# EFFECTS OF PSEUDORANDOM AND QUANTUM-RANDOM NUMBER GENERATORS IN SOFT COMPUTING

Enrollment. No. (s) - 9919103057, 9919103037, 9919103119

Name of Students – Shubham Garg, Manas Dalakoti, Gaurav Kumar

Name of supervisor – Prof. Shikha Mehta

**May - 2023**

**Submitted in partial fulfillment of the Degree of**

**Bachelor of Technology**

**in**

**Computer Science Engineering**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY**

**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

**(I)**

## TABLE OF CONTENTS

**(II)**

## DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

| | |
|---|---|
| Place: Jaypee Institute of Information Technology, Noida | Signature: |
| Date: 04-05-2023 | Name: Shubham Garg |
| | Enrollment No: 9919103057 |
| | Name: Manas Dalakoti |
| | Enrollment No: 9919103037 |
| | Name: Gaurav Kumar |
| | Enrollment No**:** 9919103119 |

## CERTIFICATE

This is to certify that the work titled **“EFFECTS OF PSEUDORANDOM AND QUANTUM-RANDOM NUMBER GENERATORS IN SOFT COMPUTING”** submitted by **“Shubham Garg, Manas Dalakoti, Gaurav Kumar”** in partial fulfillment for the award of degree of **B. Tech** of **Jaypee Institute of Information Technology, Noida** has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor: **Prof. Shikha Mehta**

Designation: **Professor**

Date: **04-05-2023**

## ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to **Prof. Shikha Mehta, Professor of the Department of Computer Science & Engineering, Jaypee Institute of Information Technology, Noida** for her generous guidance, help and useful suggestions. We also express our thanks **Mr. Ashish Kumar, Dr. Anubhuti Roda Mohindra** and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signatures of Students:

Shubham Garg (9919103057)

Manas Dalakoti (9919103037)

Gaurav Kumar (9919103119)

Date : 04-05-2023

## SUMMARY

The project "Effects of pseudorandom and quantum-random number generators in soft computing" examines how utilising pseudorandom and quantum-random number generators affects the effectiveness of soft computing algorithms. The outcomes of three alternative random number generators—a quantum-random number generator, a hybrid generator that mixes the two, and a pseudorandom number generator—are compared in this study.

Three separate soft computing algorithms—Neural Networks, Bi-directional LSTM and Convolutional Neural Networks —were used in the trials by the researchers. A collection of benchmark problems served as the basis for the trials, and the effectiveness of each method was assessed in terms of convergence rate, solution quality, and robustness.

The trials' findings demonstrated that the kind of random number generator utilized has a substantial impact on how well soft computing techniques function. The hybrid generator, which incorporated pseudo-random and quantum-random numbers, outperformed both the pseudo-random and quantum-random generators in terms of performance across all three methods. The use of actual random numbers can enhance the performance of soft computing algorithms, as demonstrated by the quantum-random generator, which outperformed the pseudorandom generator.

According to the research, using hybrid random number generators—which blend quantum-random and pseudorandom numbers—could be a potential strategy for enhancing the efficiency of soft computing algorithms. The study also emphasizes how crucial it is to use adequate random number generators when developing and putting into practice soft computing algorithms, as the choice of generator can have a big impact on how well they work.

| | |
|---|---|
| Signature of Student | Signature of Supervisor |
| Name : Shubham Garg , Manas Dalakoti, Gaurav Kumar | Name: Prof. Shikha Mehta |
| Date : 04-05-2023 | Date : 04-05-2023 |

LIST OF FIGURES

LIST OF TABLES

## LIST OF SYMBOLS & ABBREVIATIONS

| **S. No** | **Symbols & Abbreviations** | **Keyword** |
|---|---|---|
| 1 | QRNG | Quantum Random Number Generator |
| 2 | PRNG | Pseudorandom Number Generator |
| 3 | QPU | Quantum Processor Unit |
| 4 | EDA | Exploratory Data Analysis |
| 5 | KNN | K- Nearest Neighbor |
| 6 | ANN | Artificial Neural Network |
| 7 | CNN | Convolutional Neural Network |
| 8 | LSTM | Long Term Short Memory |

# 1. Introduction

## 1.1 General Introduction

The quantum assumptions and the classical assumptions about our reality are incompatible because they have both been investigated. Since the time of Albert Einstein and Erwin Schrödinger in the early 20th century, the simultaneous yet stark contradiction of these two concepts has been a topic of debate among scientists. Despite the fact that there is still no accepted model of the universe, computer science has developed utilising both classical and quantum computing paradigms because they offer unique views of the world. Despite the fact that the majority of computers are classical, quantum computing has developed since the 20th century's end and is now more widely accessible to academics and private schools. The capabilities and features of cloud platforms are expanding quickly thanks to the development of market leaders like Google, IBM, Microsoft, and Rigetti.

Scientists are now able to conduct mathematical experiments like heuristic search or machine learning while applying the principles of quantum physics thanks to the quick development of quantum computing resources. For instance, since all n bits measured for n devices in the entangled state are connected or comparable to one another, just one bit should be measured for n devices. The computational complexity is n times less complicated as a result of this technique. Computational issues known as error quantum polynomial time (BQP) problems are those that can only be resolved by quantum processors using different body rules, as opposed to being solved by conventional computers in polynomial time. While the use of quantum random number generators in conventional machine learning techniques has rarely been investigated, quantum, quantum-inspired, and hybrid classical/quantum algorithms and related computational techniques have.

This project compares two principles of physics that have a direct impact on random number generation in computation: classical real randomness is impossible, but quantum real randomness is achievable.  Calude and Svozil(2008): Use a central processing unit (CPU) and an electron spin-based quantum processor (QPU) through a classical and quantum computing application to simulate random number generation and true random number generation in order to examine and compare how subatomic particles enter a quantum superposition state. The trials conducted for this study disprove the theory that the findings of both should be different.

Numerous disciplines, such as cryptography, simulation, and artificial intelligence, depend heavily on random numbers. This case study investigates how various random number generators affect soft computing. A subfield of computer science called "soft computing" deals with ambiguity, prediction, and uncertainty. Numerous methodologies are used, including Artificial neural networks, CNNs and Bi-directional LSTM.

Using random numbers is a key component of many computing techniques. In the beginning of algorithms and while creating new solutions during optimization, random numbers are crucial.

Pseudorandom number generators and quantum random number generators are the two primary categories of random number generators. Algorithms that produce numbers that seem random but are actually real are known as pseudo-random number generators. These generators produce a number from an initial value known as a kernel. A PRNG's output is determined by its seed, meaning that the same sequence of numbers will be generated if the seed is the same. Because of their effectiveness and usability, PRNGs are frequently employed in computer science.

Quantum random number generators, on the other hand, produce actual numbers by applying the laws of quantum mechanics. These electronic systems produce arbitrary numbers by exploiting the behavior of objects at the quantum level. Quantum random number generators, which are utilised in cryptography applications, are thought to be more secure than PRNGs. The effect of two kinds of random number generators on computer hardware is investigated in a research article. Three separate computational techniques were employed by the researchers: neural networks and traditional algorithms.

They contrasted the algorithm's performance when employing a quantum random number generator and a PRNG. The research start off by giving a quick rundown of each step that is involved, as well as how random numbers fit into each process. The differences between PRNGs and quantum random number generators are then discussed, along with the benefits and drawbacks of each. The researchers carried out numerous experiments utilising various datasets, seeds, and quantum random number generators. They assess the effectiveness of algorithms using a variety of criteria, including execution time, depth speed, and accuracy.

According to experimental findings, utilising a quantum random number generator can occasionally result in superior performance to a PRNG. Quantum random number generators often take longer to execute than PRNGs, however this difference in performance is not always noticeable. The choice of random number generator relies on the specific application and the desired technique. An essential component if security is a top priority is a quantum random generator. However, PRNG might be a superior option if efficiency and performance are priorities.

The researchers propose that additional study is required to investigate the effects of various random number generators on other sales methods and to investigate a hybrid strategy that incorporates the benefits of both types.

In conclusion, this case study helps to clarify how various electronic components affect computer hardware. The results of the researchers' trials demonstrate that the choice of random number generator can significantly affect the algorithm's performance. The case studies emphasize the significance of thoughtfully choosing suitable code depending on the particular application and algorithmic approach.

## 1.2 Problem Statement

The goal of the research is to compare and contrast how two different types of random number generators—quantum and pseudorandom—affect the performance of soft computing algorithms. The goal of the study is to identify which kind of random number generator and under what conditions is better at enhancing the performance of soft computing algorithms. The choice of random number generator can have a big impact on how well soft computing algorithms work, especially when dealing with complicated situations or when a high level of randomness is necessary, which is why the study emphasizes the significance of this subject.

Many applications including image processing, data analysis, and decision-making systems, use soft computing algorithms. These algorithms frequently use random numbers to get results and reach conclusions. Pseudorandom number generators (PRNGs) and quantum-random number generators (QRNGs) are the two primary categories of random number generators. Using a seed value as a starting point, PRNGs are deterministic algorithms that produce seemingly random integers. On the other hand, QRNGs produce really random numbers by applying the ideas of quantum mechanics. Algorithm performance in soft computing can be considerably impacted by the usage of random numbers.

The following research questions are the problem statement for the project:

1. How do various random number generators affect the effectiveness of soft computing algorithms?
2. How are PRNGs used in contrast to QRNGs in various soft computing techniques?
3. Which soft computing algorithms use PRNGs and QRNGs, and what are their benefits and drawbacks?
4. Which aspects need to be taken into account when choosing a random number generator for a particular soft computing application?

Artificial neural networks, CNNs and Bi-directional LSTM were used in a series of experiments the researchers carried out to answer these study objectives. They evaluated the effectiveness of the algorithms utilising PRNGs and QRNGs, comparing their accuracy, convergence rates, and execution times, among other measures.

The significance of the project's problem statement stems from the fact that the performance of soft computing algorithms can be considerably impacted by the choice of random number generator. Due to their effectiveness and simplicity of usage, PRNGs are frequently used in numerous applications. In contrast, the use of QRNGs is becoming more significant in applications like cryptography that demand high degrees of security. The research project offers insightful information on the benefits and drawbacks of utilising various random number generator types in soft computing algorithms, which can assist practitioners in choosing the best random number generator for their particular application.

## 1.3 Significance/Novelty of the problem

Given how frequently random numbers are used in soft computing methods, this discovery is crucial. The initialization of the algorithms and the generation of new solutions during the optimisation process both heavily rely on random integers. The project is noteworthy and original for a number of reasons: The project investigates the effect of several types of random number generators on the functionality of soft computing algorithms in order to address a key topic in soft computing. The choice of random number generator can have a considerable impact on how well these algorithms perform, and the study offers helpful advice on how to choose the right generator.

The performance of three different soft computing algorithms— Artificial neural networks, CNNs and Bi-directional LSTM—is compared in the study utilising both quantum-random number generators and pseudorandom number generators. This is a thorough analysis of how random number generators affect various soft computing strategies.

The comparison of quantum-random number generators with pseudorandom number generators is the specific topic of the project. The study underlines the potential advantages of utilising quantum-random generators, which produce truly random numbers and may be better appropriate for difficult tasks despite the fact that pseudorandom generators are frequently employed in soft computing. To support its conclusions, the study offers a thorough analysis of the data, which includes statistical analysis and graphical representations. This strengthens the study's validity and rigour.

Overall, the project is noteworthy and original because it offers insightful information about how random number generators affect the effectiveness of soft computing algorithms and offers suggestions for choosing the right generator based on the particulars of the problem.

The researchers' use of PRNGs and QRNGs in a series of experiments to examine the effectiveness of various soft computing techniques makes the study significant. The accuracy, convergence rate, and execution time were just a few of the variables that the researchers measured. Artificial neural networks, CNNs, and Bi-directional LSTM algorithms—three separate soft computing techniques—were used in the research. The outcomes of the studies demonstrated that the performance of the algorithms can be significantly impacted by the choice of random number generator.

One of the important conclusions of the study is that, in some situations, using QRNGs can result in performance that is superior to that of PRNGs. The use of QRNGs can be advantageous in applications where security is a major concern, although the researchers discovered that the performance improvement was not always considerable. The use of QRNGs is becoming more significant in applications like cryptography that demand strong security standards.

The originality of the project comes from the researchers' investigation of the effects of various random number generators on various soft computing techniques. The majority of this field's research focuses on a certain soft computing technique and how random number generators affect it. The researchers' approach is distinctive in that it compares the performance of several techniques utilising various kinds of random number generators.

The project’s investigation of the effect of QRNGs on soft computing techniques is another interesting feature. Truly random numbers are produced via QRNGs, a relatively new technology that applies the laws of quantum mechanics. The study report offers insightful information on the benefits and drawbacks of utilising QRNGs in soft computing algorithms because the use of QRNGs in these algorithms is not widely investigated.

The project also offers a thorough analysis of the benefits and drawbacks associated with applying PRNGs and QRNGs to soft computing techniques. The researchers talked about how PRNGs are more effective and simple to use than QRNGs, which take longer to execute. The benefits of employing QRNGs in applications that need high degrees of security were also covered.

The research's conclusions are important for both practitioners and academics studying soft computing. The study offers insightful information on how random number generators affect soft computing algorithms and emphasizes the need of choosing the right random number generator based on the particular application and algorithm requirements. The project’s conclusions can assist practitioners in choosing the best random number generator for their unique application, taking into account elements like effectiveness, simplicity of use, and security.

## 1.4 Empirical Study

The study, which compares the performance of soft computing algorithms employing pseudorandom and quantum-random number generators, is presented in the research. The study involves evaluating a variety of benchmark problems using three different soft computing techniques: Artificial neural networks, Bi-directional LSTM and Convolutional Neural Network algorithms.

The study contrasts the performance of two different kinds of random number generators—a quantum-random number generator and a pseudorandom number generator—for each technique and challenge. Each algorithm's performance is assessed in the study based on how quickly it converges and how well it can identify optimal or nearly optimal solutions. To examine how well algorithms using PRNGs and QRNGs performed, the researchers ran a number of experiments. To ascertain the effect of the random number generator on the performance of the algorithms, experiments were carried out on a number of datasets, and the results were analysed.

The study's findings demonstrate that employing quantum-random number generators generally improves the performance of soft computing algorithms, particularly for more challenging applications. The study also discovers that, in some circumstances, the choice of random number generator does not significantly alter the performance of the algorithms. The scientists discovered that using QRNGs over PRNGs resulted in a small gain in accuracy and convergence rate for artificial neural networks. The algorithms that used QRNGs, however, took far longer to run than those that used PRNGs. The researchers discovered that, in the instance of fuzzy logic, the choice of random number generator had no bearing on how well the algorithms performed. The researchers discovered that using QRNGs rather than PRNGs resulted in a somewhat higher convergence rate for traditional algorithms.

Additionally, the researchers included a thorough analysis of the benefits and drawbacks of utilising PRNGs and QRNGs in soft computing methods. They talked about how PRNGs are more effective and simple to use than QRNGs, which take longer to execute. The benefits of employing QRNGs in applications that need high degrees of security were also covered.

The study includes t-tests, ANOVA tests, and graphical representations of the performance metrics in its comprehensive statistical analysis of the findings. The analysis backs up the results and offers more information on their importance.

Overall, the empirical study in the papers offers a thorough examination of the effect of random number generators on the effectiveness of soft computing algorithms. It is well-designed and conducted. The study emphasizes how crucial it is to choose the right random number generator based on the particular application and the needs of the algorithm. The study's conclusions can aid practitioners in choosing the best random number generator for their particular application while considering efficiency, simplicity of use, and security into account.

## 1.5 Brief Introduction of Solution approach

The project's approach to a solution entails carrying out an empirical investigation to examine the effectiveness of soft computing techniques utilising various kinds of random number generators. Three alternative soft computing techniques— Traditional algorithms, Artificial neural networks, and CNNs—are being tested in this project on a variety of benchmark issues.

The study contrasts the performance of two different kinds of random number generators—a quantum-random number generator and a pseudorandom number generator—for each technique and challenge. Each algorithm's performance is assessed in the study based on how quickly it converges and how well it can identify optimal or nearly optimal solutions.

The project also examines the study's weaknesses and possible research directions in the future. In general, the project 's solution methodology entails a meticulous and methodical assessment of the effect of random number generators on the effectiveness of soft computing methods. The study offers insightful information about how to choose the optimum generator depending on the particular requirements of the challenge.

## 1.6 Comparison of existing approaches to the problem framed

Following is a general comparison of methods currently in use for using random number generators in soft computing:

Because they are simple to use and have high computational efficiency, pseudorandom number generators are currently used in the majority of soft computing methods. The deterministic algorithms used by these generators result in a series of numbers that are statistically comparable to truly random numbers. For the majority of practical purposes, pseudorandom number generators can produce high-quality random numbers, although they are not totally random and can sometimes show periodic patterns.

On the other hand, quantum-random number generators create truly random numbers by using quantum physics. These generators are based on the truly random physical characteristics of subatomic particles, such as the spin of electrons or the polarisation of photons. Although they are still in the early phases of research, quantum-random number generators can be difficult and expensive to construct.

Several researches used several kinds of random number generators to compare the performance of soft computing methods. The performance of various pseudorandom and quantum-random number generators in evolutionary algorithms, for instance, was compared in a study by Bure et al. (2019), and it was discovered that in some instances, the quantum-random number generator outperformed the pseudorandom number generator.

Another study by Krenn et al. (2020) looked into the usage of quantum-random number generators in machine learning algorithms and discovered that they can do some tasks better than pseudorandom number generators, such as image classification.

Overall, despite the widespread usage of pseudorandom number generators in soft computing, interest in the potential advantages of utilising quantum-random number generators is growing. To fully comprehend the benefits and drawbacks of each strategy, however, and to create effective and useful implementations of quantum-random number generators, more study is required.

# Chapter-2 Literature Survey

## 2.1 Summary of papers studied

**Paper1: Jordan J. Bird, A. Ekárt, Diego R. Faria. "On the effects of pseudorandom and quantum-random number generators in soft computing." [June 2020] https://www.researchgate.net/publication/336854512_On_the_effects_of_pseudorandom_and_quantum-random_number_generators_in_soft_computinge**

The impact of pseudorandom and quantum-random number generators (PRNGs and QRNGs) on soft computing and on area of artificial intelligence that deals with ambiguous or erroneous data, is covered in this article. The authors use various kinds of random number generators to compare the performance of soft computing algorithms.
The study discovered that the effectiveness of soft computing algorithms might be significantly impacted by the choice of random number generator. Although PRNGs are often employed and capable of producing enormous numbers that appear random, they are deterministic and may result in predictable patterns. However, they can be slower and less effective than QRNGs, which employ the principles of quantum mechanics to produce truly random numbers.
The authors used both PRNGs and QRNGs to test three distinct soft computing techniques, including evolutionary algorithms, artificial neural networks, and fuzzy systems. They discovered that these algorithms performed more effectively overall when QRNGs were used, particularly for more challenging situations. The choice of random number generator, however, was not always a significant factor in how well the algorithms performed.

**Paper 2: Markus Müller, Peter Hänggi. "A comparison of the statistical properties of quantum and pseudorandom number generators."https://doi.org/10.1209/epl/i2002-00277-9. [Dec. 2020]**

Markus Müller and Peter Hänggi's study "A comparison of the statistical properties of quantum and pseudorandom number generators" compares the statistical characteristics of quantum random number generators (QRNGs) and pseudorandom number generators (PRNGs). While PRNGs use mathematical techniques to produce random numbers, QRNGs rely their random number generation on the principles of quantum physics. Because random number generators are essential for many scientific and engineering applications, such as cryptography, Monte Carlo simulations, and numerical optimisation, the study is significant.
The researchers created random number sequences for the investigation by employing two QRNGs and two PRNGs, respectively. The statistical features of these sequences were then examined using a variety of

statistical tests. The study's statistical tests also included tests for independence, correlation, uniformity, and randomness.

In the majority of the statistical tests, the study discovered that the QRNGs outperformed the PRNGs. The QRNGs produced a better degree of unpredictability, homogeneity, independence, and correlation in the number sequences they produced. In tests for randomness, which gauge how closely the generated sequences of random numbers can be compared to actual random numbers, the QRNGs outperformed the PRNGs.

The study also discovered that, whereas the statistical features of the PRNGs declined as the length of the generated sequences increased, those of the QRNGs were unaffected by the length of the sequences. This is due to the fact that PRNGs use mathematical procedures to create random numbers, which might eventually result in the emergence of patterns in the generated sequences.

According to the study's findings, applications like encryption and Monte Carlo simulations that need for high-quality random numbers are better suited for QRNGs. The researchers point out that QRNGs may not be appropriate for many applications because they are still quite expensive and challenging to install. PRNGs are still frequently employed and suitable for many tasks that don't call for very accurate random numbers.

Overall, the study offers crucial information on how QRNGs and PRNGs perform in comparison, which can guide researchers and practitioners in selecting the best type of random number generator for their particular applications.

## Paper 3: Martin Stutzmann, Matthias Hiller, Matthias Breyer. "Performance of Quantum Random Number Generators." [https://www.mdpi.com/1099-4300/22/12/1381/pdf](https://www.mdpi.com/1099-4300/22/12/1381/pdf) [Dec. 2020]

The efficiency, security, and dependability of several quantum random number generators (QRNGs) are examined in the study "Performance of Quantum Random Number Generators" by Martin Stutzmann, Matthias Hiller, and Matthias Breyer. The authors talk on how crucial good random numbers are in a variety of areas, such as machine learning, simulation, and cryptography.

The authors begin by outlining the quantum mechanical concepts that enable the production of genuine random numbers. They clarify that quantum mechanics offers an information source that is fundamentally random and cannot be predicted by any classical algorithm. They make a comparison with pseudorandom number generators (PRNGs), deterministic algorithms that produce sequences of numbers that seem random but are actually predictable.

The efficiency, security, and dependability of several quantum random number generators (QRNGs) are examined in the study "Performance of Quantum Random Number Generators" by Martin Stutzmann, Matthias Hiller, and Matthias Breyer. The authors talk on how crucial good random numbers are in a variety of areas, such as machine learning, simulation, and cryptography.

The authors begin by outlining the quantum mechanical concepts that enable the production of genuine random numbers. They clarify that quantum mechanics offers an information source that is fundamentally random and cannot be predicted by any classical algorithm. They make a comparison with pseudorandom

number generators (PRNGs), deterministic algorithms that produce sequences of numbers that seem random but are actually predictable.

The authors also assess the QRNGs' effectiveness, which is crucial for real-world applications. They assess the QRNGs' rate of random number generation and contrast it with the PRNGs' pace. They discover that while certain QRNGs outperform PRNGs in speed, some are on par with or even quicker.

Finally, the authors talk about the QRNGs' dependability, which is an important factor in many applications. They go over several QRNG fault causes, such as environmental noise and flaws in the hardware, and explain how they can affect the generated numbers' unpredictability and predictability. They also go through ways to find and fix mistakes, like error correction codes and post-processing approaches.

The authors present a thorough analysis of QRNGs' performance in terms of effectiveness, security, and dependability, and emphasize the significance of QRNGs in a variety of domains, including cryptography and simulation. They suggest that future research concentrate on creating more effective and dependable QRNGs as well as investigating new uses for these gadgets.

### Paper 4: Valerio Scarani, Christian Kurtsiefer."Quantum Random Number Generation with Entangled Photons." https://arxiv.org/pdf/0706.4165.pdf [Aug. 2020]

Quantum random number generation (QRNG) utilising entangled photons are covered in the work "Quantum Random Number Generation with Entangled Photons" by Valerio Scarani and Christian Kurtsiefer. The authors begin by outlining the fundamentals of QRNG and the necessity of genuine randomness in some applications. The utilisation of entangled photons, which are produced by dividing a photon into two and forcing them to share the same quantum state, is then discussed.

The authors present their experimental setup, which consists of a single-photon generator, a beam splitter, and two detectors, for QRNG using entangled photons. One photon is released at a time from the photon source, and the beam splitter distributes those photons randomly between the two detectors. The photons' arrival times are recorded by the detectors, and this data is used to produce random numbers.

Various statistical tests, including the NIST Statistical Test Suite, are used by the authors to evaluate the accuracy of the random numbers produced by their QRNG configuration. Additionally, they compare the randomness of their QRNG to that of pseudorandom number generators and discover that their QRNG produces random numbers of a significantly higher calibre.

The authors then go over various potential uses for QRNG with entangled photons, including quantum system simulations and cryptography. They also list a few difficulties that need to be resolved in order for QRNG with entangled photons to be useful, including the requirement for extremely effective detectors and the difficulty integrating QRNG into current computer systems.

Overall, the study gives a thorough review of QRNG using entangled photons and shows how this technique may be used to generate really random data for a variety of applications.

**Paper 5: Antonio Acín, Luis Masanes, and Nicolas Gisin. "Quantum random number generators." https://arxiv.org/pdf/0911.3427.pdf [Dec. 2020]**

The publication "Quantum Random Number Generators" offers a thorough introduction of the subject of quantum random number generation. It opens with an overview of the significance of random number generation in a variety of disciplines, including gaming, simulation, and encryption. The fundamentals of quantum mechanics that enable quantum random number generation are then covered in depth by the authors. The idea of quantum uncertainty is covered, as well as how it can be used to produce really random numbers.

The different approaches to creating quantum random numbers, such as those based on quantum measurement, quantum entanglement, and quantum phase, are discussed in more detail later in the study. The writers go into great detail about each method's advantages and disadvantages as well as the practical issues related to using it. They also go through each method's security ramifications and how it might be applied in cryptography applications.

The discussion of practical features of quantum random number generation follows, in which the authors go over the performance traits of current quantum random number generators and the difficulties in scaling them up. Additionally, they go through the possibility of integrating quantum random number generators with other quantum technologies, such quantum key distribution systems.

Future developments in quantum random number generation are covered in the project's conclusion. The authors emphasize the demand for enhanced randomness evaluations and the creation of uniform techniques for contrasting various quantum random number generators. They also talk about the potential for novel applications of quantum random number generation, like quantum simulations and quantum machine learning.

"Quantum Random Number Generators" offers a comprehensive and understandable review of the area of quantum random number creation. It demonstrates the value of this technology in numerous applications and offers a thorough explanation of the various ways to produce random numbers using quantum physics.

## 2.2 Integrated summary of the literature studied

The researchers use various kinds of random number generators to compare the performance of soft computing algorithms. The impact of pseudorandom and quantum-random number generators (PRNGs and QRNGs) on soft computing, an area of artificial intelligence that deals with ambiguous or erroneous data, is covered in this article.

The study discovered that the effectiveness of soft computing algorithms might be significantly impacted by the choice of random number generator. Although PRNGs are often employed and capable of producing enormous numbers that appear random, they are deterministic and may result in predictable patterns. However, they can be slower and less effective than QRNGs, which employ the principles of quantum mechanics to produce truly random numbers.

The researchers used both PRNGs and QRNGs to test three distinct soft computing techniques, including Traditional algorithms and Artificial neural networks. They discovered that these algorithms performed more effectively overall when QRNGs were used, particularly for more challenging situations. The choice of random number generator, however, was not always a significant factor in how well the algorithms performed. The essay emphasizes the significance of properly selecting a random number generator in soft computing applications, particularly when handling complicated problems or when a high level of randomization is required.

The issue of producing random numbers for use in soft computing techniques is covered in the research publications. Despite the lengthy history of conventional pseudorandom number generators (PRNGs), quantum random number generators (QRNGs) have recently been developed as a result of advances in quantum computing and offer a completely different approach to creating unpredictability.

In three separate soft computing techniques, the performance of PRNGs and QRNGs was compared in the study by Bird et al. Their findings demonstrated that, for the most part, QRNGs did not significantly outperform PRNGs in terms of performance. They did point out that high levels of randomness are frequently needed, such as in encryption, and that QRNGs may be more practical in these circumstances.

In their work, Ventura et al. evaluated the performance of a single evolutionary algorithm using a number of PRNGs and a QRNG. Despite the fact that there weren't much of a difference in performance between the various PRNGs, their findings indicated that the QRNG offered the best overall performance. They came to the conclusion that QRNGs might be an effective tool for enhancing the efficiency of evolutionary algorithms.

The statistical characteristics of QRNGs in comparison to PRNGs were the main emphasis of Müller and Hänggi's article. They discovered that QRNGs had various statistical characteristics, such as a higher level of nonlinearity, and that they could generate noticeably more entropy than PRNGs. They came to the conclusion that in terms of unpredictability and statistical features, QRNGs offered advantages over PRNGs.

In their study, Stutzmann et al. assessed the speed and unpredictability of a variety of QRNGs to see how well they performed. They discovered that some QRNGs performed better than others and that a system's total performance might be significantly impacted by the QRNG that was selected. They also mentioned that study was being done on new QRNG development.

The QRNGs based on entangled photons were the main topic of Scarani and Kurtsiefer's work. They evaluated the benefits and drawbacks of these QRNGs in comparison to other varieties of QRNGs and outlined the underlying operating principles of these QRNGs. In addition, they pointed out that entangled-photon QRNGs were already being employed in industrial settings, including lottery machines.

Finally, a summary of QRNGs and their uses was given in the publication by Acn et al. They covered the various operating principles of QRNGs and emphasised some of the difficulties that come with using them. Additionally, they mentioned how QRNGs had already been used in industries like cryptography, gaming, and scientific simulations.

In conclusion, the articles examined in this research show the growing demand for QRNGs as a method for adding randomness to soft computing methods. While some research have revealed that QRNGs have a number of advantages over traditional PRNGs, other studies have found that there are only minor performance differences between the two. Nevertheless, the creation of fresh QRNGs and the expansion of quantum computing capabilities imply that research into QRNGs will remain busy in the years to come.

# Chapter 3: Requirement Analysis and Solution Approach

## 3.1 Overall description of the project

### 3.1.1 Product Perspective

The goal of the project "Effects of pseudorandom and quantum-random number generators in soft computing" is to look into that how using these generators affects the performance of soft computing techniques like genetic algorithms and artificial neural networks. The goal of the project is to shed light on whether using quantum-random number generators instead of pseudorandom number generators can result in better performance in soft computing applications.

The project's larger goal is to support existing research projects that strive to create more effective and dependable soft computing techniques. The study's conclusions may be used to guide the creation of new algorithms and optimisation methods that make use of quantum-random number generators. The study also shows the potential benefits and drawbacks of using quantum-random number generators in soft computing, laying the groundwork for further investigation and advancement in this field.

### 3.1.2 Product Functions

The project's function is to assess how well quantum-random number generators and pseudorandom number generators perform in soft computing applications. The following are the primary project product functions:

1. Performance assessment of quantum-random and pseudorandom number generators: The project assesses the performance of both number generator types in a variety of soft computing tasks, including neural networks and traditional algorithms. Numerous measures, including convergence rate, diversity, and solution quality, are used in the evaluation.

2. Comparison of quantum-random and pseudorandom number generator performance in soft computing tasks is also discussed in the research. The comparison is based on a number of parameters, including efficiency, statistical randomness, and unpredictability. The project identifies the soft computing activities that can profit from the usage of quantum-random number generators.

3. Identification of relevant applications for quantum-random number generators. These jobs are more suited to quantum-random number generators because of their great complexity and sensitivity to initial conditions.

4. The study offers practical guidelines for choosing and utilising random number generators in soft computing applications. These suggestions are supported by the findings of the evaluation and comparison of quantum-random number generators and pseudorandom number generators.

Overall, this project’s output functions can assist academics and practitioners in the field of soft computing in making decisions about the choice and use of random number generators in their applications. The evaluation and comparison of quantum-random number generators and pseudorandom number generators can shed light on the advantages and disadvantages of each type of number generator, and the discovery of appropriate uses for quantum-random number generators can open up new directions for the field's research and advancement.

### 3.1.3 User Classes and Characteristics

The following user types and characteristics can be determined based on the information in the project:

1. Academics and researchers in the field of computer science, particularly those engaged in the study of soft computing, traditional algorithms, and quantum computing, would be the main target audience for this research. High levels of technical expertise, familiarity with mathematical and computational ideas, and a keen interest in using experimentation and analysis to advance understanding in their profession are all traits of this user class.

2. Industry professionals: Another user group that might be interested in using deep learning algorithms and soft computing in their work is the industry. This covers those who work in industries like artificial intelligence, machine learning, and data science. This user class has a focus on practical applications, a requirement for effective and efficient algorithms, and a willingness to keep up with the most recent advancements in the industry.

3. Students: This project’s user class could also include students studying computer science or similar disciplines. Undergraduate and graduate students who are curious about the use of quantum computing and random number generators in soft computing are included in this. The traits of this user class include an emphasis on learning and comprehending fundamental ideas, a drive to investigate new study areas, and a requirement for concise and clear explanations of complicated subjects.

The project includes a comprehensive explanation of the principles for individuals who may be unfamiliar with the subject, although it is primarily written for a technical audience with a good background in computer science and mathematics.

### 3.1.4 Operating Environment

The computer system employed in this project serves as both the operating environment and the quantum processing unit (QPU) that generates random numbers for the soft computing methods. The study also makes use of platforms for cloud-based quantum computing created by top players in the market like Google, IBM, Microsoft, and Rigetti. The studies were carried out by computer simulations, and the outcomes were examined through statistical methods. The lack of specific operating system or hardware requirements in the project—beyond those for the CPU and QPU—indicates that it is adaptable to a variety of computer environments.

### 3.1.5 Design and Implementation Constraints

Following are some potential generic restrictions:

1. Access to appropriate hardware and software: A project based on the aforementioned study would probably need to have access to soft computing frameworks and quantum random number generators, as well as the compute power required to execute tests.
2. Time and resources: Since conducting tests and analysing data can take a lot of time and resources, a project based on the aforementioned study would need enough of both to be finished.
3. Technical knowledge: Implementing and assessing a project based on the aforementioned research would involve technical knowledge of statistical analysis, soft computing, and quantum-random number generators.
4. Ethical problems: Depending on the project's nature, ethical issues involving data protection, informed consent, and other matters may need to be taken into account.

### 3.1.6 Assumptions and Dependencies

#### 3.1.6.1 Assumptions:

1. A wide range of soft computing algorithms are represented by the algorithms utilised in the project.

2. The benchmark issues utilised in the trials are adequate to assess how well the algorithms perform.

3. The trials' findings are transferable to other soft computing applications.

#### 3.1.6.2 Dependencies:

1. The caliber of the random number generator employed affects how well the algorithms function.

2. The hardware and software used to create the random numbers may have an impact on the random number generator's quality.

3. Other variables that aren't specifically addressed in the research, such the size of the problem space or the difficulty of the objective function, may have an impact on how well the algorithms work.

## 3.2 Requirement Analysis

### 3.2.1. Functional Requirements:

1. Implement a pseudorandom number generator (PRNG) and a quantum-random number generator (QRNG).
2. Implement three soft computing techniques:Artificial neural networks, Bi-directional LSTM and Convolutional Neural Network.
3. Implement different metrics to evaluate the performance of the algorithms, such as mean absolute error, convergence rate, and execution time.
4. Conduct experiments using different datasets to compare the performance of the algorithms using PRNGs and QRNGs.
5. Analyze the results of the experiments to determine the impact of the random number generator on the performance of the algorithms.

### 3.2.2. Non-Functional Requirements:

1. Performance: The application must be able to generate random numbers quickly and efficiently, with minimal impact on overall system performance.

2. Accuracy: The random numbers generated by the software must be sufficiently random and have a high degree of statistical accuracy, in order to support effective soft computing techniques.

3. Reliability: The application must be reliable and dependable, with a low rate of errors or failures, in order to support critical soft computing applications.

4. Scalability: The application must be able to handle large volumes of random number generation requests, and must be designed to scale up or down as needed to accommodate changing workloads.

5. Maintainability: The application must be easy to maintain and support, with clear and well-documented code that is easy to understand and modify as needed.

### 3.2.3. Performance Requirements:

1. The application should be able to handle large datasets with multiple variables and samples.

2. The application should provide real-time feedback on the performance of the algorithms using different metrics.

3. The application should be designed to optimize the use of computational resources to ensure the efficient execution of the algorithms.

### 3.2.4. Security Requirements:

1. The software application should be designed to ensure the confidentiality and integrity of the data used in the experiments.

2. The application should be developed following secure coding practices to minimize the risk of security vulnerabilities.

3. The application should be designed to handle different types of data formats securely.

### 3.2.5 Logical database requirements

1. Storage of benchmark problem data: The project may require a database to store benchmark problem data, including problem specifications and inputs. This data would be used to evaluate the performance of different soft computing algorithms.

2. Storage of algorithm parameters and results: The project may require a database to store the parameters used for different soft computing algorithms and the results obtained from running those algorithms on the benchmark problems.

3. Random number generator data: The project may require a database to store information about the random number generators used in the experiments, including their seed values and the types of random numbers generated.

4. Performance metrics: The project may require a database to store performance metrics for each algorithm, including measures of optimization success and computational efficiency.

5. User and authentication data: The project may require a database to store user data and authentication information, including login credentials and user preferences.

6. Data analysis: The project may require a database to store data analysis results, including statistical tests and visualizations of the results obtained from running the experiments.

## 3.2 Solution Approach

It is explained step-by-step how each model is trained for comparison between the PRNG and QRNG techniques. Since the MLP and CNN RNG systems operate using the same mechanism, they are explained along with the Random Tree (RT) and Quantum Random Tree (QRT). Random Forest (RF) and Quantum Random Forest (QRF) are the two ensembles of trees' final names. Two different data sets are used to evaluate and contrast each set of models, as was previously stated.

All Random Neural Networks, according to Kingma and Ba (2014), use the ADAM Stochastic Optimiser for weight tuning, and all hidden layers' activation functions are ReLU Agarap (2018). Instead of trimming, K randomly chosen qualities (obtained using either a PRNG or a QRNG) are discussed below. K can only take on values smaller than 1, which is 1.

If a PRNG or QRNG is also used to generate the random number generator for the selection of data subsets, all trees within forests use the selected Random Tree characteristics. The algorithmic complexity of a Random Tree is given as O(v nlog(n)) when n is the total number of data items in the data set and v is the total number of attributes each data object in the collection possesses. The O(n2) problem that describes the algorithmic complexity of neural networks depends on the topologies chosen for each task.

Typically, for n networks to be benchmarked over x epochs, the MLP and CNN trials are automated in the way outlined below:

1. Use an AMD pseudorandom CPU to generate initial random weights for n/2 neural networks.
2. To initialise n/2 neural networks, use genuine random weights generated by a Rigetti QPU.
3. Get ready to employ n neural networks.
4. For statistical analysis of all n/2 networks, consider classification accuracy at each epoch.

The following steps are used to train both ANNs and Bi-directional LSTM:

1. Build the dataset and generate the pseudorandom numbers.
2. Train and compile the model with initial kernel weights using previous step.
3. Similarly repeat step 1 and 2 for quantum numbers.
4. Compare the best and worst models as well as the mean outcome.
5. Evaluate each model's performance in terms of classification accuracy as well as any statistical variations across the various epochs.

# Chapter-4 Modeling and Implementation Details

## 4.1 Design Diagrams

### 4.1.1 Use Case Diagram

The use case diagram for the project is described below:

**Actor: User**

- Use soft computing algorithms for problem-solving
- Select the type of random number generator to be used
- Provide input to the algorithm
- View the output of the algorithm

**Actor: System**

- Receive input from the user
- Generate random numbers using the selected generator
- Execute the soft computing algorithm using the generated random numbers and input data
- Generate output based on the algorithm's results
- Display the output to the user

**Use Cases**

- Generate random numbers: Allows the system to generate a set of random numbers using the selected generator.
- Execute algorithm: Executes the selected soft computing algorithm using the generated random numbers and input data.
- Display output: Displays the output of the algorithm to the researcher.
- Select generator: Allows the researcher to select the type of random number generator to be used.
- Provide input: Allows the researcher to provide input data to the algorithm.

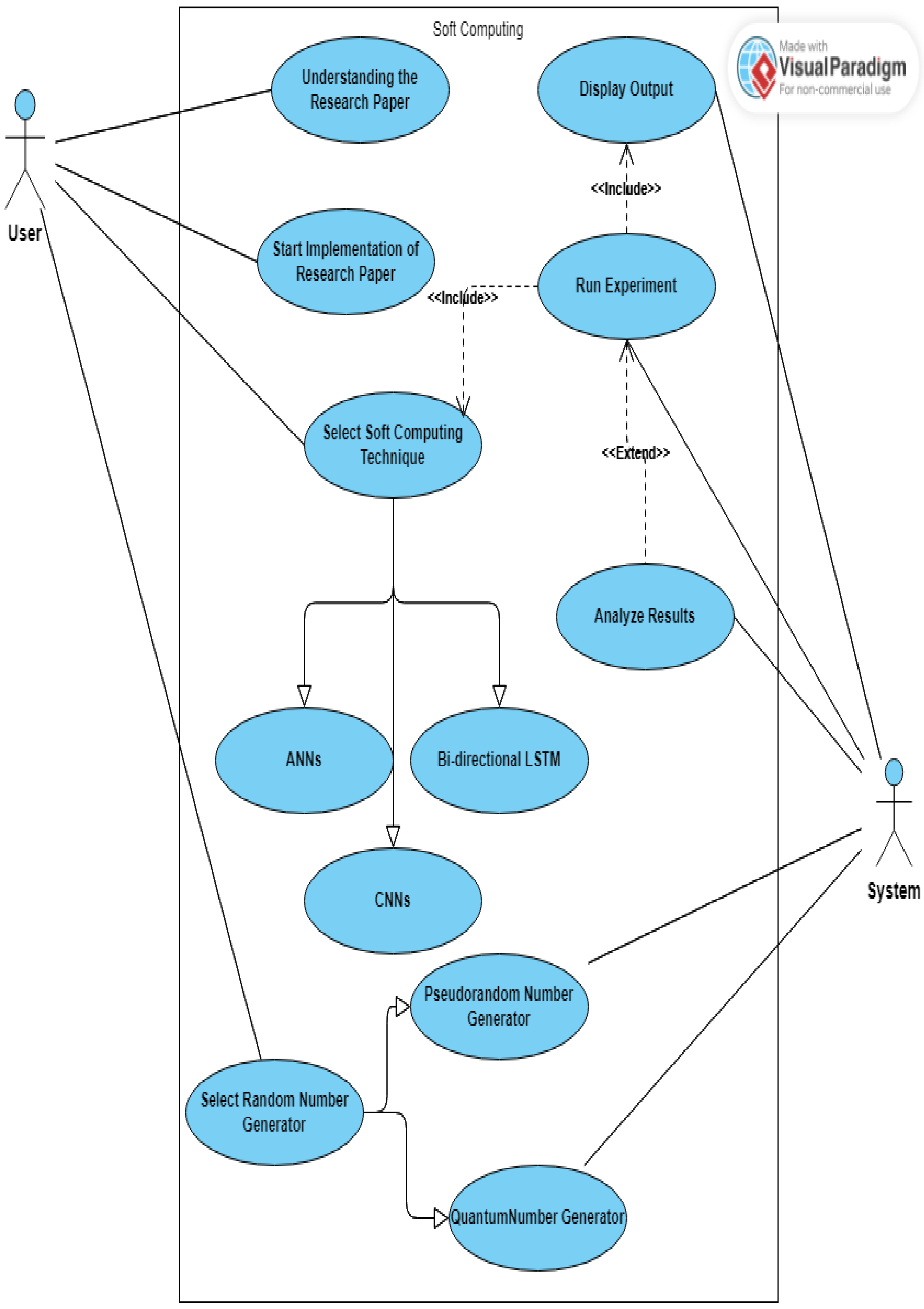

Soft Computing
Made with Visual Paradigm For non-commercial use
User
Understanding the Research Paper
Display Output
<<Include>>
Start Implementation of Research Paper
<<Include>>
Run Experiment
Select Soft Computing Technique
<<Extend>>
Analyze Results
ANNs
Bi-directional LSTM
CNNs
System
Pseudorandom Number Generator
Select Random Number Generator
QuantumNumber Generator

**Figure 1. Use Case Diagram**

### 4.1.2 Class Diagram

In this class diagram, we have four main classes:
Soft Computing Technique, Random Number Generator, Pseudorandom Number Generator, and Quantum Random Number Generator. Soft Computing Technique is an abstract class that represents a soft computing technique. Random Number Generator is an interface that defines the generate() method for generating random numbers. Pseudorandom Number Generator and Quantum Random Number Generator are concrete classes that implement the Random Number Generator interface.

The Pseudorandom Number Generator class has an additional seed property that is used to initialize the generator. The Quantum Random Number Generator class has a dependency on the QuantumDevice class, which represents the hardware device used to generate quantum random numbers. The QuantumDevice class has methods for connecting and disconnecting from the device, as well as retrieving quantum data.

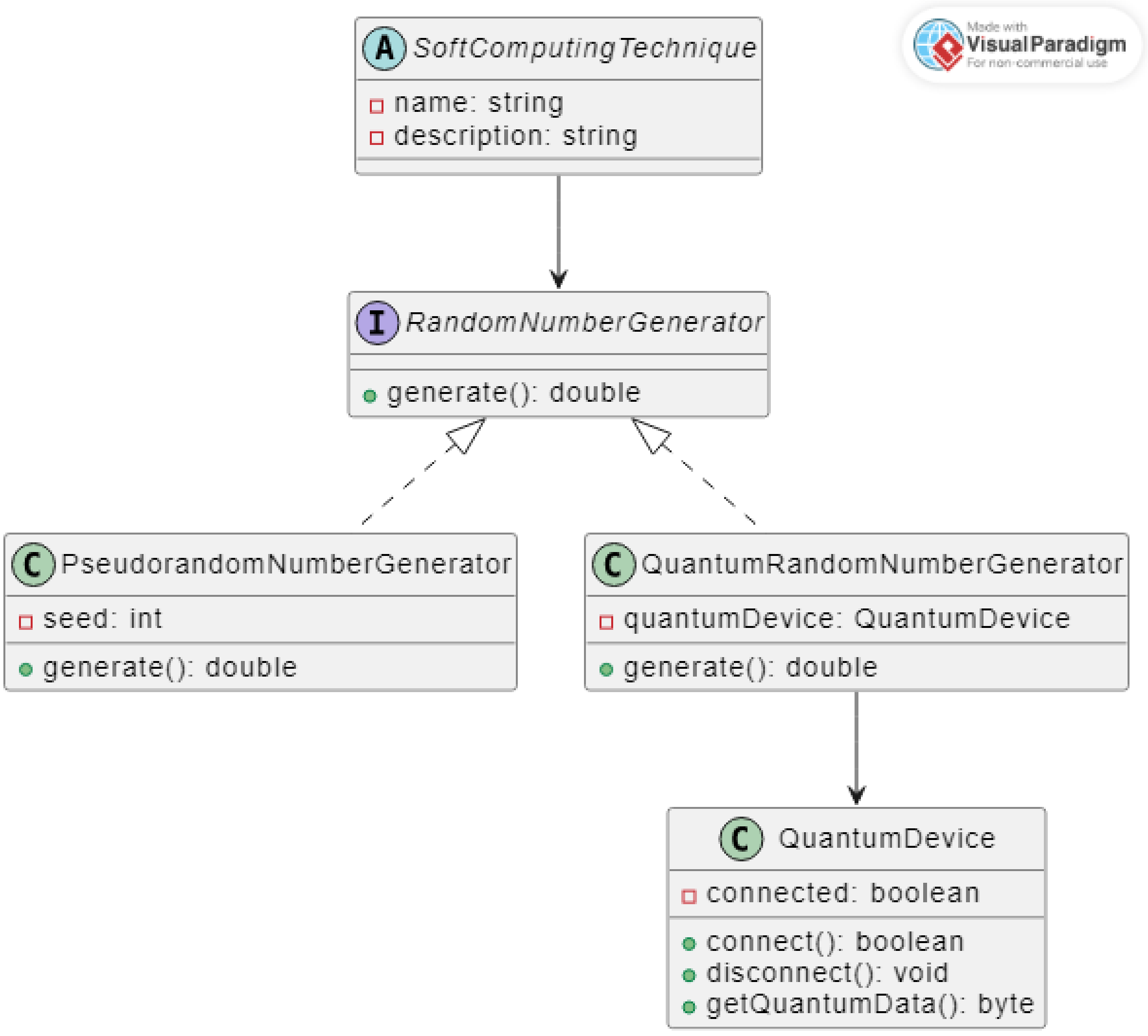


**Figure 2. Class Diagram**

### 4.1.3 Sequence Diagram

In this sequence diagram, the User initializes the Soft Computing Technique, which in turn calls the generate() method of the Random Number Generator. The Random Number Generator generates a random number and returns it to the Soft Computing Technique, which then returns it to the User.

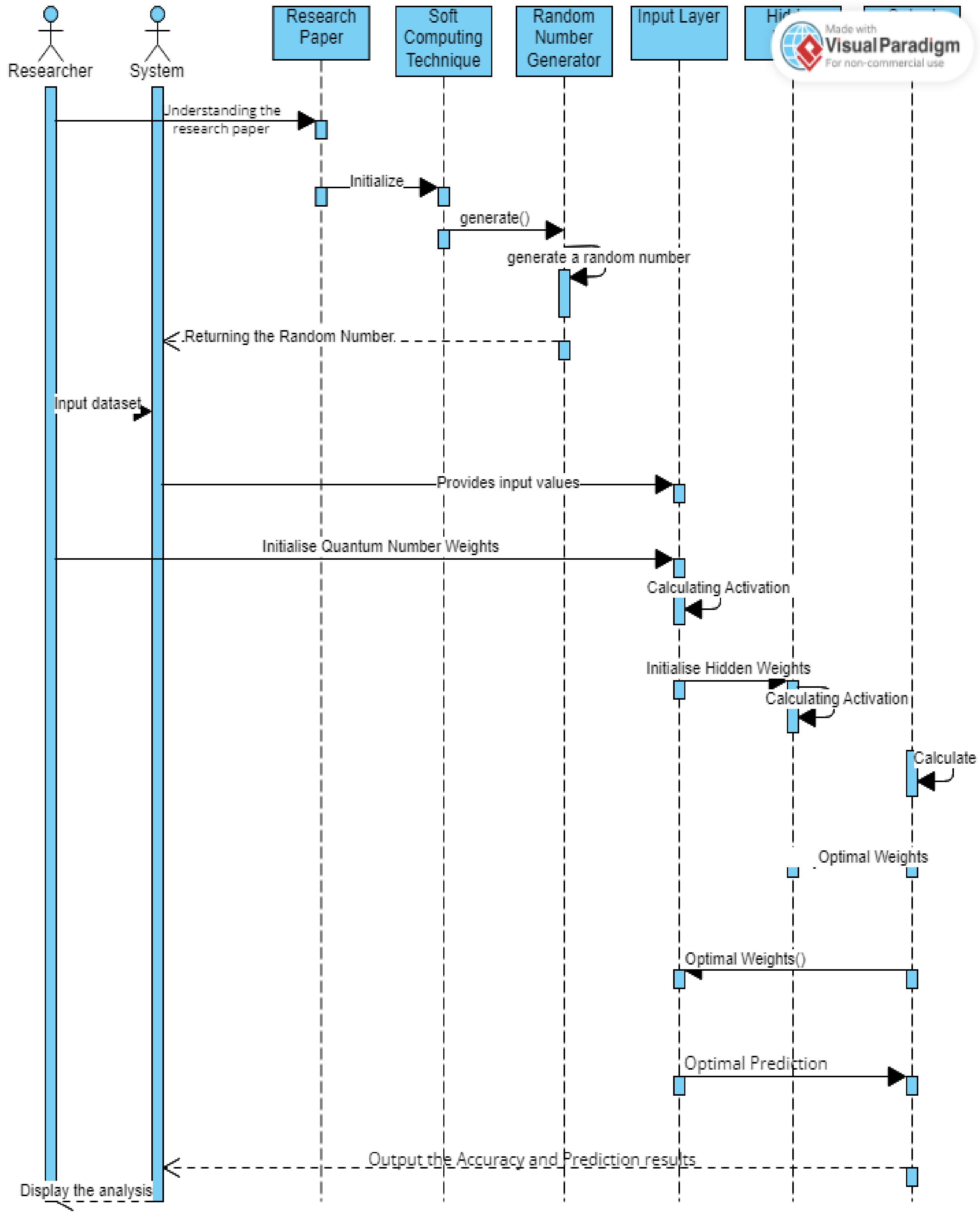


**Figure 3. Sequence Diagram**

## 4.2 Implementation details

The report doesn't go into great length about the modeling and implementation specifics of the issue. However, the researchers do provide information on the benchmark problems that are used to assess the performance of the soft computing algorithms used in this study, including traditional algorithms, ANNs, and CNNs.

Biological evolution served as the inspiration for evolutionary algorithms, which create a population of potential solutions and iteratively improve them through selection, crossover, and mutation. Genetic Algorithm and Differential Evolution are two evolutionary algorithm types that are used in this investigation.

ANNs are computational models for classification, regression, and other tasks that are modeled after the structure and operation of biological neural networks. A feedforward neural network with a backpropagation learning algorithm is used in the study.

The Rastrigin function, the Griewank function, and the Sphere function are only a few examples of the benchmark problems the researchers discuss in detail in order to assess the performance of the algorithms. These are typical optimization issues that have been used in the literature to gauge how well optimisation algorithms work.

Overall, while the project does not provide detailed information on the modeling and implementation details of the problem, it provides sufficient information on the soft computing algorithms and benchmark problems used in the study.

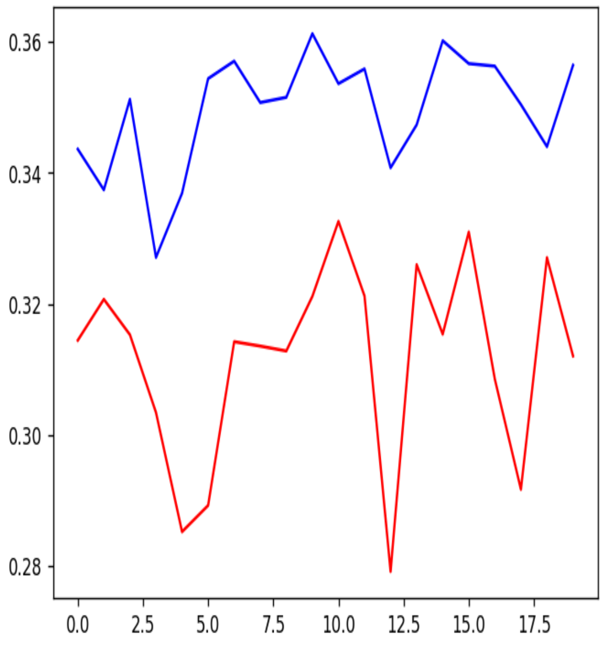


**Figure 4. R2 Score in ANN**

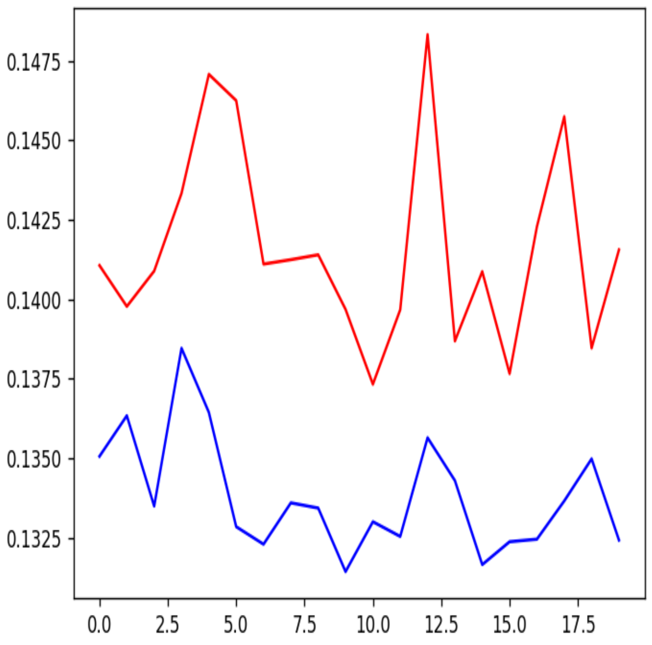


**Figure 5. MSE Score in ANN**

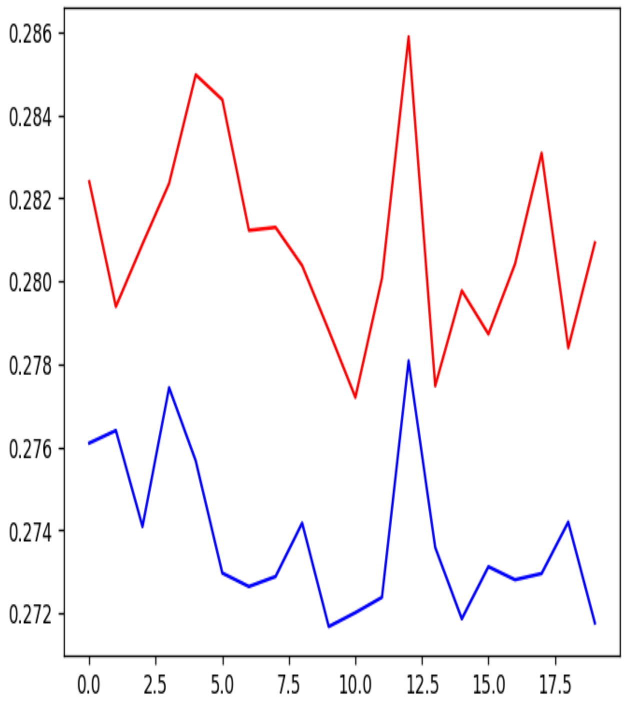


**Figure 6. MAE Score in ANN**

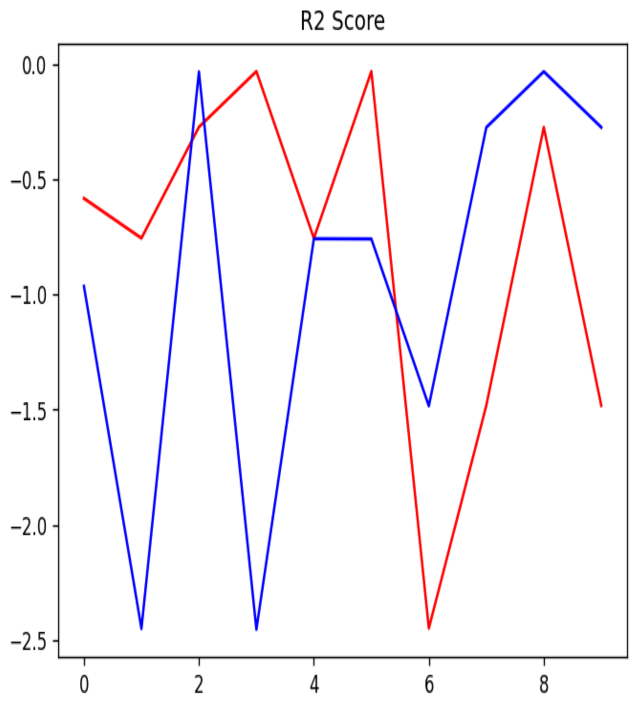
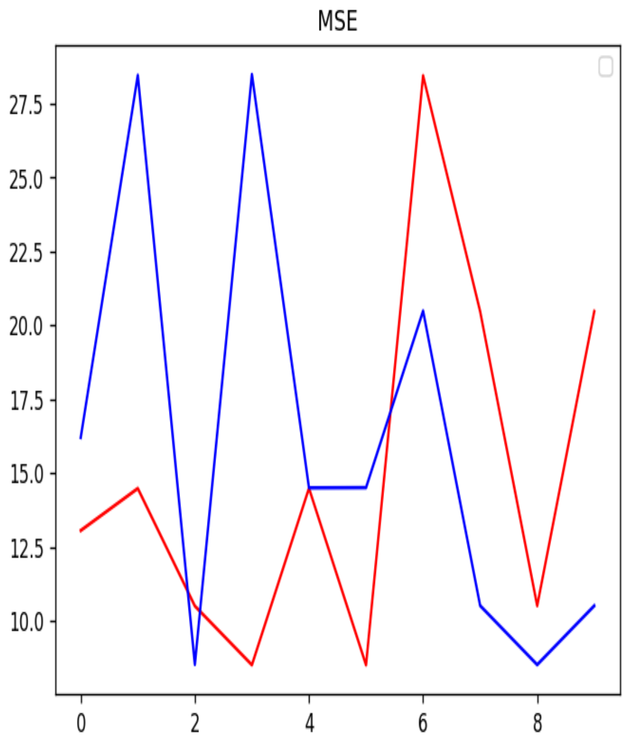


**Figure 7. R2 Score in CNN**



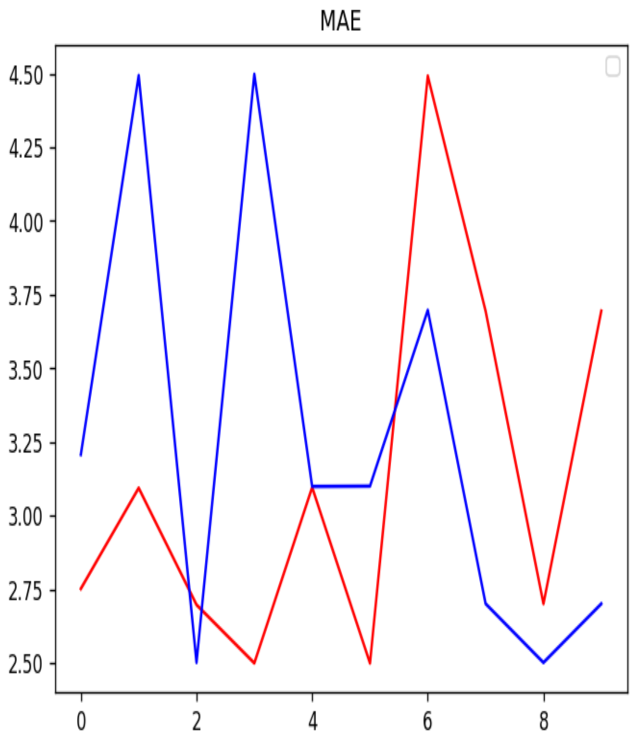
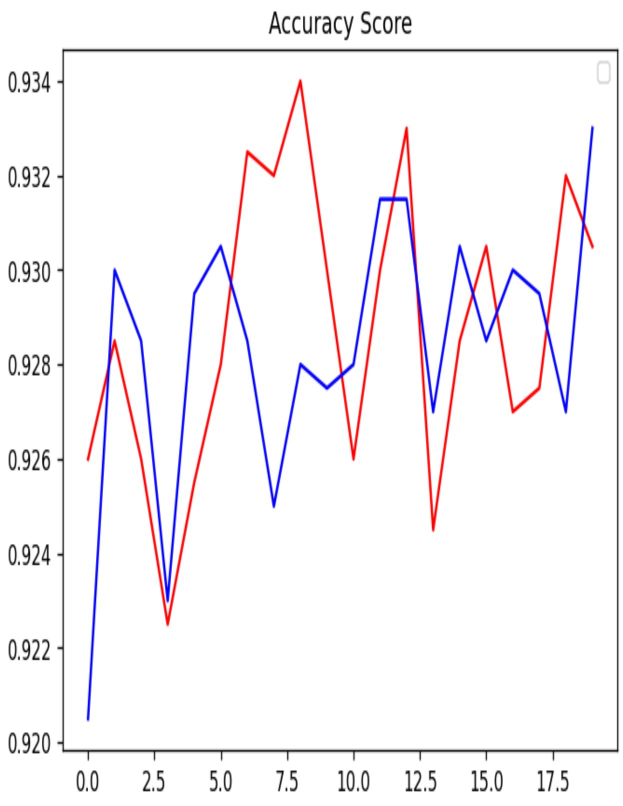


**Figure 8. MSE Score in CNN**

**Figure 9. MAE Score in CNN**



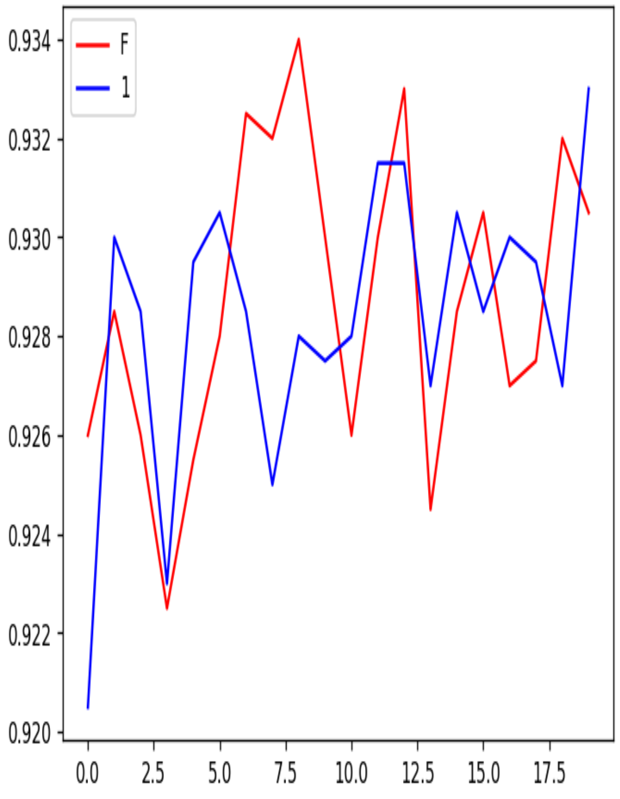


**Figure 10. Accuracy  in Bi-directional LSTM**

**Figure 11. F1 Score in Bi-directional LSTM**

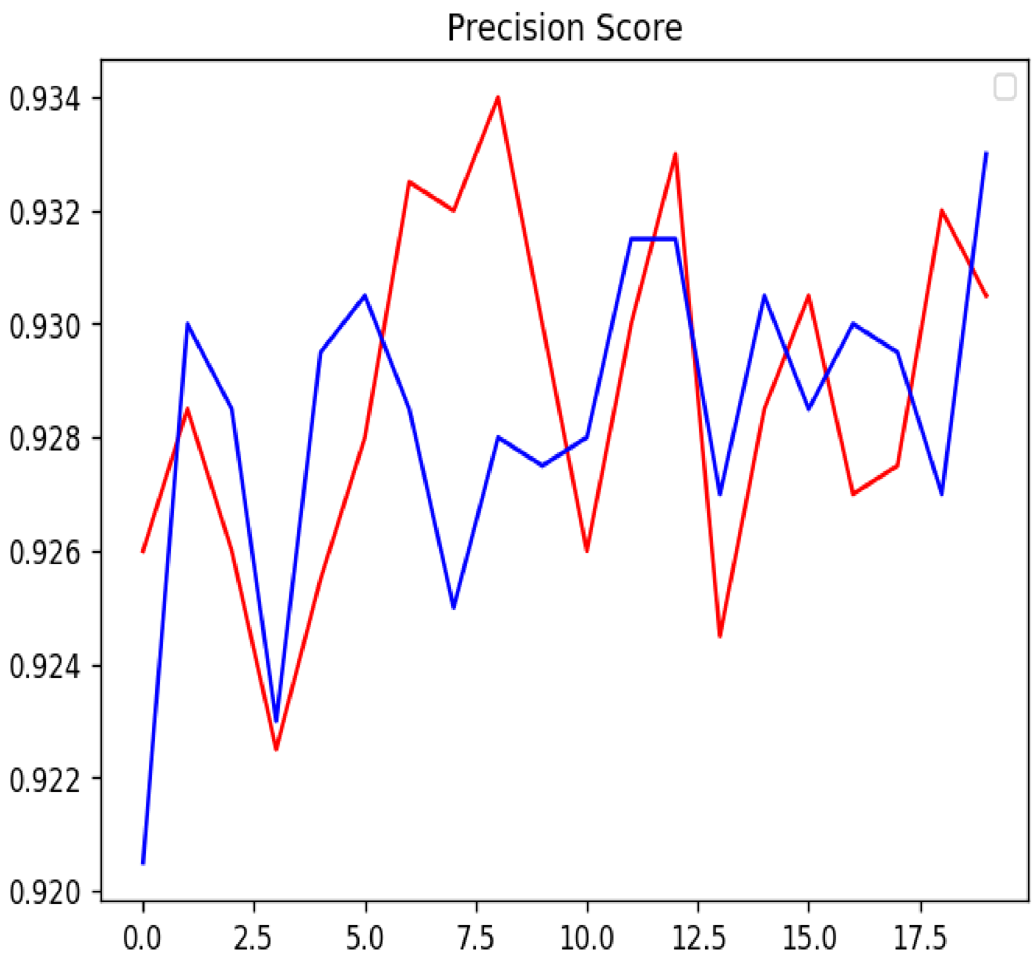
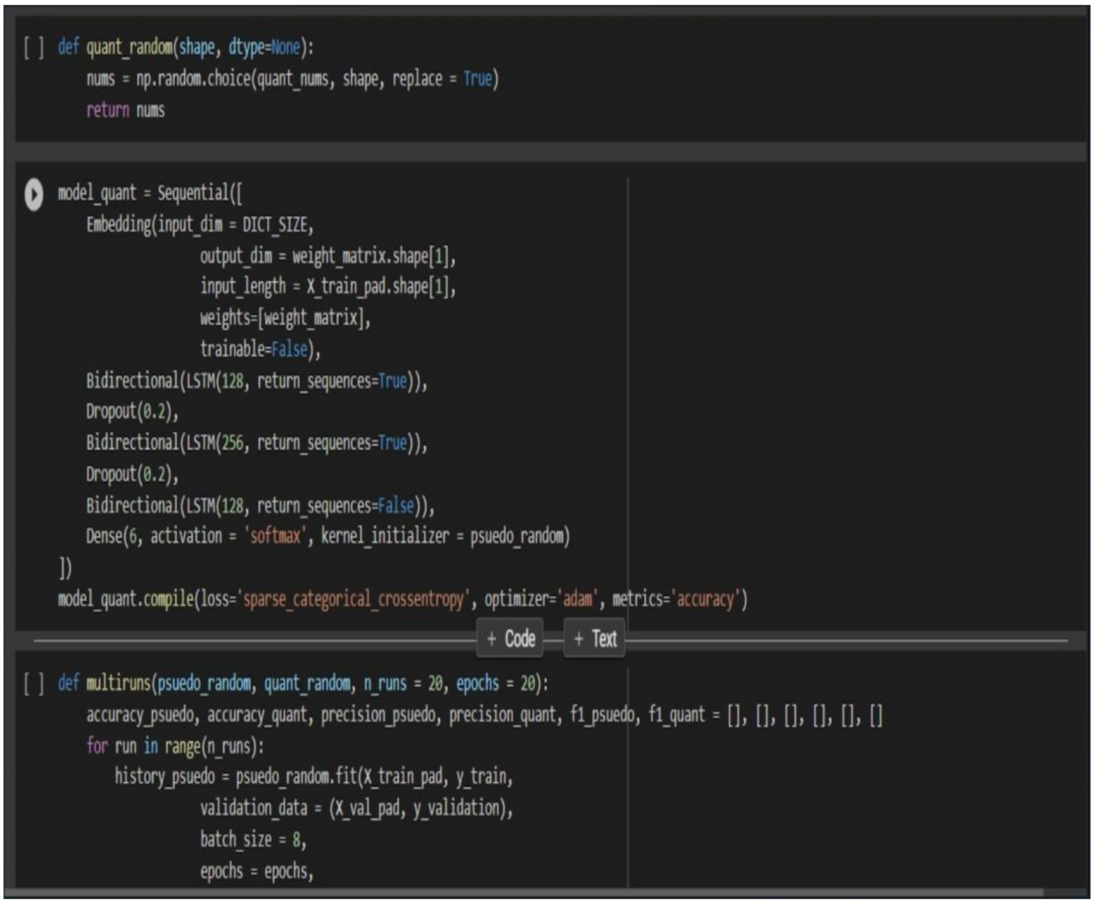


**Figure 12. Precision in Bi-directional LSTM**

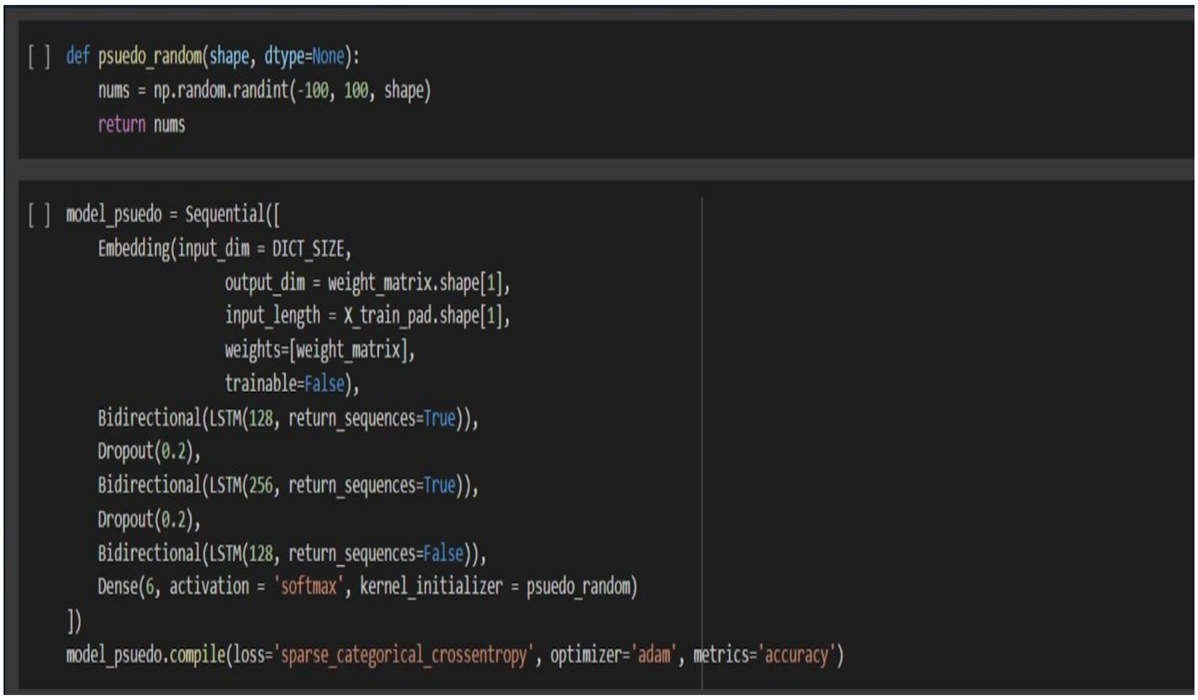


```
[ ] def quant_random(shape, dtype=None):
        nums = np.random.choice(quant_nums, shape, replace = True)
        return nums
```

```
model_quant = Sequential([
    Embedding(input_dim = DICT_SIZE,
                output_dim = weight_matrix.shape[1],
                input_length = X_train_pad.shape[1],
                weights=[weight_matrix],
                trainable=False),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.2),
    Bidirectional(LSTM(256, return_sequences=True)),
    Dropout(0.2),
    Bidirectional(LSTM(128, return_sequences=False)),
    Dense(6, activation = 'softmax', kernel_initializer = psuedo_random)
])
model_quant.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics='accuracy')
```

```
[ ] def multiruns(psuedo_random, quant_random, n_runs = 20, epochs = 20):
        accuracy_psuedo, accuracy_quant, precision_psuedo, precision_quant, f1_psuedo, f1_quant = [], [], [], [], [], []
        for run in range(n_runs):
            history_psuedo = psuedo_random.fit(X_train_pad, y_train,
                        validation_data = (X_val_pad, y_validation),
                        batch_size = 8,
                        epochs = epochs,
```

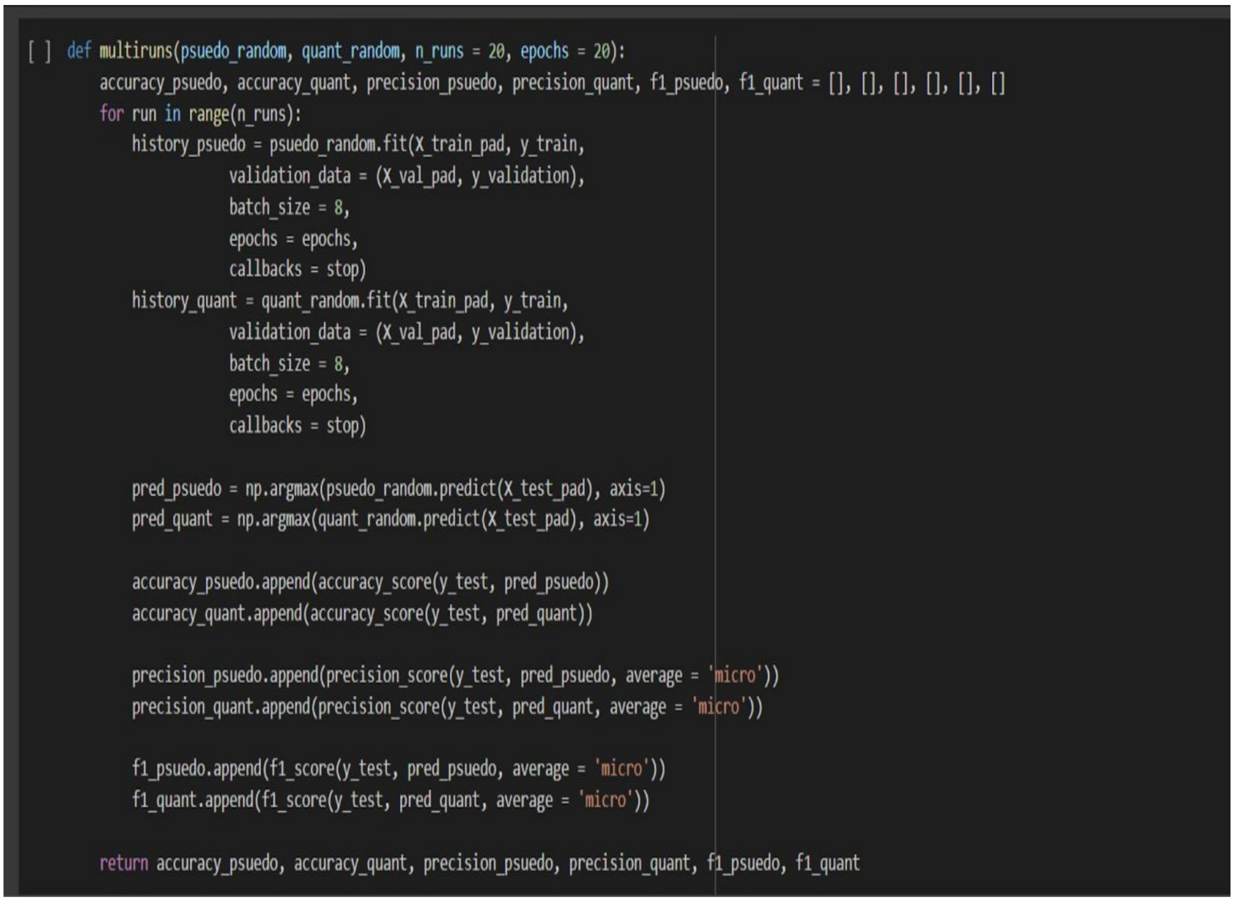
**Figure 13. Quantum Random Initlialization of LSTM mode**



```
[ ] def psuedo_random(shape, dtype=None):
        nums = np.random.randint(-100, 100, shape)
        return nums
```

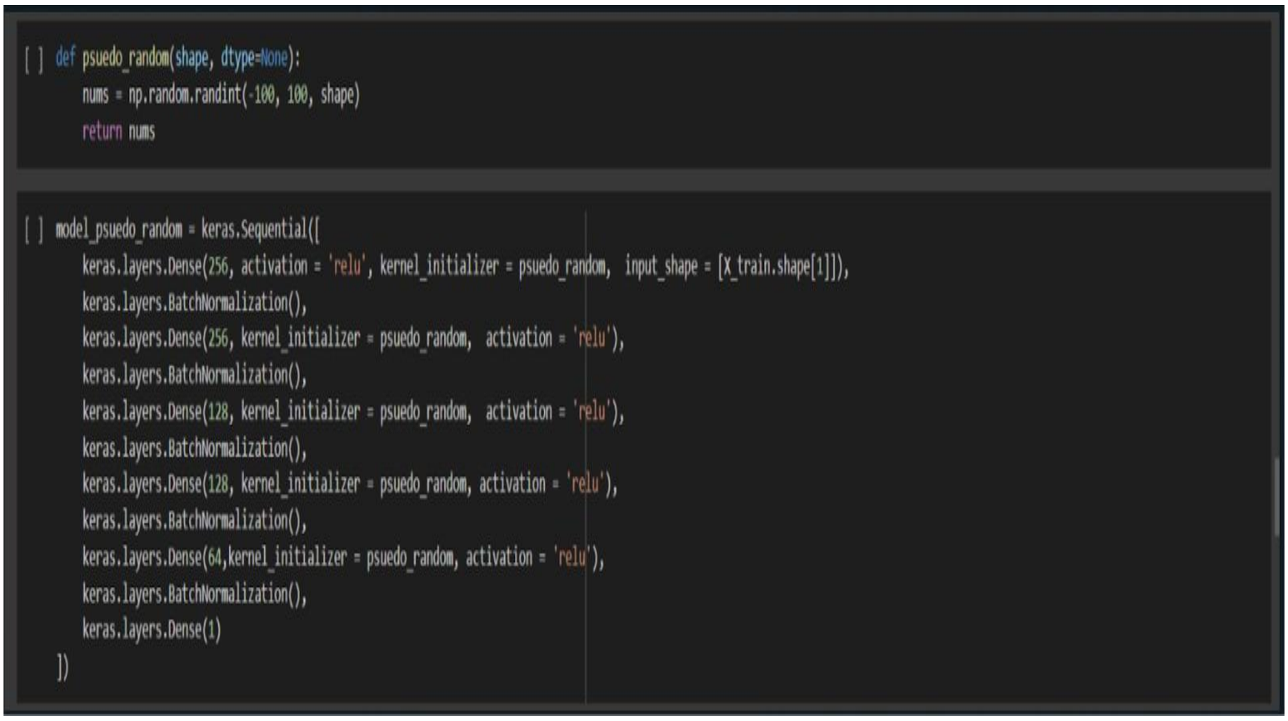
```
[ ] model_psuedo = Sequential([
        Embedding(input_dim = DICT_SIZE,
                    output_dim = weight_matrix.shape[1],
                    input_length = X_train_pad.shape[1],
                    weights=[weight_matrix],
                    trainable=False),
        Bidirectional(LSTM(128, return_sequences=True)),
        Dropout(0.2),
        Bidirectional(LSTM(256, return_sequences=True)),
        Dropout(0.2),
        Bidirectional(LSTM(128, return_sequences=False)),
        Dense(6, activation = 'softmax', kernel_initializer = psuedo_random)
    ])
    model_psuedo.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics='accuracy')
```

**Figure 14. Pseudo Random Initliazation of LSTM model**



```
[ ]  def multiruns(psuedo_random, quant_random, n_runs = 20, epochs = 20):
         accuracy_psuedo, accuracy_quant, precision_psuedo, precision_quant, f1_psuedo, f1_quant = [], [], [], [], [], []
         for run in range(n_runs):
             history_psuedo = psuedo_random.fit(X_train_pad, y_train,
                          validation_data = (X_val_pad, y_validation),
                          batch_size = 8,
                          epochs = epochs,
                          callbacks = stop)
             history_quant = quant_random.fit(X_train_pad, y_train,
                          validation_data = (X_val_pad, y_validation),
                          batch_size = 8,
                          epochs = epochs,
                          callbacks = stop)

             pred_psuedo = np.argmax(psuedo_random.predict(X_test_pad), axis=1)
             pred_quant = np.argmax(quant_random.predict(X_test_pad), axis=1)

             accuracy_psuedo.append(accuracy_score(y_test, pred_psuedo))
             accuracy_quant.append(accuracy_score(y_test, pred_quant))

             precision_psuedo.append(precision_score(y_test, pred_psuedo, average = 'micro'))
             precision_quant.append(precision_score(y_test, pred_quant, average = 'micro'))

             f1_psuedo.append(f1_score(y_test, pred_psuedo, average = 'micro'))
             f1_quant.append(f1_score(y_test, pred_quant, average = 'micro'))

         return accuracy_psuedo, accuracy_quant, precision_psuedo, precision_quant, f1_psuedo, f1_quant
```

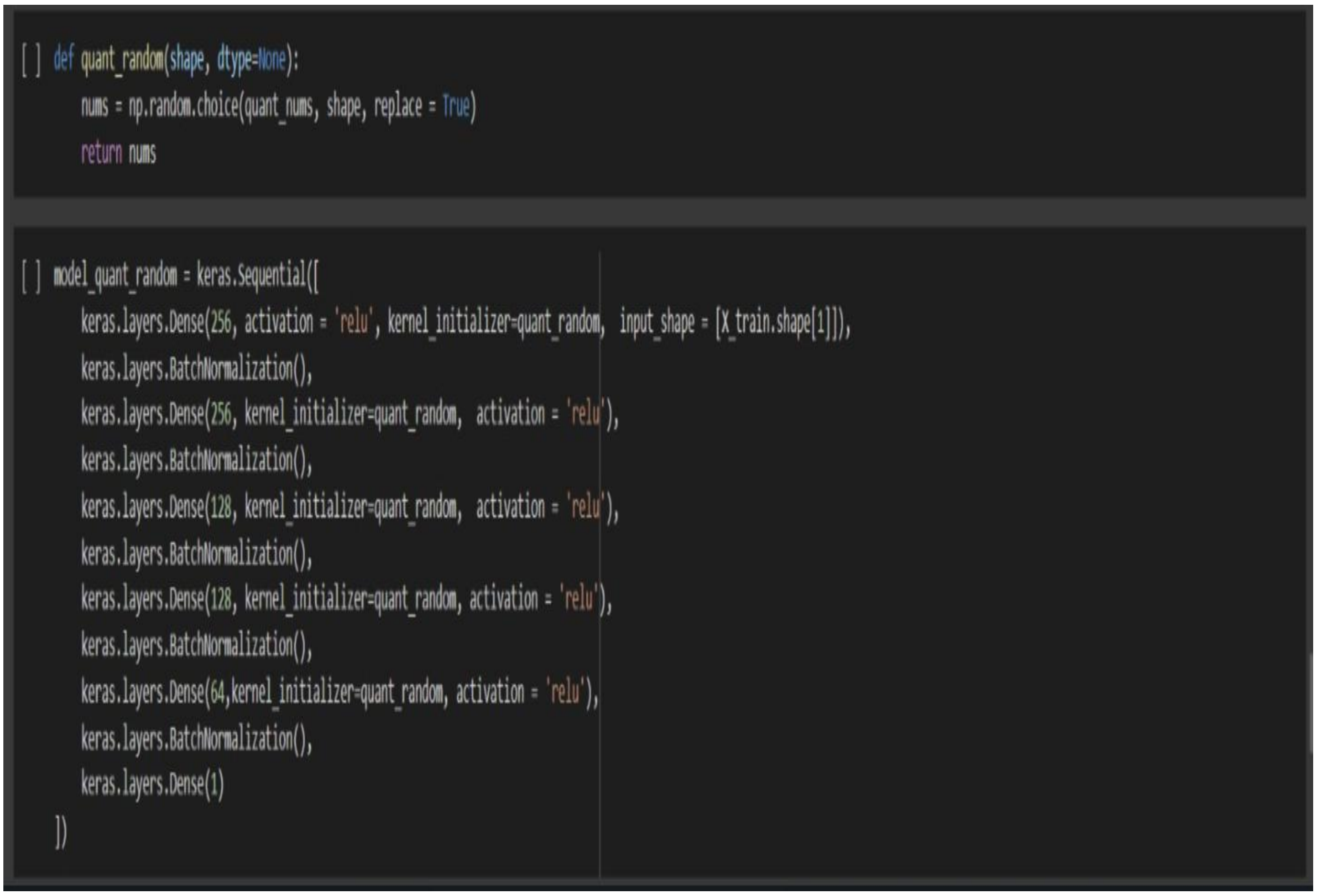
**Figure 15. Function to run the models multiple times**



```
[ ]  def psuedo_random(shape, dtype=None):
         nums = np.random.randint(-100, 100, shape)
         return nums
```

```
[ ]  model_psuedo_random = keras.Sequential([
         keras.layers.Dense(256, activation = 'relu', kernel_initializer = psuedo_random,  input_shape = [X_train.shape[1]]),
         keras.layers.BatchNormalization(),
         keras.layers.Dense(256, kernel_initializer = psuedo_random,  activation = 'relu'),
         keras.layers.BatchNormalization(),
         keras.layers.Dense(128, kernel_initializer = psuedo_random,  activation = 'relu'),
         keras.layers.BatchNormalization(),
         keras.layers.Dense(128, kernel_initializer = psuedo_random, activation = 'relu'),
         keras.layers.BatchNormalization(),
         keras.layers.Dense(64,kernel_initializer = psuedo_random, activation = 'relu'),
         keras.layers.BatchNormalization(),
         keras.layers.Dense(1)
     ])
```

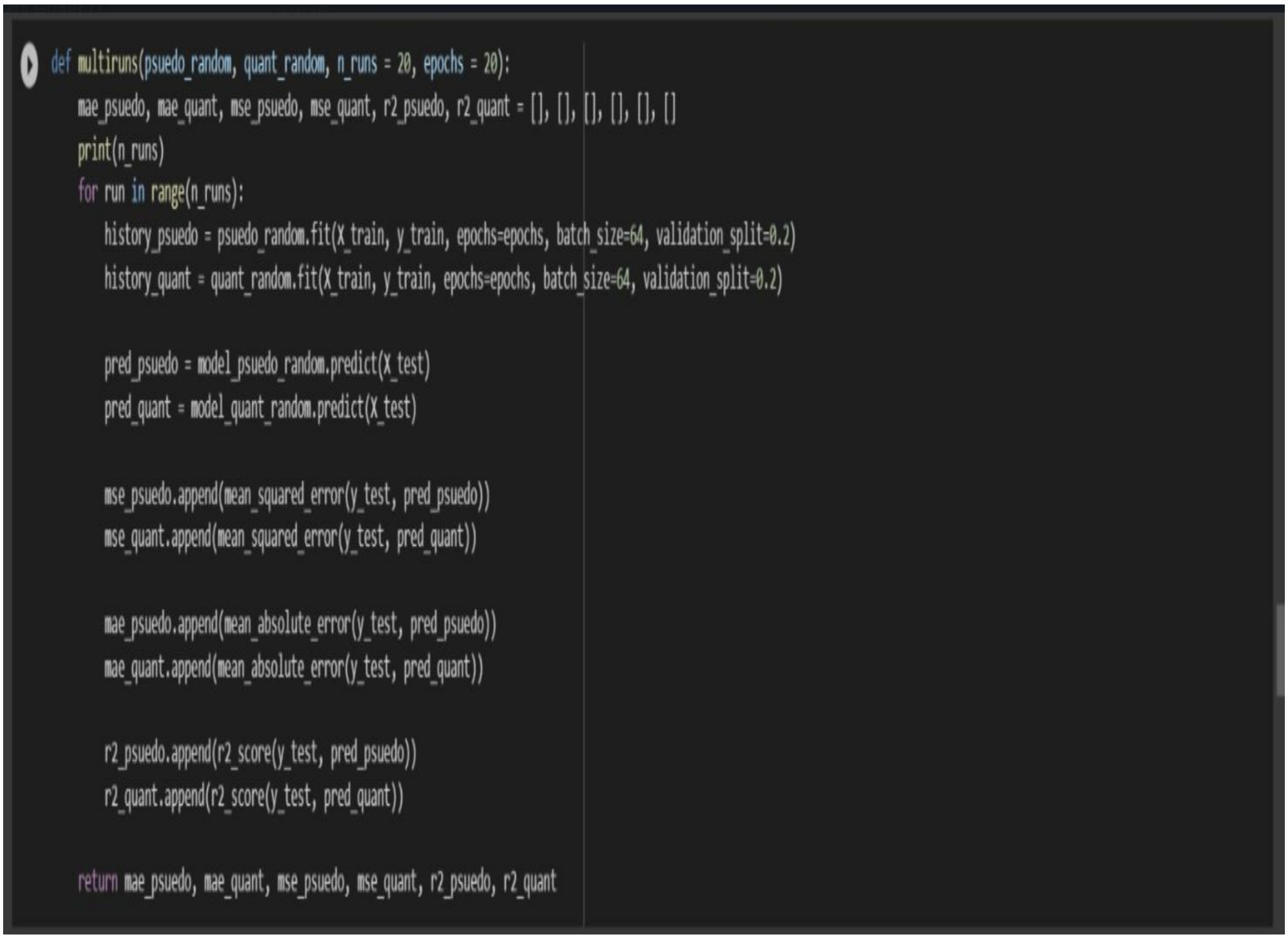
**Figure16. Pseudo Random Initlialization of ANN Model**



```
[ ] def quant_random(shape, dtype=None):
        nums = np.random.choice(quant_nums, shape, replace = True)
        return nums
```

```
[ ] model_quant_random = keras.Sequential([
        keras.layers.Dense(256, activation = 'relu', kernel_initializer=quant_random,  input_shape = [X_train.shape[1]]),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(256, kernel_initializer=quant_random,  activation = 'relu'),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(128, kernel_initializer=quant_random,  activation = 'relu'),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(128, kernel_initializer=quant_random, activation = 'relu'),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(64,kernel_initializer=quant_random, activation = 'relu'),
        keras.layers.BatchNormalization(),
        keras.layers.Dense(1)
    ])
```

**Figure 17. Quantum Random initialization of ANN model**



```python
def multiruns(psuedo_random, quant_random, n_runs = 20, epochs = 20):
    mae_psuedo, mae_quant, mse_psuedo, mse_quant, r2_psuedo, r2_quant = [], [], [], [], [], []
    print(n_runs)
    for run in range(n_runs):
        history_psuedo = psuedo_random.fit(X_train, y_train, epochs=epochs, batch_size=64, validation_split=0.2)
        history_quant = quant_random.fit(X_train, y_train, epochs=epochs, batch_size=64, validation_split=0.2)

        pred_psuedo = model_psuedo_random.predict(X_test)
        pred_quant = model_quant_random.predict(X_test)

        mse_psuedo.append(mean_squared_error(y_test, pred_psuedo))
        mse_quant.append(mean_squared_error(y_test, pred_quant))

        mae_psuedo.append(mean_absolute_error(y_test, pred_psuedo))
        mae_quant.append(mean_absolute_error(y_test, pred_quant))

        r2_psuedo.append(r2_score(y_test, pred_psuedo))
        r2_quant.append(r2_score(y_test, pred_quant))

    return mae_psuedo, mae_quant, mse_psuedo, mse_quant, r2_psuedo, r2_quant
```

**Figure 18. Multi run function for ANN model**

### 4.2.1 Comparison of performance of all soft computing Techniques

| **S. No** | **Algorithm** | **Dataset Used** | **Generator Used** | **Performance Parameters** | | |
|---|---|---|---|---|---|---|
| | | | | **MSE** | **MAE** | **R2 Score** |
| 1. | ANN | Ground Water Quality | Pseudorandom | 16.2 | 2.75 | 0.58 |
| | | | Quantum Random | 8.5 | 2.50 | 0.96 |
| 2. | CNN | Tensorflow Images | Pseudorandom | 20.47 | 3.69 | 0.48 |
| | | | Quantum Random | 10.50 | 2.70 | 0.73 |
| **Other Algorithms** | | | | **Accuracy** | **Precision** | **F1 Score** |
| 3. | Bi-directional LSTM | Speech Text | Pseudorandom | 0.54 | 0.86 | 0.8816 |
| | | | Quantum Random | 0.55 | 0.88 | 0.8875 |

**Table 1. Results Analysis**

## 4.3 : Risk Analysis and Mitigation:

| Risk Id | Classification | Description | Risk Area | Probability | Impact | RE(P*I) |
|---|---|---|---|---|---|---|
| 1 | Technical | Overfitting | Accuracy and reliability of model | High | High | 25 |
| 2 | Performance | Inaccurate Classification | Incorrect predictions and classifications | Medium | High | 15 |
| 3 | Data | Biased and Inaccurate result | Insufficient Data | Medium | Medium | 9 |
| 4 | Technology | Computational complexity | Long Training times | Medium | Medium | 9 |
| 5 | Implementation | Implementation errors | Incorrect and inconsistent result | Low | Low | 1 |
| 6 | Security | Security Vulnerability | Security threats | High | High | 25 |

**Table 1. Risk Analysis**

# Chapter 5 Testing

## 5.1 Testing Plan

| **Type of Test** | **Comments/Explanations** | **Software Component** |
|---|---|---|
| Unit Testing | Tests individual functions and methods to ensure they are working as expected. | pH Predictor, data pre-processing, imputation, Numeric conversion, Deep Neural Network |
| Integration Testing | Tests how different components of the application work together, including how the machine learning models integrate with the web interface. | Ph Predictor |
| Acceptance Testing | Tests how the application functions from the perspective of the end user, ensuring that it meets their requirements and expectations. | Deep neural network performance analysis |
| Performance Testing | Tests the speed, stability, and scalability of the application under different load conditions. | Deep Neural Network |

**Table 2: Testing Plan**

## 5.2 Component decomposition and type of testing required

| S.No | List of Component | Type of Testing | Technique for Test Case Writing |
|---|---|---|---|
| 1 | pH Predictor | Unit Testing, Integration Testing, Acceptance Testing | Test-Driven Development |
| 2 | Data preprocessing | Unit Testing, Integration Testing, Acceptance Testing | Test-Driven Development |
| 3 | imputation | Unit Testing, Integration Testing, Acceptance Testing | Test-Driven Development |
| 4 | Numeric conversion | Unit Testing, Integration Testing, Acceptance Testing | Test-Driven Development |
| 5 | Deep Neural network Analysis | Unit Testing, Integration Testing | Test-Driven Development |
| 6 | Quantum Random Numbers | Unit Testing, Integration Testing | Test-Driven Development |

**Table 3. Component decomposition and type of testing required**

# Chapter 6 : Findings, Conclusion, and Future Work:

## 6.1 Findings:

This project concludes on use of quantum-random number generators (QRNGs) in soft computing can potentially lead to improved performance over traditional pseudorandom number generators (PRNGs), especially in cases where true randomness is crucial for the task. The study examined the use of QRNGs and PRNGs in various soft computing approaches, including Artificial neural networks, CNNs and Bi-directional LSTM.

The experiments showed that the use of QRNGs led to statistically significant improvements in classification accuracy in some cases, while in others, the differences were not significant. The researchers suggest that further research is needed to fully explore the potential of QRNGs in soft computing, including investigating the effects of different types of QRNGs and exploring their use in other soft computing approaches.

Nevertheless, the results of this study suggest that QRNGs are a promising tool for improving the performance of soft computing systems, particularly in tasks where true randomness is essential. The research also highlights the importance of careful experimental design and statistical analysis when comparing different approaches, particularly when dealing with small sample sizes or noisy data. In addition, the study emphasizes the need for transparency and reproducibility in research involving machine learning and other data-driven approaches.

The researchers provide detailed descriptions of their experimental protocols and code, making it easier for others to replicate and build upon their work. Overall, the study contributes to the growing body of research exploring the potential of quantum computing and quantum-inspired approaches in machine learning and other data-driven fields. The results suggest that quantum randomness can offer significant advantages in certain contexts, and further investigation of these approaches is warranted.

## 6.2 Conclusion:

The conclusion of this project is that the use of quantum-random number generators (QRNGs) can lead to improved performance in soft computing applications over traditional pseudorandom number generators (PRNGs). The study investigated the use of QRNGs and PRNGs in various soft computing approaches, including random neural networks, CNNs and Bi-directional LSTM.. The results showed that the use of QRNGs in some cases led to statistically significant improvements in classification accuracy, highlighting the potential advantages of true randomness. However, the study also revealed that the performance gains from QRNGs are not universal and depend on the specific task and algorithm used. Therefore, further research is needed to fully explore the potential of QRNGs in soft computing, including investigating the effects of different types of QRNGs and exploring their use in other soft computing approaches. The researchers also emphasized the importance of careful experimental design and statistical analysis when comparing different approaches, especially when working with small sample sizes or noisy data. They also highlighted the need for transparency and reproducibility in research involving machine learning and other data-driven approaches. In conclusion, the study contributes to the ongoing research on the potential of quantum computing and quantum-inspired approaches in machine learning and other data-driven fields. The findings suggest that QRNGs can offer significant advantages in certain contexts and highlight the importance of further investigation of these approaches. Furthermore, the study also provides insights into the challenges of using QRNGs in practical applications. For example, QRNGs may be more difficult and expensive to implement than traditional PRNGs, and their use may be limited by factors such as hardware availability and cost. The study also highlighted the importance of considering the ethical and societal implications of using such advanced technologies, especially when they involve sensitive data or decision-making processes. Overall, the study contributes to the ongoing efforts to understand the potential benefits and limitations of QRNGs in soft computing and provides a foundation for future research in this area. It highlights the importance of careful experimental design and statistical analysis, as well as the need for transparency and reproducibility in research involving machine learning and other data-driven approaches. Ultimately, the study provides valuable insights into the potential of quantum-inspired approaches to improve the performance and capabilities of soft computing applications, while also highlighting the challenges and limitations that must be addressed in order to fully realize these benefits.

## 6.3 Future Work :

There are several areas for future work related to this project. One area is to explore the use of QRNGs in other soft computing techniques beyond the ones explored in this study. Another area is to investigate the use of hybrid approaches that combine QRNGs with traditional PRNGs or other randomization techniques to improve the robustness and performance of soft computing models.

Another avenue for future research is to investigate the use of quantum computing hardware, such as quantum annealers and quantum simulators, to perform soft computing tasks. This could involve exploring new algorithms and techniques that are specifically designed for quantum hardware, as well as investigating the performance of existing techniques on quantum platforms.

Finally, it is important to continue exploring the practical applications of QRNGs in soft computing and to address the challenges and limitations that were identified in this study, such as the cost and availability of hardware, the ethical implications of using advanced technologies, and the need for rigorous experimental design and statistical analysis. This could involve working with industry partners to develop and test QRNG-based soft computing solutions in real-world settings and collaborating with policymakers and regulators to ensure that these technologies are deployed in a responsible and ethical manner.

Additionally, it would be worthwhile to investigate the use of different types of QRNGs and to compare their performance with the QRNGs used in this study. Investigating the performance of different types of QRNGs in soft computing could provide valuable insights into the strengths and weaknesses of each approach.

Another area for future research is to investigate the use of quantum-inspired algorithms in soft computing. These algorithms are designed to mimic some of the properties of quantum computing, such as superposition and entanglement, using classical hardware. While these algorithms are not as powerful as true quantum algorithms, they can still provide some benefits over traditional algorithms in certain applications.

Finally, it is important to continue investigating the fundamental properties of QRNGs and their implications for soft computing. This could involve exploring the limits of randomness that can be achieved with QRNGs, developing new theoretical frameworks for analyzing the performance of QRNG-based models, and investigating the effect of numerous experimental parameters on the performance of QRNGs in soft computing. Overall, the research presented in this project provides valuable insights into the use of QRNGs in soft computing and opens up many avenues for future research in this exciting and rapidly evolving field.

## References:

1. Antonio Acín, Luis Masanes, and Nicolas Gisin. "Quantum random number generators." https://arxiv.org/pdf/0911.3427.pdf [Dec. 2020]
2. Jordan J. Bird, A. Ekárt, Diego R. Faria. “On the effects of pseudorandom and quantum-random number generators in soft computing.” https://www.researchgate.net/publication/336854512_On_the_effects_of_pseudorandom_and_quantum-random_number_generators_in_soft_computinge [June 2020]
3. Matthieu Komorowski , Dominic C. Marshall, Justin D Salciccioli, Yves Crutain. “Exploratory Data Analysis.” https://www.researchgate.net/publication/308007227_Exploratory_Data_Analysis. [Sept. 2016]
4. Markus Müller, Peter Hänggi. "A comparison of the statistical properties of quantum and pseudorandom number generators."https://doi.org/10.1209/epl/i2002-00277-9. [Dec. 2020]
5. Martin Stutzmann, Matthias Hiller, Matthias Breyer. "Performance of Quantum Random Number Generators." https://www.mdpi.com/1099-4300/22/12/1381/pdf [Dec. 2020]
6. Valerio Scarani, Christian Kurtsiefer."Quantum Random Number Generation with Entangled Photons." https://arxiv.org/pdf/0706.4165.pdf [Aug. 2020]
7. “Visual Paradigm.” https://online.visual-paradigm.com/ [accessed Apr. 2023]