress in Physics 80(12):124001, DOI 10.1088/1361-6633/aa8731, URL `http://dx.doi.org/10.1088/1361-6633/aa8731`

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

Bird JJ, Ekárt A and Faria DR (2020) On the effects of pseudorandom and quantum-random number generators in soft computing. Soft Computing 24(12):9243–9256

Boixo S, Isakov SV, Smelyanskiy VN et al. (2018) Characterizing quantum supremacy in near-term devices. Nature Physics 14(6):595–600, DOI 10.1038/s41567-018-0124-x, URL `https://doi.org/10.1038/s41567-018-0124-x`

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

Bruzewicz CD, Chiaverini J, McConnell R and Sage JM (2019) Trapped-ion quantum computing: Progress and challenges. Applied Physics Reviews 6(2):021314, DOI 10.1063/1.5088164, URL `http://dx.doi.org/10.1063/1.5088164`

Calude CS, Dinneen MJ, Dumitrescu M and Svozil K (2010) Experimental evidence of quantum randomness incomputability. Physical Review A 82(2), DOI 10.1103/physreva.82.022102, URL `http://dx.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, DOI 10.1162/tacl_a_00018, URL `https://doi.org/10.1162/tacl_a_00018`, `https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00018/1567606/tacl_a_00018.pdf`

Forbes C, Evans M, Hastings N and Peacock B (2011) Statistical Distributions. John Wiley & Sons, Hoboken, New Jersey

Georgescu IM, Ashhab S and Nori F (2014) Quantum simulation. Rev Mod Phys 86:153–185, DOI 10.1103/RevModPhys.86.153, URL `https://link.aps.org/doi/10.1103/RevModPhys.86.153`

Glorot X and Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Teh YW and Titterington M (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, URL `http://proceedings.mlr.press/v9/glorot10a.html`

Grumbling E and Horowitz M (2019) Quantum Computing: Progress and Prospects. The National Academies Press, Washington, DC, DOI 10.17226/25196

Haw JY, Assad SM, Lance AM et al. (2015) Maximization of extractable randomness in a quantum random-number generator. Phys Rev Applied 3:054004, DOI 10.1103/PhysRevApplied.3.054004, URL `https://link.aps.org/doi/10.1103/PhysRevApplied.3.054004`

He K, Zhang X, Ren S and 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

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

Herrero-Collantes M and Garcia-Escartin JC (2017) Quantum random number generators. Reviews of Modern Physics 89(1), DOI 10.1103/revmodphys.89.015004, URL `http://dx.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 and Schmidhuber J (1997) Long Short-Term Memory. Neural Computation 9(8):1735–1780, DOI 10.1162/neco.1997.9.8.1735, URL `https://doi.org/10.1162/neco.1997.9.8.1735`, `https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf`

Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2):65–70, URL `http://www.jstor.org/stable/4615733`

Huang HL, Wu D, Fan D and Zhu X (2020) Superconducting quantum computing: A review. URL `https://arxiv.org/abs/2006.10433`, 2006.10433

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

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

Kofler J and Zeilinger A (2010) Quantum information and randomness. European Review 18(4):469–480, DOI 10.1017/s1062798710000268, URL `http://dx.doi.org/10.1017/S1062798710000268`

LaRose R (2019) Overview and Comparison of Gate Level Quantum Software Platforms. Quantum 3:130, DOI 10.22331/q-2019-03-25-130, URL `https://doi.org/10.22331/q-2019-03-25-130`

Lecun Y, Bottou L, Bengio Y and Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324, DOI 10.1109/5.726791

LeCun Y, Cortes C and 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, DOI 10.1063/5.0022526, URL `https://doi.org/10.1063/5.0022526`

Leymann F and Barzen J (2020) The bitter truth about gate-based quantum algorithms in the NISQ era. Quantum Science and Technology 5(4):044007, DOI 10.1088/2058-9565/abae7d, URL `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), DOI 10.3390/e22101134, URL `https://www.mdpi.com/1099-4300/22/10/1134`

Martínez AC, Solis A, Díaz Hernández Rojas R et al. (2018) Advanced statistical testing of quantum random number generators. Entropy 20(11), DOI 10.3390/e20110886, URL `https://www.mdpi.com/1099-4300/20/11/886`

Nachman B and Geller MR (2021) Categorizing readout error correlations on near term quantum computers. URL `https://arxiv.org/abs/2104.04607`, 2104.04607

Nielsen MA and Chuang IL (2011) Quantum Computation and Quantum Information: 10th Anniversary Edition, 10th edn. Cambridge University Press, USA

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 BG et al. (2020) A lightweight true random number generator using beta radiation for IoT applications. ETRI Journal 42(6):951–964, DOI https://doi.org/10.4218/etrij.2020-0119, URL `https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.2020-0119`, `https://onlinelibrary.wiley.com/doi/pdf/10.4218/etrij.2020-0119`

Pasqualini L and Parton M (2020) Pseudo random number generation: a reinforcement learning approach. Procedia Computer Science 170:1122–1127, DOI https://doi.org/10.1016/j.procs.2020.03.057, URL `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: Wallach H, Larochelle H, Beygelzimer A et al. (eds) Advances in Neural Information Processing Systems 32, Curran Associates, Inc., pp 8024–8035

Pearson K (1895) Notes on regression and inheritance in the case of two parents. In: Proceedings of the Royal Society of London, vol 58, pp 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, DOI 10.1080/14786440009463897

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

Pirandola S, Andersen UL, Banchi L et al. (2020) Advances in quantum cryptography. Advances in Optics and Photonics 12(4):1012, DOI 10.1364/aop.361502, URL `http://dx.doi.org/10.1364/AOP.361502`

Preskill J (2018) Quantum computing in the NISQ era and beyond. Quantum 2:79, DOI 10.22331/q-2018-08-06-79, URL `http://dx.doi.org/10.22331/q-2018-08-06-79`

Rajashekharan L and Shunmuga Velayutham C (2016) Is differential evolution sensitive to pseudo random number generator quality? – an investigation. In: Berretti S, Thampi SM and Srivastava PR (eds) Intelligent Systems Technologies and Applications, Springer International Publishing, Cham, pp 305–313

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

Shikano Y, Tamura K and Raymond R (2020) Detecting temporal correlation via quantum random number generation. Electronic Proceedings in Theoretical Computer Science 315:18–25, DOI 10.4204/eptcs.315.2, URL http://dx.doi.org/10.4204/EPTCS.315.2

Symul T, Assada SM and Lamb PK (2011) Real time demonstration of high bitrate quantum random number generation with coherent laser light. Appl Phys Lett 98:231103, DOI 10.1063/1.3597793, URL https://aip.scitation.org/doi/10.1063/1.3597793

Tamura K and Shikano Y (2021) Quantum random numbers generated by a cloud superconducting quantum computer. In: Takagi T, Wakayama M, Tanaka K et al. (eds) International Symposium on Mathematics, Quantum Theory, and Cryptography, Springer Singapore, Singapore, pp 17–37

Vacchini B (2016) Quantum noise from reduced dynamics. Fluctuation and Noise Letters 15(03):1640003, DOI 10.1142/s0219477516400034, URL http://dx.doi.org/10.1142/S0219477516400034

Virtanen P, Gommers R, Oliphant TE et al. (2020) SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17:261–272, DOI 10.1038/s41592-019-0686-2

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

Welch BL (1947) The generalization of 'student's' problem when several different population variances are involved. Biometrika 34(1-2):28–35, DOI 10.1093/biomet/34.1-2.28, URL https://doi.org/10.1093/biomet/34.1-2.28, https://academic.oup.com/biomet/article-pdf/34/1-2/28/553093/34-1-2-28.pdf

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

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, DOI 10.1155/2019/2545123, URL https://doi.org/10.1155/2019/2545123

Zeiler MD (2012) ADADELTA: An adaptive learning rate method. URL https://arxiv.org/abs/1212.5701, 1212.5701

Zurek WH (2007) Decoherence and the transition from quantum to classical — revisited. In: Duplantier B, Raimond JM and Rivasseau V (eds) Quantum Decoherence: Poincaré Seminar 2005, Birkhäuser Basel, Basel, pp 1–31, DOI "10.1007/978-3-7643-7808-0_1, URL https://doi.org/10.1007/978-3-7643-7808-0_1