Soft Computing
https://doi.org/10.1007/s00500-019-04450-0

METHODOLOGIES AND APPLICATION

On the effects of pseudorandom and quantum-random number generators in soft computing

Jordan J. Bird[1] · **Anikó Ekárt**[1] · **Diego R. Faria**[1]

Abstract
In this work, we argue that the implications of pseudorandom and quantum-random number generators (PRNG and QRNG) inexplicably affect the performances and behaviours of various machine learning models that require a random input. These implications are yet to be explored in soft computing until this work. We use a CPU and a QPU to generate random numbers for multiple machine learning techniques. Random numbers are employed in the random initial weight distributions of dense and convolutional neural networks, in which results show a profound difference in learning patterns for the two. In 50 dense neural networks (25 PRNG/25 QRNG), QRNG increases over PRNG for accent classification at +0.1%, and QRNG exceeded PRNG for mental state EEG classification by +2.82%. In 50 convolutional neural networks (25 PRNG/25 QRNG), the MNIST and CIFAR-10 problems are benchmarked, and in MNIST the QRNG experiences a higher starting accuracy than the PRNG but ultimately only exceeds it by 0.02%. In CIFAR-10, the QRNG outperforms PRNG by +0.92%. The *n*-random split of a Random Tree is enhanced towards and new Quantum Random Tree (QRT) model, which has differing classification abilities to its classical counterpart, 200 trees are trained and compared (100 PRNG/100 QRNG). Using the accent and EEG classification data sets, a QRT seemed inferior to a RT as it performed on average worse by −0.12%. This pattern is also seen in the EEG classification problem, where a QRT performs worse than a RT by −0.28%. Finally, the QRT is ensembled into a Quantum Random Forest (QRF), which also has a noticeable effect when compared to the standard Random Forest (RF). Ten to 100 ensembles of trees are benchmarked for the accent and EEG classification problems. In accent classification, the best RF (100 RT) outperforms the best QRF (100 QRF) by 0.14% accuracy. In EEG classification, the best RF (100 RT) outperforms the best QRF (100 QRT) by 0.08% but is extremely more complex, requiring twice the amount of trees in committee. All differences are observed to be situationally positive or negative and thus are likely data dependent in their observed functional behaviour.

Keywords Quantum computing · Soft computing · Machine learning · Neural networks · Classification

1 Introduction

Quantum and classical hypotheses of our reality are individually definitive and yet are independently paradoxical, in that they are both scientifically verified though contradictory to one another. These concurrently antithetical, nevertheless infallible natures of the two models have enflamed debate between researchers since the days of Albert Einstein and Erwin Schrödinger during the early twentieth century. Though the lack of a *Standard Model of the Universe* continues to provide a problem for physicists, the field of Computer Science thrives by making use of both in classical and quantum computing paradigms since they are independently observable in nature.

Though the vast majority of computers available are classical, quantum computing has been emerging since the late twentieth century and is becoming more and more available for use by researchers and private institutions. Cloud platforms developed by industry leaders such as Google, IBM, Microsoft and Rigetti are quickly growing in resources

Communicated by V. Loia.

✉ Jordan J. Bird
birdj1@aston.ac.uk

Anikó Ekárt
a.ekart@aston.ac.uk

Diego R. Faria
d.faria@aston.ac.uk

[1] School of Engineering and Applied Science, Aston University, Birmingham, UK

Published online: 28 October 2019

and operational size. This rapidly expanding availability of quantum computational resources allows for researchers to perform computational experiments, such as heuristic searches or machine learning, but allows for the use of the laws of quantum mechanics in their processes. For example, for n computational bits in a state of entanglement, only one needs to be measured for all n bits to be measured, since they all exist in parallel or antiparallel relationships. Through this process, computational complexity has been reduced by a factor of n. Bounded-error quantum polynomial time (BQP) problems are a set of computational problems which cannot be solved by a classical computer in polynomial time, whereas a quantum processor has the ability with its different laws of physics.

Optimisation is a large multifield conglomeration of research, which is rapidly accelerating due to the growing availability of powerful computing hardware such as CUDA. Examples include ant colony optimisation inspired by the pheromone-dictated behaviour of ants Deng et al. (2019), orthogonal translations to derive a principle component analysis Zhao et al. (2019), velocity-based searches of particle swarms Deng et al. (2017), as well as entropy-based methods of data analysis and classification Zhao et al. (2018).

There are several main contributions presented by this research:

1. A comparison of the abilities of dense neural networks with their initial random weight distributions derived by pseudorandom and quantum-random methods.
2. An exploration of Random Tree models compared to *Quantum Random Tree* models, which utilise pseudo-random and quantum-random number generators in their generation, respectively.
3. A benchmark of the number of Random Trees in a Random Forest model compared to the number of Quantum Random Trees in a Quantum Random Forest model.
4. A comparison of the effects of pseudo- and true randomness in initial random weight distributions in computer vision, applied to deep neural networks and convolutional neural networks.

Although quantum, quantum-inspired and hybrid classical/quantum algorithms are explored, as well as the likewise methods for computing, the use of a Quantum Random Number Generator is rarely explored within a classical machine learning approach in which an RNG is required Kretzschmar et al. (2000).

This research aims to compare approaches for random number generation in soft computing for two laws of physics which directly defy one another: the classical *true randomness is impossible* and the quantum *true randomness is possible* Calude and Svozil (2008). Through the application of both classical and quantum computing, simulated random number generation and true random number generation are tested and compared via the use of a central processing unit (CPU) and an electron spin-based quantum processing unit (QPU) via placing the subatomic particle into a state of quantum superposition. Logic would conjecture that the results between the two ought to be indistinguishable from one another, but experimentation within this study suggests otherwise. The rest of this article is structured as follows.

Section 2 gives an overview of the background to this project and important related theories and works: specifically, quantum computing, the differing ideas of randomness in both classical and quantum computing, applications of quantum theory in computing and finally a short subsection on the machine learning theories used in this study. Section 3 describes the configuration of the models as well as the methods used specifically to realise the scientific studies in this article, before being presented and analysed in Sect. 4. The experimental results are divided into four individual experiments:

- Experiment 1—On random weight distribution in Dense Neural Networks: pseudorandom and quantum-random number generators are used to initialise the weights in Neural Network models.
- Experiment 2—On Random Tree splits: The n Random Splits for a Random Tree classifier are formed by pseudorandom and quantum-random numbers.
- Experiment 3—On Random Tree splits in Random Forests: The *Quantum Tree* model derived from Experiment 2 is used in a *Quantum Random Forest* ensemble classifier.
- Experiment 4—On Computer Vision: A Deep Neural Network and Convolutional Neural Network are trained on two image recognition data sets with pseudorandom and true-random weight distributions for the application of Computer Vision.

Experiments are separated in order to focus upon the effects of differing random number generators on a specific model. Explored in these are the effects of pseudorandom and quantum-random number generation in their processes, and a discussion of similarities and differences between the two in terms of statistics as well as their wider effect on the classification process. Section 5 outlines possible extensions to this study for future works, and finally, a conclusion is presented in Sect. 6.

2 Background and related works

2.1 Quantum computing

Pioneered by Paul Benioff's 1980 work Benioff (1980), quantum computing is a system of computation that makes

computational use of phenomena outside of classical physics such as the entanglement and superposition of subatomic particles Gershenfeld and Chuang (1998). Whereas classical computing is concerned with electronic bits that have values of 0 or 1 and logic gates to process them, quantum computing uses both classical bits and gates as well as new possible states, such as a bit being in a state of superposition (0 and 1) or entangled with other bits. Entanglement means that the value of the bit, even before measurement, can be assumed to be parallel or antiparallel to another bit of which it is entangled to Bell (1964). These extended laws allow for the solving of problems far more efficiently than computers. For example, a 64-bit system ($2^{63}-1$) has approximately 9.22 quintillion values with its individual bits at values 1 or 0, whereas unlike a three-state ternary system which QPUs are often mistaken for, the laws of superposition and the degrees of state would allow a small array of qubits to represent all of these values at once—theoretically allowing quantum computers to solve problems that classical computers will never be able to possibly solve. Since the stability of entanglement decreases with the more computational qubits used, only very small-scale experiments have been performed as of today. Quantum Processing Units (QPUs) made available for use by Rigetti, Google and IBM have up to 16 available qubits for computing via their cloud platforms.

2.2 Randomness in classical and quantum computing

In classical computing, randomness is not random; rather, it is simulated by a *pseudorandom* process. Processor architectures and Operating Systems have individual methods of generating pseudorandom numbers which must conform to cybersecurity standards such as *NIST* Barker and Kelsey (2007). Major issues arise with the possibility of *backdoors*, notably, for example, Intel's pseudorandom generator which, after hijacking, allowed for complete control of a computer system for malicious intent Degabriele et al. (2016), Schneier (2007). The Intel issue was far from a lone incident, the RANDU system was cracked by the NSA for unprecedented access to the RSA BSAFE cryptographic library, as well as in 2006 when Debian OpenSSL's random number generator was also cracked, leading to Debian being compromised for 2 years Markowsky (2014). Though there are many methods of deriving a pseudorandom number, all classical methods, due to the laws of classical physics providing limitation, are sourced through arbitrary yet deterministic events Gallego et al. (2013), such as a combination of time since n last key press, hardware temperature, system clock and lunar calendar. This arbitration could possibly hamper or improve algorithms that rely on random numbers, since the state of the executing platform could indeed directly influence their behaviour.

According to Bell's famous theorem, *"no physical theory of local hidden variables can ever reproduce all of the predictions of quantum mechanics"* Bell (1964). This directly argued against the position put forward by Einstein et. al in which it is claimed that the Quantum Mechanical 'paradox' is simply due to incomplete theory Einstein et al. (1935). Using Bell's theorem, demonstrably random numbers can be generated through the fact that observing a particle's state while in superposition gives a true 50/50 outcome (qubit value 0, 1) Pironio et al. (2010). This concretely random output for the value of the single bit can be used to build integers comprised of larger numbers of bits which, since they are all individually random, their product is too. This process is known as a Quantum Random Number Generator (QRNG).

Behaviours in Quantum Mechanics such as, but not limited to, branching path superposition Jennewein et al. (2000), time of arrival Wayne et al. (2009), particle emission count Ren et al. (2011), attenuated pulse Wei and Guo (2009) and vacuum fluctuations Gabriel et al. (2010) are all entirely random—and have been used to create true QRNGs. In 2000, it was observed that a true random number generator could be formed through the observation of photons Stefanov et al. (2000). Firstly, a beam of light is split into two streams of entangled photons and noise is reduced after which the photons of both streams are observed. The two detectors correlate to 0 and 1 values, and a detection will amend a bit to the result. The detection of a photon is nondeterministic between the two, and therefore a completely random series of values are the result of this experiment.

This study makes use of the branching path superposition method for the base QRNG, in that the observed state of a particle c at time t and the state of c are nondeterministic until only after observation t. In the classical model, the law of superposition simply states that for properties A and B with outcomes X and Y, both properties can lead to state XY. For example, the translation and rotation of a wheel can lead to a rolling state Cullerne (2000), a third superstate of the two possible states. This translates into quantum physics, where quantum states can be superposed into an additional valid state Dirac (1981).

This is best exemplified with Erwin Schrödinger's famous thought experiment, known as *Schrödinger's Cat* Schrödinger (1935). As seen in Fig. 1, a cat sits in a box along with a Geiger Counter and a source of radiation. If alpha radiation is detected, which is a completely random event, the counter releases a poison into the box, killing the cat. The thought experiment explains superposition in such a way that although the cat has two states (Alive or Dead), when unobserved, the cat is both simultaneously alive and dead. In terms of computing, this means that the two classical behaviours of a single bit, 1 or 0, can be superposed into an additional state, *1 and 0. Just as the cat only becomes alive or dead when*

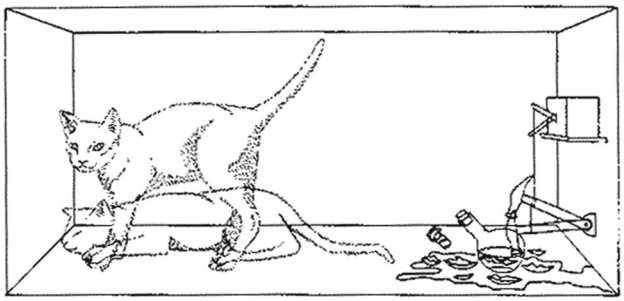

Fig. 1 Famous Schrödinger's Cat Thought Experiment. When unobserved, the cat arguably exists in two opposite states (alive and dead), which itself constitutes a third superstate Schrödinger (1935)

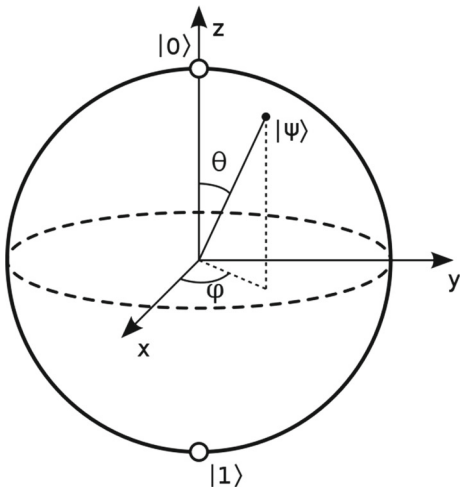

Fig. 2 A Bloch sphere represents the two basis states of a qubit (0, 1) as well as the states of superposition in between

observed, a superposed qubit only becomes 1 or 0 when measured.

A Bloch sphere is a graphical representation of a qubit in superposition Bloch (1946) and can be seen in Fig. 2. In this diagram, the basis states are interpreted by each pole, denoted as $|0\rangle$ and $|1\rangle$. Other behaviours, the rotations of spin about points ψ, ϕ and θ are used to superpose the two states to a degree. Thus, depending on the method of interpretation, many values can be encoded within only a single bit of memory.

The Hadamard gate within a QPU is a logical gate which coerces a qubit into a state of superposition based on a basis (input) state. 0 is mapped as follows:

$$|0\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{1}$$

The other possible basis state, 1, is mapped as:

$$|0\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2}$$

This single-qubit quantum Fourier transform is thus represented through the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3}$$

Just as in the thought experiment described in which Schrödinger's cat is both alive and dead, the qubit now exists in a state of quantum superposition; it is both 1 and 0. That is, until it is measured, in which there will be an equal probability that the observed state is 1 or 0, giving a completely randomly generated bit value. This is the logical basis of all QRNGs.

2.3 Quantum theory in related state-of-the-art computing application

The field of quantum computing is young, and thus there are many frontiers of research of which none have been mastered. Quantum theory, though, has been shown in some cases to improve current ideas in Computer Science as well as endow a system with abilities that would be impossible on a classical computer. This section outlines some of the state-of-the-art applications of quantum theory in computing.

Quantum Perceptrons are a theoretical approach to deriving a quantum equivalent of a perceptron unit (neuron) within an Artificial Neural Network Schuld et al. (2014). Current lines of research focus around the possibilities of associative memory through quantum entanglement of internal states within the neurons of the network. The approach is heavily inspired by the notion that the biological brain may operate within both classical and quantum physical space Hagan et al. (2002). Preliminary works have found Quantum Neural Networks have a slight statistical advantage over classical techniques within larger and more complex domains Narayanan and Menneer (2000). A very limited extent of research suggests quantum effects in a network to be the possible source of consciousness Hameroff and Penrose (1996), providing an exciting avenue for Artificial Intelligence research in the field of Artificial Consciousness. Inspiration from quantum mechanics has led to the implementation of a Neural Networks based on fuzzy logic systems Purushothaman and Karayiannis (1997), and research showed that QNNs are capable of structure recognition, which sigmoid-activated hidden units within a network cannot.

There are many statistical processes that are either more efficient or even simply possible through the use of Quantum Processors. Simon's Problem provides initial proof that there are problems that can be solved exponentially faster when executed in quantum space Arora and Barak (2009). Based on Simon's Problem, Shor's Algorithm uses quantum computing to derive the prime factors of an integer in polyno-

mial time Shor (1999), something which a classical computer is not able to do.

Some of the most prominent lines of research in quantum algorithms for soft computing are the exploration of Computational Intelligence techniques in quantum space such as meta-heuristic optimisation, heuristic search and probabilistic optimisation. Pheromone trails in Ant Colony Optimisation searches generated and measured in the form of qubits with operations of entanglement and superposition for measurement and state scored highly on the *Tennessee Eastman Process* benchmark problem, due to the optimal operations involved Wang et al. (2007). This work was applied by researchers, who in turn found that combining Support Vector Machines with Quantum Ant Colony Optimisation search provided a highly optimised strategy for solving fault diagnosis problems Wang et al. (2008), greatly improving the base SVM. Parallel Ant Colony Optimisation has also been observed to greatly improve in performance when operating similar techniques You et al. (2010). Similar techniques have also been used in the genetic search of problem spaces, with quantum logic gates performing genetic operations and probabilistic representations of solution sets in superposition/entanglement, and the technique is observed to be superior over its classical counterpart when benchmarked on the combinatorial optimisation problem Han et al. (2001).

Statistical and Deep Learning techniques are often useful in other scientific fields such as engineering Naderpour et al. (2019), Naderpour and Mirrashid (2019), medicine Khan et al. (2001), Penny and Frost (1996), chemistry Schütt et al. (2019), Gastegger et al. (2019) and astrophysics Krastev (2019), Kimmy Wu et al. (2019) among a great many others Carlini and Wagner (2017). As of yet, the applications of quantum solutions have not been applied within these fields towards the possible improvement of soft computing technique.

3 Experimental setup and design

For the generation of true random bit values, an electron-based superposition state is observed using a QPU. The Quantum Assembly Language code for this is given in Appendix A; an electron is transformed using a Hadamard gate and thus now exists in a state of superposition. When the bit is observed, it takes on a state of either 0 or 1, which is a nondeterministic 50/50 outcome, i.e. perfect randomness. A VM example of how these operations are formed into a random integer is given in Appendix B; the superposition state particle is sequentially observed, and each derived bit is amended to a result until 32 bits have been generated. These 32 bits are then treated as a single binary number. The result of this process is a truly random unsigned 32-bit integer.

For the generation of bounded random numbers, the result is normalised with the upper bound being the highest possible value of the intended number. For those that also have lower bounds below zero, a simple subtraction is performed on a higher bound of normalisation to give a range. For example, if a random weight distribution for neural network initialisation is to be generated between -0.5 and 0.5, the random 32-bit integer is normalised between 0–1 and 0.5 is subtracted from the result, giving the desired range. This process is used for the generation of both PRN and QRN since they are therefore then directly comparable with one another and thus also directly relative in their effects upon a machine learning process.

For the first data set in each experiment, a publicly available accent classification data set is retrieved.[1] This data set was gathered from subjects from the UK and Mexico, all speaking the same seven phonetic sounds ten times each. A flat data set is produced via 27 logs of their Mel-frequency Cepstral Coefficients every 200 ms to produce a mathematical description of the audio data. A four-class problem arises in the prediction of the locale of the speaker (West Midlands, London, Mexico City, Chihuahua). The second data set in each experiment is an EEG brainwave data set sourced from a previous study Bird et al. (2018).[2] The wave data have been extracted from the TP9, AF7, AF8 and TP10 electrodes and have been processed in a similar way to the speech in the first data set; exception is done so through a much larger set of mathematical descriptors. For the four-subject EEG data set, a three-class problem arises: the concentrative state of the subject (concentrating, neutral, relaxed). The feature generation process from this data set was observed to be effective for mental state classification in the aforementioned study, as well as for emotional classification from the same EEG electrodes Bird et al. (2019a).

For the final experiment, two image classification data sets are used. Firstly, the MNIST image data set is retrieved[3] LeCun and Cortes (2010) for the MLP. This data set is comprised of 60,000 32×32 handwritten single digits 0–9, a ten-class problem with each class being that of the digit written. Secondly, the CIFA-10 data set is retrieved[4] Krizhevsky et al. (2009) for a CNN. This, as with the MNIST data set, is comprised of 60,000 32×32 ten-class images of entities (eg. bird, cat, deer).

For the generation of pseudorandom numbers, an AMD FX8320 processor is used with given bounds for Experiments 1a and 1b. The Java Virtual Machine generates

[1] https://www.kaggle.com/birdy654/speech-recognition-dataset-england-and-mexico.

[2] https://www.kaggle.com/birdy654/eeg-brainwave-dataset-mental-state.

[3] http://yann.lecun.com/exdb/mnist/.

[4] https://www.cs.toronto.edu/kriz/cifar.html.

pseudorandom numbers for Experiments 2 and 3. All of the pseudorandom number generators had their seed set to the order of execution, ie. the first model has a seed of 1 and the nth model has a seed of n. Due to the high resource usage of training a large volume of neural networks, the CUDA cores of an Nvidia GTX980Ti were utilised and they were trained on a 70/30 train/test split of the data sets. For the Machine Learning Models explored in Experiments 2 and 3, tenfold cross-validation was used due to the availability of computational resources to do so.

3.1 Experimental process

In this subsection, a step-by-step process is given describing how each model is trained towards comparison between PRNG and QRNG methods. MLP and CNN RNG methods are operated through the same technique and as such are described together; following this, the Random Tree (RT) and Quantum Random Tree (QRT) are described. Finally, the ensembles of the two types of trees are then finally described as Random Forest (RF) and Quantum Random Forest (QRF). Each set of models is tested and compared for two different data sets, as previously described. For replicability of these experiments, the code for Random Bit Generation is given in Appendix A (for construction of an n-bit integer). Construction of the n-bit integer through electron observation loop is given in Appendix B.

For the Random Neural Networks, all use the ADAM Stochastic Optimiser for weight tuning Kingma and Ba (2014), and the activation function of all hidden layers is ReLU Agarap (2018). For Random Trees, K randomly chosen attributes are defined below (acquired via either PRNG or QRNG) and the minimum possible value for k is 1; no pruning is performed. Minimum class variance is set to $-inf$ since the data sets are well-balanced, the maximum depth of the tree is not limited and classification must always be performed even if confusion occurs. The chosen Random Tree attributes are also used for all trees within Forests, where the random number generator for selection of data subsets is also decided by a PRNG or QRNG. The algorithmic complexity for a Random Tree is given as $O(v \times nlog(n))$ where n is the number of data objects in the data set and v is the number of attributes belonging to a data object in the set. Algorithmic complexity of the neural networks is dependent on chosen topologies for each problem, and the complexity is presented as an $O(n^2)$ problem.

Given n number of networks to be benchmarked for x epochs, generally, the MLP and CNN experiments are automated as follows:

1. Initialise $n/2$ neural networks with initial random weights generated by an AMD CPU (pseudorandom).
2. Initialise $n/2$ neural networks with initial random weights generated by a Rigetti QPU (true random).
3. Train all n neural networks.
4. Consider classification accuracy at each epoch[5] for comparison as well as statistical analysis of all $n/2$ networks.

Given n number of trees with a decision variable K_x (K randomly chosen attributes at node x), the process of training Random Trees (RT) and Quantum Random Trees (QRT) is given as follows:

1. Train $n/2$ Random Trees, in which the RNG for deciding set K for every x is executed by an AMD CPU (pseudorandom)
2. Train $n/2$ Quantum Random Trees, in which the RNG for deciding set K for every x is executed by a Rigetti QPU (true random).
3. Considering the best and worst models, as well as the mean result, compare the two sets of $n/2$ models in terms of statistical difference.[6]

Finally, the Random Tree and Quantum Random Tree are benchmarked as an ensemble, through Random Forests and Quantum Random Forests. This is performed mainly due to the unpruned Random Tree likely overfitting to training data Hastie et al. (2005). The process is as follows:[7]

1. For the Random Forests, benchmark ten forests containing {10, 20, 30 ... 100} Random Tree Models (as generated in the *Random Tree Experimental Process* list above).
2. For the Quantum Random Forests, benchmark ten forests containing {10, 20, 30 ... 100} Quantum Random Tree Models (as generated in the *Random Tree Experimental Process* list above).
3. Compare abilities of all 20 models, in terms of classification ability as well as the statistical differences, *if any*, between different numbers of trees in the forest.

4 Results and discussion

In this section, results are presented and discussed for multiple Machine Learning models when their random number generator is either pseudorandomly or true (quantum) Randomly generated. Please note that in neural network training, lines do not correlate on a one-to-one basis. Each line is the accuracy of a neural network throughout the training

[5] Accuracy/epoch graphs are given in Sect. 4.

[6] Box-and-whisker comparisons given in Sect. 4.

[7] For further detail on the Random Decision Forest classifier selected for this study, please refer to Breiman (2001).

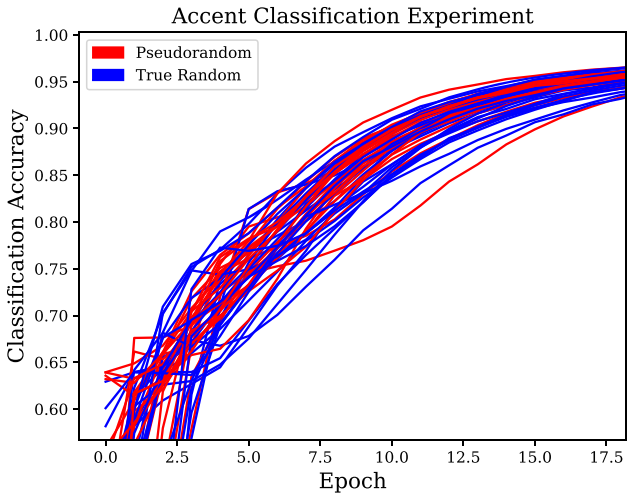

Fig. 3 Main learning curve experienced for 50 dense neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in accent classification

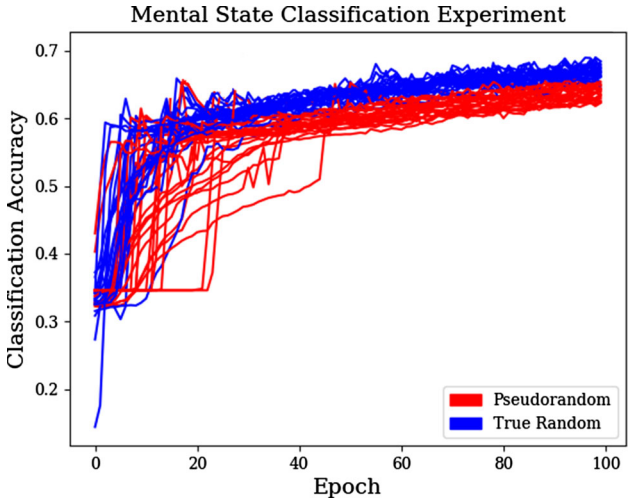

Fig. 4 Full learning process of 50 dense neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in mental state EEG classification

process, and line colour defines how that network had its weights initialised, i.e. whether or not it has pseudorandom or quantum-random numbers as its initial weights.

4.1 MLP: random initialisation of dense neural network weights

For Experiment 1, a total of 50 dense neural networks were trained for each data set. All networks were identical except for their initial weight distributions. Initial random weights within bounds of -0.5 and 0.5 were set, 25 of the networks derived theirs from a PRNG, and the other 25 from a QRNG.

4.1.1 Accent classification

For Experiment 1a, the accent classification data set was used. In this experiment, we observed initial sparse learning processes before stabilisation occurs at approximately epoch 30 and the two converge upon a similar result. Figure 3 shows this convergence of the learning processes the initial learning curve experienced during the first half of the process; in this graph, it can be observed that the behaviour of pseudorandom weight distribution is far less erratic than that of the quantum random number generator. This shows that the two methods of random number generators do have an observable effect on the learning processes of a neural network.

For PRNG, the standard deviation between all 25 final results was 0.00098 suggesting that a classification maximum was being converged upon. The standard deviation for QRNG was considerably larger, but statistically minimal at 0.0017. Mean final results were 98.73% for PRNG distributions and 98.8% for QRNG distributions. The maximum classification accuracy achieved by the PRNG initial distribution was 98.8%, whereas QRNG achieved a slightly higher result of 98.9% at epoch 49. For this problem, the differences between the initial distribution of PRNG and QRNG are minimal and QRNG distribution results are somewhat more entropic than PRNG but otherwise the two sets of results are indistinguishable from one another, and most likely simply due to random noise.

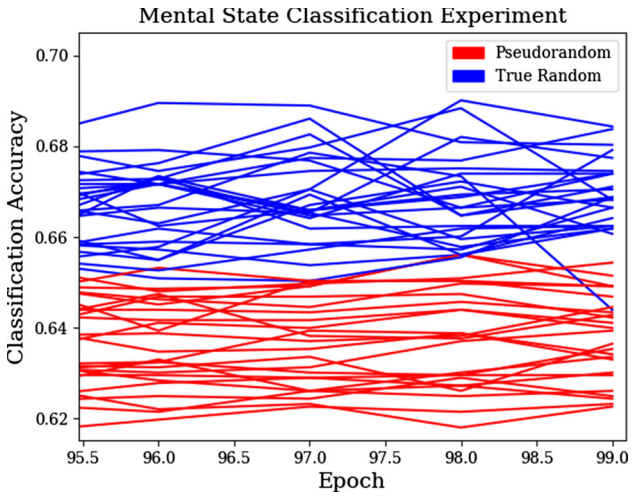

Fig. 5 Final epochs of learning for 50 dense neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in mental state EEG classification

4.1.2 Mental state classification

For Experiment 1b, the mental state EEG classification data set was used Bird et al. (2018). Figure 4 shows the full learning process of the networks from initial epoch 0 up until backpropagation epoch 100; though this graph is erratic and

crowded, the emergence of a pattern becomes obvious within epochs 20–30 where the learning processes split into two distinct groups. In this figure, a more uniform behaviour of QRNG methods is noted, unlike the previous experiment. The behaviours of PRNG distributed models are extremely erratic and in some cases, very slow in terms of improvements made. Figure 5 shows a higher resolution view of the data in terms of the end of the learning process when terminated at epoch 100, a clear distinction of results can be seen and a concrete separation can be drawn between the two groups of models except for two intersecting processes. It should be noted that by this point, the learning process has not settled towards a true best fitness, but a vast and clear separation has occurred.

For PRNG, the standard deviation between all 25 results was 0.98. The standard deviation for QRNG was somewhat smaller at 0.74. The mean of all results was 63.84% for PRNG distributions and 66.45% for QRNG distribution, a slightly superior result. The maximum classification accuracy achieved by the PRNG initial distribution was 65.35%, whereas QRNG achieved a somewhat higher best result of 68.17%. The worst–best result for PRNG distribution networks was 62.28% and was 65.31% for QRNG distribution networks. For this problem, the differences between the initial distribution of PRNG and QRNG weights are noticeable and QRNG distribution results are consistently better than PRNG approaches to initial weight distribution.

4.2 Random tree and quantum random tree classifiers

Experiments 2a and 2b make use of the same data sets as in 1a and 1b, respectively. In this experiment, 200 Random Tree classifiers are trained for each data set. These are, again, comprised of two sets; firstly 100 Random Tree (RT) classifiers which use pseudorandom numbers, and secondly, 100 *Quantum Random Tree* (QRT) classifiers, which source their random numbers from the QRNG. Random numbers are used to select the *n*-random attribute subsets at each split.

4.2.1 Accent classification

Two hundred experiments are graphically represented as a box-and-whisker in Fig. 6. The most superior classifier was the RT with a best result of 86.64% and worst of 85.68%; on the other hand, the QRT achieved the best accuracy of 86.52% and worst of 85.62%. Best and worst results of the two models are extremely similar. The standard deviation of results of the RT was 0.19, and the QRT similarly had a standard deviation of 0.17. The range of the RT results was 0.96, and QRT results had a similar range of 0.9. Interestingly, a similar pattern is not only found in results, but also with the high outlier too when considered relative to the model's median point.

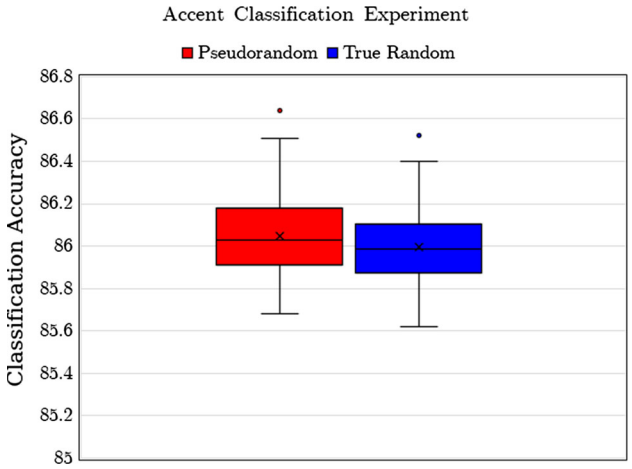

Fig. 6 A comparison of results from 200 Random Tree Classifiers, 100 using PRNG and 100 using QRNG on the accent classification data set

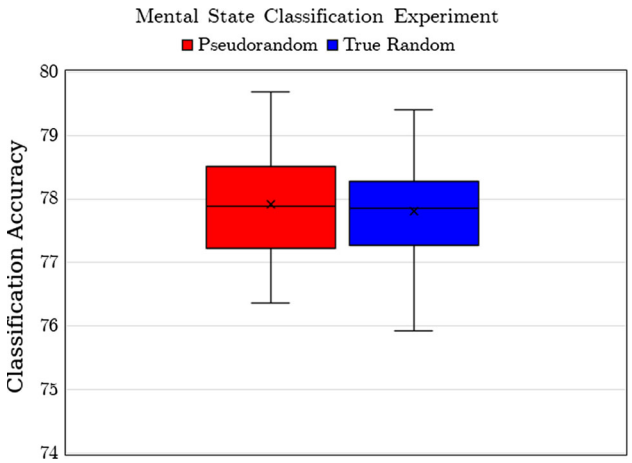

Fig. 7 A Comparison of results from 200 Random Tree Classifiers, 100 using PRNG and 100 using QRNG on the mental state EEG data set

Though an overall slight superiority is seen in pseudorandom number generation, the two models are considerably similar in their abilities.

4.2.2 Mental state classification

Figure 7 shows the distribution for the 200 Random Tree classifiers trained on the mental state data set. The standard deviation of results from the RT was 0.81, whereas it was slightly lower for QRT at 0.73. The best result achieved by the RT was 79.68% classification accuracy, whereas the best result from the QRT was 79.4%. The range of results for RT and QRT was similarly 3.31 and 3.47, respectively. Overall, very little difference between the two models occurs. The distribution of results can be seen to be extremely similar to the first RT/QRT experiment when compared to Fig. 6.

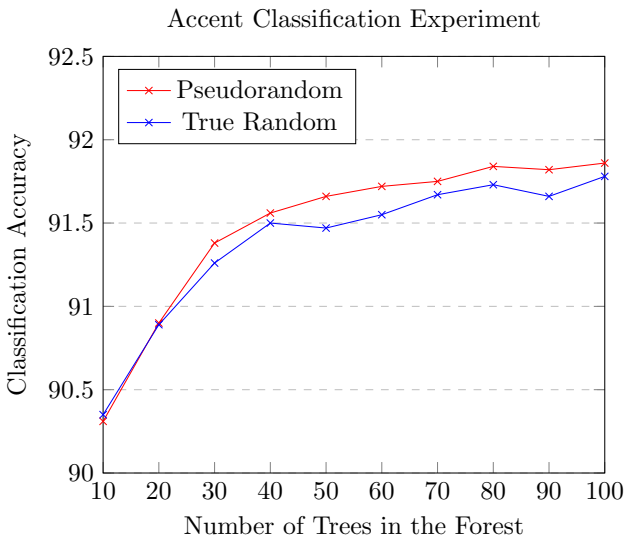

Fig. 8 Classification accuracies of ten Random Forest and ten Quantum Forest Models on the accent classification data set

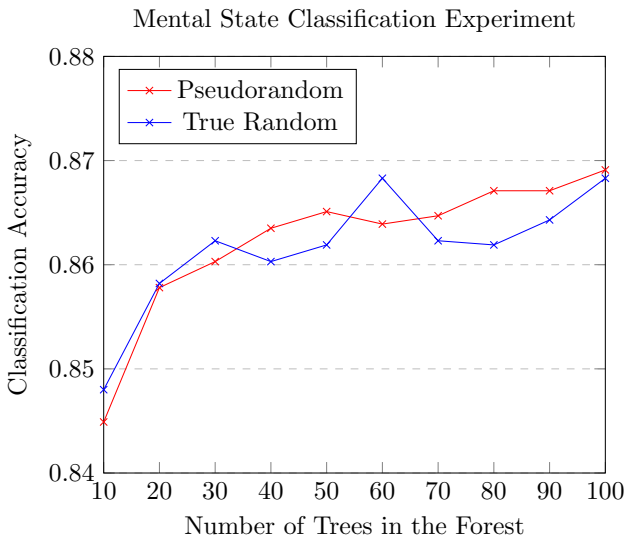

Fig. 9 Classification accuracies of ten Random Forest and ten Quantum Forest Models on the EEG mental state classification data set

4.3 Random forest and quantum random forest classifiers

In this third experiment, the data sets are classified using two models: Random Forests (RF) which use a committee of Random Trees to vote on a Class and Quantum Random Forests (QRF) which use a committee of Quantum Trees to vote on a class. For each data set, ten of these models are trained, with a committee of 10 to 100 trees respectively.

4.3.1 Accent classification

The results from the accent classification data set for the RF and QRF methods can be observed in Fig. 8. The most superior models both used a committee of 100 of their respective trees, scoring two similar results of 91.86% with pseudorandomness and 91.78% for Quantum randomness. Standard deviation of RF results is 0.5%, whereas QRF has a slightly lower deviation of 0.43. The worst result by RF was 90.31% classification accuracy at ten Random Trees, and the worst result by the QRF was similarly ten Quantum Trees at 90.36% classification accuracy (+0.05). The range of RF results was 1.55, compared to the QRF results with a range of 1.43.

4.3.2 Mental state classification

The results from the mental state EEG classification data set for the RF and QRF methods can be observed in Fig. 9. The most superior model for the RF was 86.91% with a committee of 100 trees, whereas the best result for QRF was 86.83% achieved by committees of both 100 and 60 trees. The range of QRF results was slightly lower than that of the RF, measured at 2.34 and 2.42, respectively. Although initially considered negligible, this same pattern was observed in the previous experiment in Fig. 8. Additionally, the standard deviation of RF was higher at 0.69 compared to 0.65 in QRF.

Though very similar results were produced, the first QRF best result required approximately 60% of the computational resources to achieve compared to the best RF result. Unlike the first forest experiment, the patterns of the two different models are vastly different and often alternate erratically. This suggests somewhat that the two models should both be benchmarked in order to increase the chances of discovering a more superior model, considering the level of data dependency on the classification accuracies of the models.

4.4 CNN: initial random weight initialisation for computer vision

Experiments 4a and 4b make use of the MNIST and CIFAR-10 image data sets, respectively. In 4a, an ANN is initialised following the same PRNG and QRNG methods utilised in Experiment 1 and trained to classify the MNIST handwritten digits data set. In 4b, the final dense layer of the CNN is initialised through the same methods.

4.4.1 MNIST image classification

For the purpose of scientific recreation, the architecture for MNIST classification is derived from the official Keras example.[8] This is given as two sets of two identical layers, a hidden layer of 512 densely connected neurons followed by

[8] https://github.com/keras-team/keras/tree/master/examples.

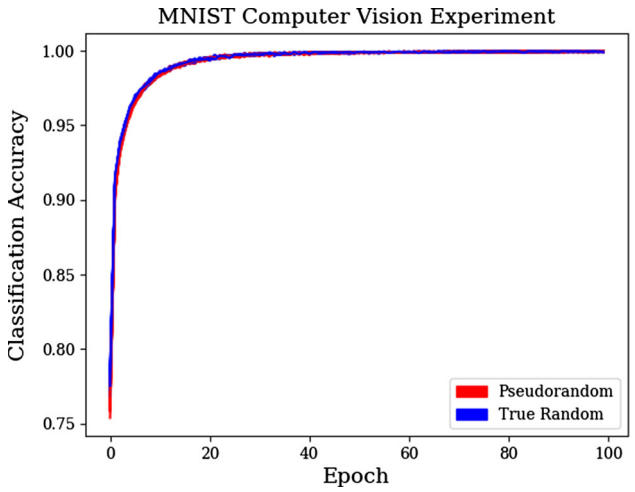

Fig. 10 Full learning process of 50 deep neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in MNIST image data set classification

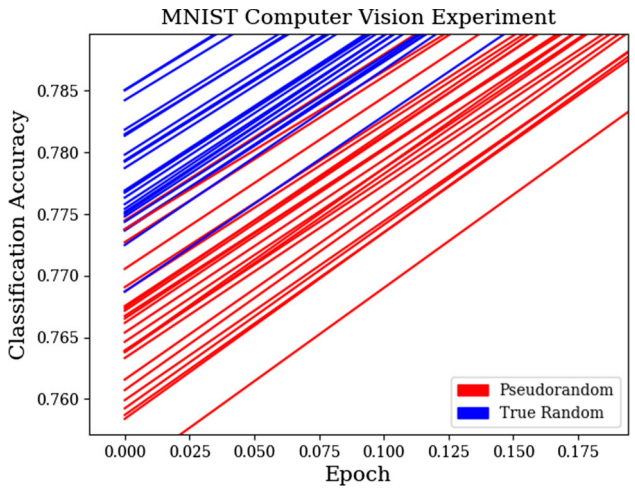

Fig. 11 Initial (pretraining) classification abilities of 50 deep neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in MNIST image data set classification

a dropout layer of 0.2 to prevent overfitting. All hidden neurons, as with other experiments in this study, are initialised randomly within the standard −0.5 to 0.5 range. Twenty-five of these are generated by a PRNG and the other 25 by a QRNG, producing observable results of 50 models in total.

Due to the concise nature and close results observed in the full process shown in Fig. 10, two additional graphs are presented; firstly, the graph in Fig. 11 shows the classification abilities of the models before any training occurs. Within this, a clear distinction can be made and the starting weights generated by QRNG are almost exclusively superior to those generated by PRNG, providing the QRNG models with a superior starting point for learning. The distinction continues to occur throughout the initial learning curve, observed in Fig. 12, not too dissimilar to the results in the previous experiment. At the pretraining abilities of

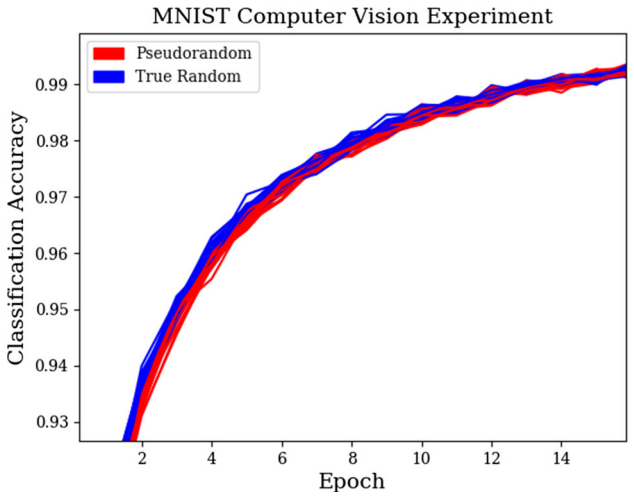

Fig. 12 Initial learning curve experienced for 50 deep neural networks, 25 with PRNG and 25 with QRNG initially distributed weights in MNIST image data set classification

the two methods of weight initialisation, dense areas can be observed at approx 77.5% Finally, at around epochs 10–14, the resultant models begin to converge and the separation becomes less prominent. This is shown through both sets of models having identical best classification accuracies of 98.64%m suggesting a true best fitness may possibly have been achieved. Worst–best accuracies are also indistinguishably close, 98.27% for QRNG models and 98.25% for PRNG models, population fitnesses are extremely dense and little entropy exists throughout the whole set of final results.

4.4.2 CIFAR-10 image classification

In the CNN experiment, the CIFAR-10 image data set is used to train a Convolutional Neural Network. The two number generators are applied for the initial random weight distribution of the final hidden dense layer, after feature extraction has been performed by the CNN operations. The network architecture is constructed as the official Keras Development Team example for Scientific purposes in ease of recreation of the experiment. In this architecture, one hidden dense layer of 512 units precedes the final classification output, and weights are generated within the bounds of −0.5 and 0.5 as is a standard in neural network generation. Fifty CNNS are trained, all of which are structurally identical except for that 25 have their dense layer weights initialised by PRNG and the other 25 have their dense layer weights initialised by QRNG.

Figure 13 shows the full learning process of the two different methods of initial weight distribution. It can be observed that there are roughly three partitions of results between the two methods, and the pattern is visually similar to the ANN learning curve in the MNIST Computer Vision experiment. Figure 14 shows the pretraining classification abilities of the initial weights; distribution is relatively equal and unre-

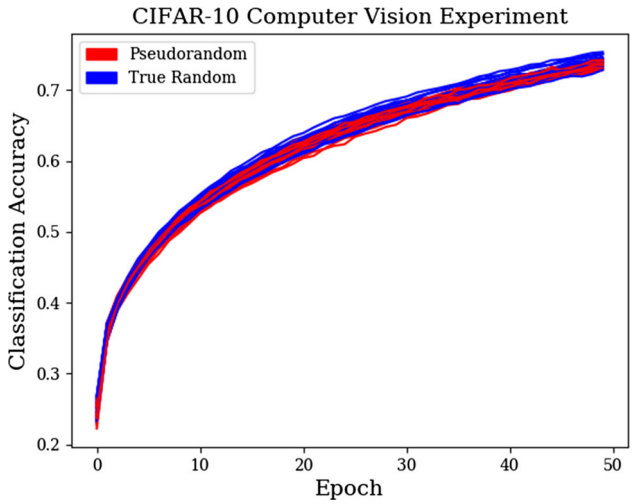

Fig. 13 Full learning process of 50 convolutional neural networks, 25 with PRNG and 25 with QRNG initially distributed weights for the final hidden dense layer in CIFAR-10 image data set classification

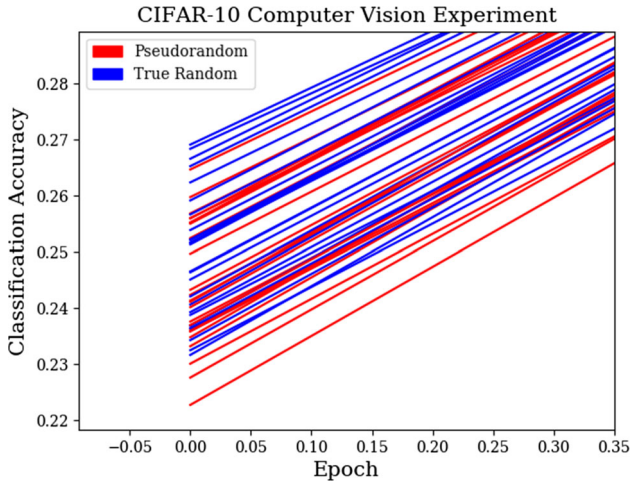

Fig. 14 Initial (pretraining) classification abilities of 50 convolutional neural networks, 25 with PRNG and 25 with QRNG initially distributed weights for the final hidden dense layer in CIFAR-10 image data set classification

markable unless compared to the final results of the training process in Fig. 15; the four best initial distributions of network weights, all are of that which have been generated by the QRNG, continue to be the four superior overall models. It must be noted, however, that the rest of the models regardless of RNG method are extremely similar and no other divide is seen by the end of the process.

The six overall most superior models were all initialised by QRNG, the best result being a classification accuracy of 75.35% at epoch 50. The seventh best model was the highest scoring model that had dense layer weights initialised by PRNG, scoring a classification accuracy of 74.43%. The worst model produced by the QRNG was that which had a classification accuracy of 71.91%, slightly behind this was the overall worst model from all experiments, a model initialised by the PRNG with an overall classification ability of 71.82%. The QRNG initialisation therefore outperformed PRNG by 0.92 in the best case and outperformed PRNG by 0.09 in the worst case. The average result between the two models was equal, at 73.3% accuracy.

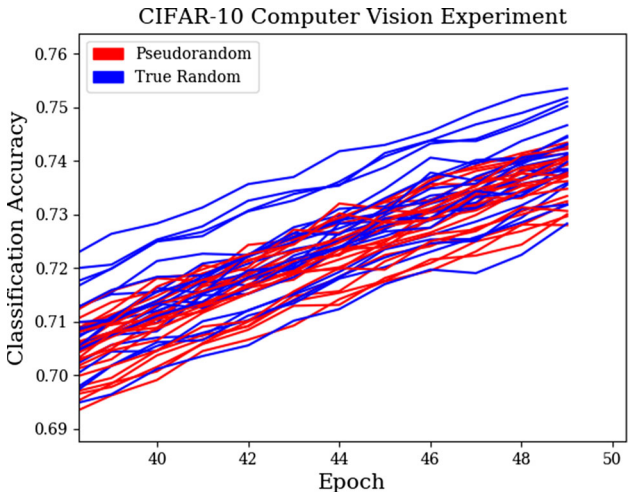

Fig. 15 Learning within the final epochs for 50 convolutional neural networks, 25 with PRNG and 25 with QRNG initially distributed weights for the final hidden layer in CIFAR-10 image data set classification

It must be noted that by epoch 50 the training process was still producing increasingly better results, but computational resources available limited the 50 networks to be trained for this amount of time.

5 Future work

It was observed in those experiments that did stabilise, results as expected reached closer similarities. With resources, future work should concern the further training of models to observe this pattern with a greater reach of examples. Extensive computational resources would be required to train such an extensive amount of networks.

Furthermore, the patterns in Fig. 9, Quantum vs Random Forest for mental state classification, suggest that the two forests have greatly different situational classification abilities and may produce a stronger overall model if both used in an ensemble. This conjecture is strengthened through a preliminary experiment; a vote of maximum probability between the two best models in this experiment (QF(60) and RF(100)) produces a result of 86.96% which is a slight and yet superior classification ability. The forests ensembled with other forests of their on type on the other hand do not improve. With this discovery, a future study should consider ensemble methods between the two for both deriving a stronger overall classification process and exploring the patterns in the ensemble of QRNG- and PRNG-based learning techniques. This, at

the very minimum, would require the time and computational resources to train 100 models to explore the two sets of ten models produced in the related experiment, though exploring beyond this, or even a full brute force of each model increasing their population of forests by 1 rather than 10 would produce a clearer view of the patterns within.

Of the most noticeable effects of QRNG and PRNG in machine learning, many of the 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 for each neuron within hidden layers. Following this, further types of neural network approaches should be explored to observe the similarities and differences that occur. In addition to this, the architectures of networks are by no means at an optimum, the heuristic nature of the network should also be explored, by techniques such as a genetic search, for it too requires the idea of random influence Bird et al. (2019b, c).

6 Conclusion

To conclude, this study performed eight individual experiments to observe the effects of quantum and pseudorandom number generators when applied to multiple machine learning techniques. Some of the results were somewhat unremarkable as expected, but some effects presented profound differences between the two, many of which are as of yet greatly unexplored. Based on these effects, possibilities of future work have been laid out in order to properly explore them.

Though observing superposition provides perfectly true randomness, this also provides a scientific issue in the replication of experiments since results cannot be coerced in the same nature a PRNG can through a seed. In terms of cybersecurity, this nature is ideal Yang et al. (2014), Stipcevic (2012), but provides frustration in a research environment since only generalised patterns at time t can be analysed Svore and Troyer (2016). This is overcome to an extent by the nature of repetition in the given experiments; many countless classifiers are trained to provide a more average overview of the systems.

The results for all of these experiments suggest that data dependency leads to no concrete positive or negative effect conclusion for the use of QRNG and PRNG since there is no clear superior method. Although this is true, pseudorandomness on modern processors is argued to be indistinguishable from true randomness, but clear patterns have emerged between the two. The two methods do inexplicably produce different results to one another when employed in machine learning, an unprecedented, and as of yet, relatively unexplored line of scientific research. In some cases, this was observed to be a relatively unremarkable, small and possibly coincidental difference, but in others, a clear division separated the two.

The results in this study are indicative of a profound effect on patterns observed in machine learning techniques when random numbers are generated either by the rules of classical or quantum physics. Their effects being positive or negative are seemingly dependent on the data at hand, but regardless, the fact that two methods of randomness ostensibly cause such disparate effects juxtaposed with the current scientific processes of their usage should not be underestimated. Rather, it should be explored.

Acknowledgements The authors would like to thank *Rigetti Computing* for granting access to their quantum computing platform.

Compliance with ethical standards

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

Appendices

1 Quantum assembly language for random number generation

Note: Code comments (#) are **not** Quantum Assembly Language and are simply for explanatory purposes. The following code will place a quanta into superposition via the Hadamard gate and then subsequently measure the state and store the observed value. The state is equally likely to be observed at either 1 or 0.

```
#Electron zero to Hadamard gate
H 0
#Declare memory space 'ro' of one bit
DECLARE ro BIT[1]
#Measure the qubit at 0th index of 'ro'
MEASURE 0 ro[0]
```

2 Python code for generating a string of random bits

The following code generates a random 32-bit integer by observing an electron in superposition which produces a true random result of either 1 or 0. The result is amended at each individual observation until 32 bits have been gen-

erated. Decimal conversion takes place, and two files are generated, a raw text file containing the decimal results and a CSV containing a column of binary integers and their decimal equivalents.

```
from pyquil.quil import Program
from pyquil.gates import H

# Select the lattice of Qubits
lattice = "Aspen-1-5Q-B"
# Initialise QPU
qpu = get_qc(lattice)

#Place electron 0 into superposition
numbers = Program(H(0))
#Observe the superposition
getNum = numbers.measure_all()
#Print the Quantum Assembly Language
print(getNum)

compiled_program = qpu.compile(numbers)

#Length of integer to generate
numbers = 32
#How many integers to generate
toGenerate = 1

print("\n Random number of " + str(numbers) +
    " bits:")


for y in range(0, 10000):
    output = ""
    for x in range(0, toGenerate):

        #Run the code on a Quantum Processing
            Unit
        result = qpu.run(compiled_program)
        #Observe the superposition
        result = result[0][0]

        output += str(result)

    print("\n\n Random no." + str(y) + " is: "
        + output)
    decimal = int(output, 2)

    with open("numbers.txt", "a") as myfile:
        myfile.write("\n" + str(decimal))

    with open("random.csv", "a") as myfile:
        myfile.write("\n" + str(output) + ","
            + str(decimal))
```

References

Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375

Arora S, Barak B (2009) Computational complexity: a modern approach. Cambridge University Press, Cambridge

Barker EB, Kelsey JM (2007) Recommendation for random number generation using deterministic random bit generators (revised). US Department of Commerce, Technology Administration, National Institute of .

Bell JS (1964) On the Einstein Podolsky Rosen paradox. Physics Physique Fizika 1(3):195

Benioff P (1980) The computer as a physical system: a microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. J Sat Phys 22(5):563–591

Bird JJ, Ekart A, Buckingham CD, Faria DR (2019a) Mental emotional sentiment classification with an EEG-based brain-machine interface. In: The international conference on digital image and signal processing (DISP'19). Springer

Bird JJ, Ekart A, Faria DR (2019b) Evolutionary optimisation of fully connected artificial neural network topology. In: SAI computing conference 2019, SAI

Bird JJ, Faria DR, Manso LJ, Ekart A, Buckingham CD (2019c) A deep evolutionary approach to bioinspired classifier optimisation for brain-machine interaction. Complexity 2019. https://doi.org/10.1155/2019/4316548

Bird JJ, Manso LJ, Ribiero EP, Ekart A, Faria DR (2018) A study on mental state classification using EEG-based brain-machine interface. In: 9th international conference on intelligent systems. IEEE

Bloch F (1946) Nuclear induction. Phys. Rev 70(7–8):460

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Calude CS, Svozil K (2008) Quantum randomness and value indefiniteness. Adv Sci Lett 1(2):165–168

Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy (SP). IEEE, pp 39–57

Cullerne J (2000) The Penguin dictionary of physics. Penguin Books, London

Degabriele JP, Paterson KG, Schuldt JC, Woodage J (2016) Backdoors in pseudorandom number generators: Possibility and impossibility results. In: Annual international cryptology conference. Springer, pp 403–432

Deng W, Zhao H, Yang X, Xiong J, Sun M, Li B (2017) Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment. Appl Soft Comput 59:288–302

Deng W, Xu J, Zhao H (2019) An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access 7:20281–20292

Dirac PAM (1981) The principles of quantum mechanics, vol 27. Oxford University Press, Oxford

Einstein A, Podolsky B, Rosen N (1935) Can quantum-mechanical description of physical reality be considered complete? Phys Rev 47(10):777

Gabriel C, Wittmann C, Sych D, Dong R, Mauerer W, Andersen UL, Marquardt C, Leuchs G (2010) A generator for unique quantum random numbers based on vacuum states. Nat Photonics 4(10):711

Gallego R, Masanes L, De La Torre G, Dhara C, Aolita L, Acín A (2013) Full randomness from arbitrarily deterministic events. Nat Commun 4:2654

Gastegger M, Schütt K, Sauceda H, Müller KR, Tkatchenko A (2019) Modeling molecular spectra with interpretable atomistic neural networks. In: APS meeting abstracts

Gershenfeld N, Chuang IL (1998) Quantum computing with molecules. Sci Am 278(6):66–71

Hagan S, Hameroff SR, Tuszyński JA (2002) Quantum computation in brain microtubules: decoherence and biological feasibility. Phys Rev E 65(6):061901

Hameroff S, Penrose R (1996) Orchestrated reduction of quantum coherence in brain microtubules: a model for consciousness. Math Comput Simul 40(3–4):453–480

Han KH, Park KH, Lee CH, Kim JH (2001) Parallel quantum-inspired genetic algorithm for combinatorial optimization problem. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546). IEEE, vol 2, pp 1422–1429

Hastie T, Tibshirani R, Friedman J, Franklin J (2005) The elements of statistical learning: data mining, inference and prediction. Math Intell 27(2):83–85

Jennewein T, Simon C, Weihs G, Weinfurter H, Zeilinger A (2000) Quantum cryptography with entangled photons. Phys Rev Lett 84(20):4729

Khan J, Wei JS, Ringner M, Saal LH, Ladanyi M, Westermann F, Berthold F, Manfred S, Antonescu CR, Peterson C (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat Med 7(6):673

Kimmy Wu W, Trivedi S, Caldeira J, Avestruz C, Story K, Nord B (2019) DeepCMB: lensing reconstruction of the cosmic microwave background with deep neural networks. In: American astronomical society meeting abstracts# 233, vol 233

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Krastev PG (2019) Real-time detection of gravitational waves from binary neutron stars using artificial neural networks. arXiv preprint arXiv:1908.03151

Kretzschmar R, Bueler R, Karayiannis NB, Eggimann F (2000) Quantum neural networks versus conventional feedforward neural networks: an experimental study. In: Neural networks for signal processing X. Proceedings of the 2000 IEEE signal processing society workshop (Cat. No. 00TH8501). IEEE, vol 1, pp 328–337

Krizhevsky A, Nair V, Hinton G (2009) Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/~kriz/cifar.html

LeCun Y, Cortes C (2010) MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/

Markowsky G (2014) The sad history of random bits. J Cyber Secur Mobil 3(1):1–24

Naderpour H, Mirrashid M (2019) Shear failure capacity prediction of concrete beam-column joints in terms of ANFIS and GMDH. Pract Period Struct Des Constr 24(2):04019006

Naderpour H, Mirrashid M, Nagai K (2019) An innovative approach for bond strength modeling in FRP strip-to-concrete joints using adaptive neuro-fuzzy inference system. In: Engineering with computers, pp 1–18

Narayanan A, Menneer T (2000) Quantum artificial neural network architectures and components. Inf Sci 128(3–4):231–255

Penny W, Frost D (1996) Neural networks in clinical medicine. Med Decis Mak 16(4):386–398

Pironio S, Acín A, Massar S, de La Giroday AB, Matsukevich DN, Maunz P, Olmschenk S, Hayes D, Luo L, Manning TA (2010) Random numbers certified by bell's theorem. Nature 464(7291):1021

Purushothaman G, Karayiannis NB (1997) Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks. IEEE Trans Neural Netw 8(3):679–693

Ren M, Wu E, Liang Y, Jian Y, Wu G, Zeng H (2011) Quantum random-number generator based on a photon-number-resolving detector. Phys Rev A 83(2):023820

Schneier B (2007) Did NSA put a secret backdoor in new encryption standard. http://www.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115:2007

Schrödinger E (1935) Die gegenwärtige situation in der quantenmechanik. Naturwissenschaften 23(49):823–828

Schuld M, Sinayskiy I, Petruccione F (2014) The quest for a quantum neural network. Quantum Inf Process 13(11):2567–2586

Schütt K, Gastegger M, Tkatchenko A, Müller KR, Maurer R (2019) Unifying machine learning and quantum chemistry—a deep neural network for molecular wavefunctions. arXiv preprint arXiv:1906.10033

Shor PW (1999) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev 41(2):303–332

Stefanov A, Gisin N, Guinnard O, Guinnard L, Zbinden H (2000) Optical quantum random number generator. J Mod Opt 47(4):595–598

Stipcevic M (2012) Quantum random number generators and their applications in cryptography. In: Advanced photon counting techniques VI, international society for optics and photonics, vol 8375, p 837504

Svore KM, Troyer M (2016) The quantum future of computation. Computer 49(9):21–30

Wang L, Niu Q, Fei M (2008) A novel quantum ant colony optimization algorithm and its application to fault diagnosis. Trans Inst Meas Control 30(3–4):313–329

Wang L, Niu Q, Fei M (2007) A novel quantum ant colony optimization algorithm. In: International conference on life system modeling and simulation. Springer, pp 277–286

Wayne MA, Jeffrey ER, Akselrod GM, Kwiat PG (2009) Photon arrival time quantum random number generation. J Mod Opt 56(4):516–522

Wei W, Guo H (2009) Quantum random number generator based on the photon number decision of weak laser pulses. In: Conference on lasers and electro-optics/Pacific Rim, Optical Society of America, p TUP5_41

Yang YG, Jia X, Sun SJ, Pan QX (2014) Quantum cryptographic algorithm for color images using quantum fourier transform and double random-phase encoding. Inf Sci 277:445–457

You X, Liu S, Wang Y (2010) Quantum dynamic mechanism-based parallel ant colony optimization algorithm. Int J Comput Intell Syst 3(sup01):101–113

Zhao H, Yao R, Xu L, Yuan Y, Li G, Deng W (2018) Study on a novel fault damage degree identification method using high-order differential mathematical morphology gradient spectrum entropy. Entropy 20(9):682

Zhao H, Zheng J, Xu J, Deng W (2019) Fault diagnosis method based on principal component analysis and broad learning system. IEEE Access 7:99263–99272