ACADEMIA QUANTUM

Research Article
Published: 2025-05-26
https://doi.org/10.20935/AcadQuant7699

A quantum reservoir computing approach to computer-aided music composition

Eduardo Reck Miranda[1,2,*], Hari Vignesh Shaji[1,2]

Academic Editors: Misha Erementchouk, Hichem Eleuch

Abstract

This paper presents an approach to developing computer-aided music composition (CAMC) systems with quantum computing. CAMC systems are aimed at supporting musicians in creating music rather than generating complete pieces of music autonomously. For instance, they may generate ideas or prompts to be further developed during the creative process. We developed a CAMC system using a machine learning method called Quantum Reservoir Computing (QRC) that learns to create tunes in the style of given examples. The system proved to run satisfactorily on the Noisy Intermediate-Scale Quantum (NISQ) machines available today. It does not require large amounts of qubits to outperform equivalent classical Deep Learning methods. Furthermore, it can serve the purposes of a CAMC system without the need for big data for training. This paper provides a general introduction to reservoir computing fundamentals and explains how quantum mechanics can harness its performance. Then, it walks the reader through the development of our system and presents demonstrations. A comparative experiment against a classic Recurrent Neural Network (RNN) model is also discussed.

Keywords: *quantum computing, quantum computer music, quantum reservoir computing, machine learning, generative AI, music technology*

Citation: Miranda ER, Shaji HV. A quantum reservoir computing approach to computer-aided music composition. *Academia Quantum* 2025;2. https://doi.org/10.20935/AcadQuant7699

1. Introduction

1.1. Motivation

We are interested in developing quantum computing technology for Computer-Aided Music Composition (CAMC) systems [1–4]. These are software tools that assist composers in the analysis and creation of music, enabling them to explore musical ideas in ways that they would possibly not have explored otherwise. Distinguished examples of CAMC systems range from programming tools such as OpenMusic [5] and Opusmodus [6], to ready-made applications such as Band-in-a-Box [7].

Generative algorithms powered by Artificial Intelligence (AI) technology constitute an important component of CAMC systems. Several generative AI techniques for music were developed within the last 50 years or so [8–12], which have been embedded in CAMC systems in one way or another. For instance, Band-in-a-Box incorporates generative AI to make automatic accompaniments [13, 14].

Recently, the emergence of Large Language Models (LLMs) for natural language processing using Deep Learning methods [15], such as Recurrent Neural Networks (RNNs) [16] and Transformers [17], instigated the development of look-alike AI for music. Thought-provoking attempts at using Deep Learning to generate music include MuseNet by OpenAI [18] and Music Transformer by Magenta [19].

Notwithstanding the potential of Deep Learning methods for generating music, their usefulness for CAMC is hampered by the need for extraordinarily huge datasets for training. This is inconvenient because such large datasets are not always readily available. Moreover, they require tremendous computational resources for processing, which only large corporations can afford.

We note that CAMC systems should not need to learn from vast datasets. CAMC is neither aimed at embodying some sort of musical artificial general intelligence [20] or required to make fully fledged pieces of music autonomously. Rather, they are aimed at adapting to the needs of specific users to support them in creating music.

As an alternative way forward, we are exploring emerging quantum computing technology for CAMC [21]. Quantum computing leverages quantum mechanical phenomena such as superposition, entanglement, and interference, to process information. It handles information in different ways, with the potential to break new ground for unconventional alternatives to AI. We believe this can lead to novel methods and approaches to creating music. However, realising this potential requires overcoming significant technical challenges in the physical implementation of *qubits* (the quantum

[1]Interdisciplinary Centre for Computer Music Research (ICCMR), School of Art, Design and Architecture, University of Plymouth, Plymouth PL4 8AA, UK.
[2]Moth, Somerset House, West Wing, London WC2R 1LA, UK.
*email: eduardo.miranda@plymouth.ac.uk

computing counterpart to bits in classical computing to represent information), error correction, and scalability [22, 23].

Quantum computers today are in a phase of development coined by John Preskill as Noisy Intermediate-Scale Quantum (NISQ) [24]. NISQ is characterised by devices with a small number of qubits—ranging from tens to hundreds at the time of writing—and limited fidelity due to noise and errors in quantum operations. The industry is racing towards the next phase: the Fault-Tolerant Quantum Computing (FTQC) era. Nevertheless, NISQ devices represent an important development toward practical and scalable quantum computing technologies.

In this paper, we introduce a Quantum Reservoir Computing (QRC) approach to building AI for CAMC systems. QRC is a machine learning method that works satisfactorily on NISQ devices. However, make no mistake, this does not mean that QRC would not benefit from advances towards the FTQC era. On the contrary. Even though recent research demonstrated that a type of quantum noise referred to as 'amplitude damping' can be beneficial for QRC under some controlled circumstances, other types of noise found in quantum devices caused by phase damping and depolarising effects are detrimental [25]. The jury is out. Much research is required to understand which types of quantum noise might be harnessed for QRC. Nevertheless, there is good evidence that QRC is robust for current NISQ technology and, as we demonstrate below, can be useful for CAMC systems.

The remainder of this paper is structured as follows: it begins with a general introduction to the notion of reservoir computing and explains how quantum mechanics can harness its performance. Then, we briefly review previous research, which demonstrated the advantages of QRC over classical Deep Learning machine learning methods.

Next, we discuss issues pertaining to quantum reservoir design for machine learning and explain what is required to process music effectively. Then, we look into possible ways to build QRC systems and highlight the challenges of a music system. After this, we introduce a proof-of-concept system and a demonstration. Here, we show how the system can be leveraged to control the musical output. Before the concluding section, we present an experiment comparing our QRC method with a classical Recurrent Neural Network called an LSTM (Long Short-Term Memory) network [26] and discuss the results.

1.2. Reservoir computing

Reservoir computing (RC) is a computational framework, which emerged from efforts to understand and model the dynamics of complex systems, particularly biological neural networks, and their ability to process and adapt to temporal information [27]. In computational neuroscience, RC constitutes an important approach to model the behaviour of the pre-frontal cortex of the human brain with trainable readouts [28].

Technically, the RC framework is based on a Recurrent Neural Network (RNN) architecture where a fixed, randomly connected recurrent network of artificial neurons (the reservoir) transforms input signals into a high-dimensional space [27]. The presence of feedback, or recurrent, connections is a key aspect of handling temporal information, making RC suitable for tasks involving temporal patterns, such as series prediction, speech recognition, and music [29].

In RC, input signals are fed into the reservoir, where they create dynamic patterns of activity. A linear regression model is then trained on the outputs from the reservoir to produce desired outcomes. Note that the internal connections of the RC network are typically not trained; only the output layer of the network undergoes training.

Comparatively, in a typical classical Deep Learning system, all layers of the network need to be trained through back-propagation. This is an intensive process, which renders the method inefficient for handling sequential data. RC is an economical alternative to classical Deep Learning because only the output layer is trained.

In a recent study, we compared a classical RC model against a highly valued neural network model, known as Long Short-Term Memory (LSTM), available in the TensorFlow library [30], to learn music (also see Section 4). We demonstrated that the reservoir approach brings a significant reduction in trainable parameters for equivalent learning [31]. The RC model required considerably fewer resources to process and learn the same amount of information than the LSTM model.

The reduction in optimisation requirements contributes to the simplicity and efficiency of the reservoir method. A reservoir was used in machine learning to reduce the complexity of training RNNs before gradient descent algorithms were mastered and widely adapted [27]. However, with the use of a gradient descent algorithm, tackling more data-intensive tasks in modern applications can become prohibitive. RNNs require an unfolding process for training, referred to as Back-Propagation Through Time (BPTT), which is computationally demanding [29]. To address the problems associated with RNNs, new architectures were introduced for sequence modelling, such as the abovementioned LSTM network and Transformer-based models. Still, despite the improved learning performance of these new architectures, they are inefficient in terms of computational resources required to train them.

Another important difference between RC and classical Deep Learning methods is RC's ability to support multitasking. That is, multiple output layers can simultaneously learn different tasks. Whereas the hidden reservoir layer is fixed, the output layers can be reused. Moreover, increasing the number of those output layers would not affect performance. This is not possible to achieve with classical Deep Learning methods.

As a generalisation, a reservoir does not necessarily need to be implemented as an RNN model. Quoting Jaeger [32], "A full theory of reservoir dynamics would be a full theory of everything that evolves in time". Thus, almost any arbitrary system sporting a fixed network of nodes with recurrent connections can act as a reservoir. Indeed, the reservoir can also be a physical substrate [33]. For instance, one of the early experiments with a physical reservoir used a bucket of water to build a simple system for pattern recognition [34]. The possibility of using a physical reservoir paves the way for the development of new kinds of hardware based on a neuromorphic architecture. Here, the memory and processing units are integrated and the processing takes place asynchronously [35].

1.3. Quantum reservoir

In a nutshell, Quantum Reservoir Computing (QRC) replaces the fixed RNN of the RC framework with a quantum circuit. QRC utilizes quantum mechanics principles to create the reservoir, employing quantum states and operations [36]. This leverages quantum superposition and entanglement to expand the input into a hyper-dimensional space in ways that would be unachievable using a classical RNN [37]. Additionally, the choice of measurement basis, which exponentially scales with the number of qubits, provides a rich feature space for machine learning tasks.

QRC was first introduced in 2017 for temporal learning tasks [38]. Even with just a handful of qubits, QRC can exhibit powerful performance comparable to hundreds of neurons in artificial neural network approaches.

Thus, in contrast to treating a quantum computer as a logical computational device, the Hilbert space presented by a quantum system can be harnessed as a physical substrate for reservoir computing without the need for optimising variational circuit parameters that most quantum machine learning (QML) algorithms would require [39].

Several QML models, such as RNNs that use quantum processing in one way or another [40], are quantum counterparts of classical machine learning algorithms that are often well suited to run in an ideal fault-tolerant setting. Hence, they may experience training bottlenecks on a noisy device [41].

In contrast, QRC is generic and considers quantum dissipative dynamics arising in noisy quantum devices as a means of computation [42]. Hence, NISQ devices can be useful for building QRC-based AI systems [43].

Different types of quantum computing hardware have been experimented with for QRC, where the qubits can be made with different substrates, such as niobium or aluminium (for their superconductivity), trapped ions, optical fibres, and electron or nuclear spins in materials like silicon or diamond. A recent study used Rydberg atoms, which are highly excited atoms characterized by one or more electrons that have been excited to very high principal quantum levels, usually significantly higher than the ground state. Rydberg atoms exhibit strong dipole–dipole interactions with each other, which can be exploited in quantum computing and quantum simulation applications. The authors concluded that QRC was able to evolve probability distributions faster than a classical RNN. They also demonstrated that their QRC model displayed a memory capacity that outpaces that of any classical RNN [44].

The behaviour of a quantum reservoir can be mathematically described using Hamiltonian matrices, or operators, which determine how a quantum state evolves in time.

The evolution of a pure quantum state Ψ in time t can be derived from Schrödinger's equation for a closed quantum system as shown in Equation (1), where U is the Unitary evolution given as $e^{-iH\tau}$, with H as the Hamiltonian operator and τ an evolution in time.

$$|\Psi_t\rangle = U|\Psi_{t-1}\rangle \tag{1}$$

Ideally, the quantum state should be represented here as a density matrix ρ rather than as a wave function Ψ. Wave functions tend to lose their purity and become statistically mixed due to noise or upon measurement. In this case, the evolution can be derived from the von Neumann equation shown in Equation (2):

$$\rho_t = U\rho_{t-1}U^\dagger \tag{2}$$

The above can be generalised by considering the evolution through quantum channels rather than Unitary transformations, as shown in Equation (3). In this case, T is a Completely Positive Trace-Preserving (CPTP) map that accounts for Unitary operations along with any naturally occurring noise [43].

$$\rho_t = T\rho_{t-1} \tag{3}$$

2. Materials and methods

2.1. Quantum reservoir design and criteria for learning

In the initial implementations of the QRC framework [33, 38], the reservoir used to be based on the Hamiltonian H of a fully connected Ising model, which simulates the non-linear dynamics arising from interacting spins in an ensemble of quantum subsystems, or qubits. In Equation (4), X and Z are Pauli matrices [36], which are used to describe quantum operations acting on qubit pairs i and j. The Hamiltonian parameters J and h, representing the inter-qubit interaction strength and the magnetic coupling, respectively, are randomly set to establish the reservoir. In this setup, the information encoded through a single auxiliary qubit traverses through the ensemble of subsystems in the reservoir, evolving it in time.

$$H = \sum J_{i,j}X_iX_j + h_iZ_i \tag{4}$$

More recent QRC implementations [43, 45] proposed architectures with gate-based implementation on NISQ devices. In this case, the quantum dissipative dynamics come into effect as the systems undergo interaction with surroundings subject to noise and decoherence, which can be described as a CPTP map (Equation (3)). This phenomenon, which would otherwise be undesirable in conventional applications of quantum computing, shows the relevance of reservoir computing in the NISQ era. Some of the recent architectures adapted for NISQ devices are discussed in Section 2.2.

Since the fixed setup of a reservoir can simulate any arbitrary quantum system, a recent study showed the opportunity for the optimal design of a quantum reservoir [46] where circuits with significantly fewer quantum gates show better performance than a circuit simulating the commonly used Ising model described in Equation (4). Section 3 shows how this is adapted to reduce the complexity of a reservoir circuit.

Although the choice of a physical system for a reservoir can be arbitrary, the key property that makes a reservoir—or indeed any RNN—capable of processing sequences effectively is its ability to create a high-dimensional, non-linear, and temporal expansion of the input [27]. The requirements to meet this property and associated hyperparameters, along with the ones specific to reservoir computing theory, are discussed below.

2.1.1. Dimensionality

Higher dimensionality enhances the separation ability of the reservoir to process input effectively and improve its memory capacity. While this may hold for artificial neural networks, they differ in important aspects compared to natural, or biologically informed, processing architectures. Indeed, reservoir computing has been used to describe the hypothesis in neuroscience that the higher cognitive functions of the brain are computations performed in high-dimensional state space with non-linear dynamics [47].

The size of a reservoir, referred to as *dimensionality,* can be specified by the degrees of freedom in its measured response. In the context of artificial neural networks, this corresponds to the number of neurons in the network. However, for a quantum system, the dimensionality can be considered as 2^n states for an n-qubit configuration measured in a given basis, e.g., computational basis, or Z basis. Increasing the number of qubits results in exponentially high degrees of freedom.

The underlying relationships between inputs are often complex and non-linear. These relationships become linearly separable in the Hilbert space; hence, the use of a linear readout layer at the output is often sufficient to map the measurements to desired targets. A similar method was explored in Quantum Kernel methods [48, 49] to exploit the dimensionality of the Hilbert space in enhancing learning performance.

2.1.2. Non-linearity

The neurons in a typical artificial neural network are activated or deactivated based on the activation function, which defines the behaviour of the network. This mechanism is inspired by the synaptic plasticity that a biological neuron undergoes. Non-linear activation functions are used to learn complex relationships in data. Some of the common activation functions used in RNNs are *TanH* and *ReLU* [50].

In a quantum reservoir network, the non-linear dynamics can be introduced by quantum operations such as qubit rotations and entanglements. If an **RY** rotation gate is used, the Unitary transformation on a qubit state dependent on input x is represented as shown in Equation (5). Here, Y is a Pauli operator that rotates a qubit by 180° (π radians) around the y-axis, as described in Equation (6).

$$U(x) = RY(x) = e^{-ixY} \tag{5}$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{6}$$

Various types of non-linear transformations of input can be achieved using different quantum gate operations. For example, **Figure 1** shows a simple circuit consisting of a rotation gate **RY** and a gate **CX** (also written as **CNOT**, or conditional not) that together apply a non-linear and multi-dimensional ($2^2 = 4$ states) transformation on a simple mono-dimensional input φ.

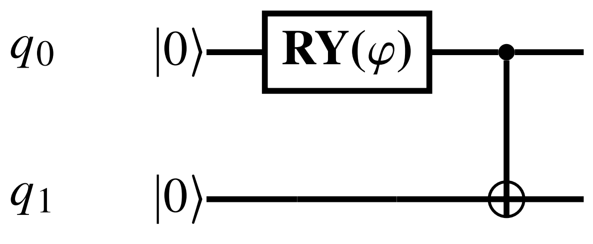

Figure 1 • Non-linearity with quantum gates.

2.1.3. Recurrence

Non-linearity and high dimensionality create a rich representation of the input. However, to take time into account, a reservoir network must contain *recurrence*. This makes any input fed into the network reverberate for a longer period. This functions as a short-term memory. The output at a given time depends on the current input and also on the fading memory of past inputs interacting with it. Hence, the network will now have a sense of order. Feeding the same inputs in different orders will result in different outputs. This allows for the capturing of temporal patterns and dependencies present during the training of a sequence.

For a single unit in a recurrent network, the output at a time step t can be described as shown in Equation (7), where the predicted output y_t at a time t is a function of the current input x_t and the previous memory state h_{t-1}. Here, f can be any activation function discussed in Section 2.1.2.

$$y_t = f(x_t, h_{t-1}) \tag{7}$$

Recurrence is common in biological neural networks with the presence of high recurrence even in the nervous systems of the brains of simple animals [27]. For physical reservoir computing, although not all physical systems have an intrinsic recurrence, it can be realised by establishing a connection between the output and the input. For example, a physical reservoir computing experiment based on organic electrochemical transistors in an electrolyte solution had conductive polymer fibres connected to both the input and output of the reservoir, creating recurrence [51]. In QRC, recurrence can be introduced in different ways, depending on the type of architecture that is adopted. This is discussed further in Section 2.2.

The updating of internal memory due to recurrence can be further controlled with a parameter referred to as a leaking rate, represented as ε. This governs how much of past information is to be retained or leaked through each time step. The effect of the leaking rate can be described as shown in Equation (8).

$$h_t = (1-\varepsilon) \times h_{t-1} + \varepsilon \times f(x_t, h_{t-1}); \qquad 0 \leq \varepsilon \leq 1 \tag{8}$$

The leaking rate is a hyperparameter that can be tuned to optimise a reservoir based on the task in question. For instance, if the leaking rate is set to a maximum value, $\varepsilon = 1$, then the previous states would not leak into the current state. This impacts the performance of the learning model.

2.1.4. Echo state property

In addition to the properties discussed above, which form common criteria for learning in both the reservoir and RNNs, the random and fixed nature of a reservoir network is also required to satisfy a condition known as the *echo state property*. The property asserts that the output at any given time should depend on the current fading memory rather than on initial conditions [33].

Essentially, the effect of initial conditions should be washed out gradually, to ensure that the random initialisation of parameters does not affect the response of the following time steps. Hence, the RC framework uses a parameter $T_\triangle$, referred to as the *washout period,* during which the collected states from the reservoir are not trained. Only the input and output pairs after $t > T_\triangle$ are considered for training.

A stable reservoir is expected to satisfy the echo state property [27]. This can be validated by measuring the responses of the reservoir with different initial conditions and ensuring convergence. The time it takes for the responses to converge can be a good estimate to define the washout period. The echo state property is commonly represented with a hyperparameter called spectral radius ρ, which is a measure of the non-linear transformation capability of the reservoir. It is denoted as the largest eigenvalue of the internal weight matrix W of the reservoir. A common condition for stability is to satisfy $\rho(W) < 1$.

2.2. Architectures for music learning

A reservoir based on a quantum substrate can be harnessed for temporal information processing by devising a suitable architecture that satisfies the criteria discussed in Section 2.1. At a single time-step, the QRC process can be generalised as shown in **Figure 2**.

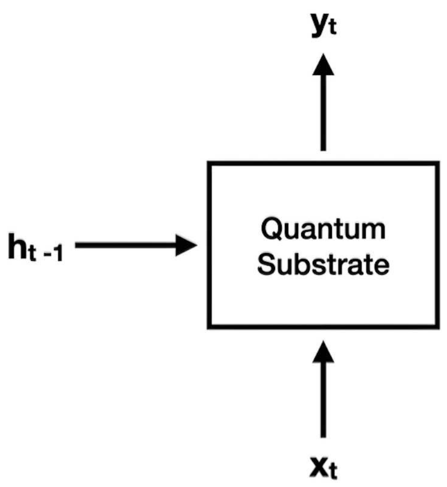

Figure 2 • General QRC process at given time step t.

According to Equation (7), the processing at a single time step has three main elements, irrespective of the type of architecture:

- A memory until the previous time step h_{t-1}.
- A current input x_t.
- A current output y_t.

A short review of different QRC architectures is described below (Section 2.2.1). The problems and limitations of implementing recurrence are discussed in Section 2.2.2 followed by a discussion in Section 2.3 on devising an architecture suitable for learning music.

2.2.1. QRC architectures

As mentioned in Section 2.1, in the original QRC framework [38], researchers simulated quantum dynamics using the Ising model shown in Equation (4). This can be simply described as the time evolution of injecting an input into the reservoir that spreads through the network and observing the states. A general workflow of such temporal learning architecture for music sequence modelling is shown in **Figure 3**.

As shown in Equation (1), the Unitary evolution is given as $e^{-iH\tau}$, with H as the Hamiltonian operator and τ an evolution in time. For learning a sequence consisting of k time steps t, the input x is encoded and fed through an auxiliary qubit q_0 and the reservoir is set to evolve at every i^{th} time step $t = i\tau$ with a total simulation time of $k\tau$. The results extracted from the simulation as reservoir states S_t can then be trained using a readout layer to learn a specific task such as non-linear mapping to output y, prediction, or classification.

This type of architecture has been recently demonstrated using gate-based quantum computers as an application for the temporal trajectory prediction of mobile wireless networks in comparison with artificial neural networks [52]. The simulation of analogue time evolution was performed on digital-gate-based systems by employing a technique referred to as Suzuki Trotterization. Effectively, this technique discretises the evolution time into smaller steps by decomposing the Unitary operator U into smaller components.

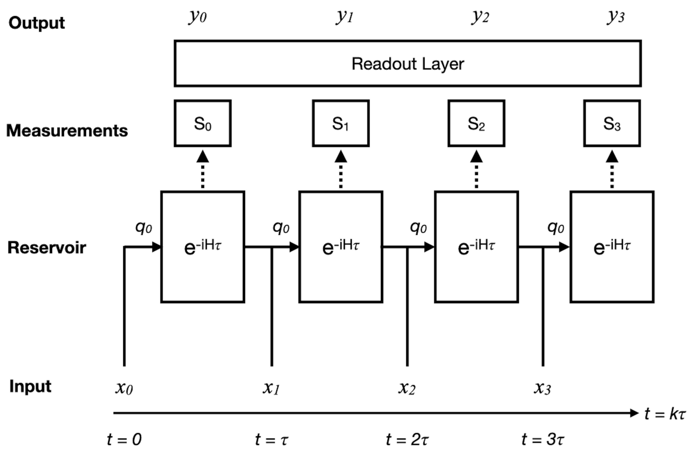

Figure 3 • General workflow of temporal learning for musical sequences.

Due to the increased number of gates required to implement the quantum dynamics in the above class, a new class of QRC architecture was developed for digital-gate-based implementation on NISQ devices for temporal learning [45]. Here, the reservoir dynamics is modelled digitally using arbitrary parameterised circuits instead of using the Ising Hamiltonian, bypassing the need for Suzuki Trotterization. The input is not directly injected but encoded as probabilities controlling the overall evolution of the reservoir: an input x_t at a time step t is encoded onto the state of a control qubit, which can be described in terms of density matrix ρ as stated in Equation (9). The input-dependent quantum reservoir dynamics at time t can be modelled using a CPTP map, as shown in (Equation (10)).

$$\rho_c = (x_t)\,|0\rangle\,\langle 0| + (1 - x_t)\,|1\rangle\,\langle 1| \tag{9}$$

$$\rho_t = T(x_t)\rho_{t-1} \tag{10}$$

The leaking rate ε is also integrated into the architecture as a control qubit. This qubit controls the probability of swapping (or resetting) the evolved state with an arbitrary state σ. In a nutshell, this models the rate of forgetting the initial state of the reservoir.

Following the gate-based QRC architectures, researchers demonstrated an implementation different from the previously discussed ones [43]. In this case, the Unitary evolutions are not arbitrary. They are parameterised directly by the input, encoded as angles, to evolve a default initial state of $|0\rangle$. This implementation is specifically designed to study the influence of naturally occurring noise in quantum hardware, as opposed to simulation on classical digital hardware. The Unitary-input-dependent circuit schematics are intentionally made simpler to let noise, such as decoherence and depolarising noise, contribute to the reservoir dynamics. Therefore, the CPTP map corresponds to the quantum device in operation.

2.2.2. Recurrence and measurement efficiency

We have discussed earlier that a quantum reservoir architecture needs to establish recurrence and sport an effective way of measuring the states at each time step. It is not straightforward to create feedback connections in quantum circuits. In quantum mechanics, the act of measurement collapses the quantum state. Therefore, at least for now, the circuit for the next time step must be re-run from the initial time step.

As discussed in Section 2.2.1, in most implementations of QRC frameworks in NISQ devices, increasing the sequence length extends the circuit horizontally with each time step. For instance, to measure the state S_3 in **Figure 3**, the circuit has to be re-run from the initial time step x_0, repeating the evolution circuit four times. This is far from efficient. It increases the circuit complexity and the time taken to run the model.

A typical sequence modelling task requires the following parameters:

- N = Number of iterations for training.
- B = Batch size of each iteration.
- L = Length of the sequence of each batch.

With back action in effect, the total circuit measurements required to collect data on reservoir states will be $N \times B \times L$, which will equal several thousands of circuits growing in length.

This restarting measurement protocol can be realised to be efficient if the echo state property of a reservoir is taken into account. By introducing a washout period W, the measurement restarting time step is now made to be $x_W - x_L$ instead of x_0. This saves the projective measurement of $N \times B \times W$ circuits. Exploiting this property even further, recent research [52] suggested a rewinding protocol. In this case, the total execution length can be fixed to W, sliding the restarting time step to $x_{(t-W)-x_t}$ for the measurement of a state at time t. Hence, experimentation with different measurement protocols may provide insight into the efficiency of scaling reservoirs in NISQ devices.

2.3. The architecture for the proof-of-concept

Given the requirements and alternatives discussed above, this section introduces the architecture that we developed for the proof-of-concept system.

Recurrence can be achieved with different quantum reservoir architectures and measurement protocols, as discussed above. However, most temporal learning experiments with QRC were demonstrated for 1-D time series, where the input is encoded through a single auxiliary qubit [38], a control qubit [45], or a gate parameter [43]. This can be considered a limitation if the input is a sequence of discrete elements that is to be encoded as a high-dimensional vector. Music is a case in point. Along with this, an architecture for music should be flexible for reconfiguration.

A general design criterion for an RNN to learn music is to handle variable-length input and output sequences [29]. Therefore, the same model used for training can be reconfigured for generation following an initial sequence of any length. With an RNN, music sequences are commonly trained with many-to-many configurations, where a target output sequence is a continuation of the input sequence to keep track of the temporal dependencies. To generate novel music from a trained model, the training should be reconfigured as a one-to-many scheme. In this case, the system would be able to follow a randomly initialised (or a user-given) note and continue generating notes one after another automatically. Considering these requirements for music learning and the architectures reviewed above, we require a simpler and more efficient architecture for music learning, which closely resembles the architecture of an RNN, supporting recurrence, flexibility, and high-dimensional input encoding.

Accordingly, we developed an architecture inspired by a recent work on QRC for the study of fluid and thermodynamics [53]. This work is not directly applied to sequence modelling. However, we successfully adapted it for sequence modelling, as demonstrated in Section 3.

In our case, the circuit length is fixed for each time step but uses a feedback strategy similar to an RNN, where the memory of previous time steps is fed back to the input externally, as shown in **Figure 4**. The workflow of the architecture has three blocks of Unitary evolution. The circuit is initialised with state $|0\rangle$, and a Unitary evolution dependent on previously measured probability amplitudes h_{t-1} is applied. This follows with the application of Unitary evolution based on current input x_t. This is subsequently followed by an arbitrary Unitary evolution with a random set of parameters β, which acts as a reservoir.

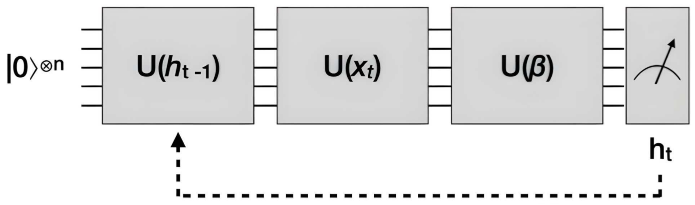

Figure 4 • Hybrid classical-quantum RC architecture.

In this implementation, each parameter h, x, and β can be vectors of different lengths. They can be encoded in a compact number of qubits, with the length of h set to 2^n states and β set to n qubits. The vector size of the input at a time step x can be of any length. The circuit schematic adopted can be the same for all these Unitary blocks, where the encoding is performed through rotating and entangling the values together using **RY** and **CX** gates, as illustrated in **Figure 5**, like the circuit in **Figure 1**, satisfying the non-linearity requirement.

Once the final qubit is reached, the encoding is continued in a zigzag manner to accommodate any length vector. This satisfies the compatibility of encoding discrete input elements as vectors.

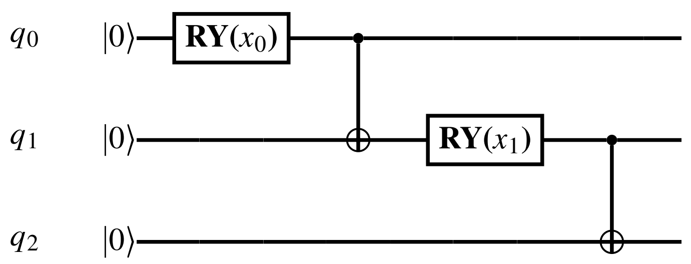

Figure 5 • An example of a simple circuit satisfying the non-linearity requirement.

The main advantage of this approach is that the feedback of measurements bypasses the back action problem mentioned earlier, and avoids the growing number of gates and circuit complexity seen with other QRC architectures. Hence, it is efficient to run them for longer sequence lengths and more batch sizes and iterations required for typical sequence modelling tasks. The fixed length of the circuit makes it modular and reconfigurable for training and generation purposes.

Our architecture solution is combined with the optimal reservoir design presented in [46] to replace the circuit block **U(**β**)** in **Figure 4** with a simple reservoir circuit consisting of an arbitrary assembly of **H** (Hadamard), **X**, and **CX** gates, as shown in **Figure 6**.

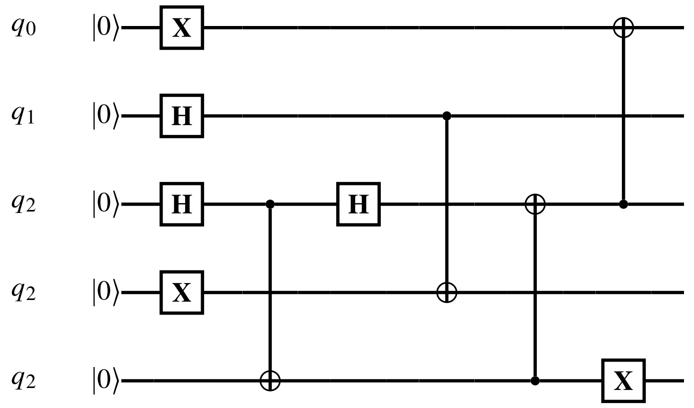

Figure 6 • An example of a randomly regenerated reservoir circuit.

3. Results

This section presents a demonstration of the system. First, we trained the model with the soundtrack theme of the American espionage television series *Mission: Impossible*, consisting of a total of 117 notes. Then, we let the model generate new tunes. The results from experimenting with different settings are discussed in Section 3.7. See also Section 4.

3.1. Data preparation

An excerpt from the beginning of the tune we used to train the system is shown in **Figure 7**. The tune is monophonic, consisting of a sequence of 117 notes. This produced a vocabulary of 23 unique events. Each element of the vocabulary Λ was encoded by considering its pitch and duration values together (Equation (11)).

They were indexed as unique events, or *classes*, ranging from 0 to 22. Hence, the output of the machine learning model would typically be a probability distribution over the 23 possible classes.

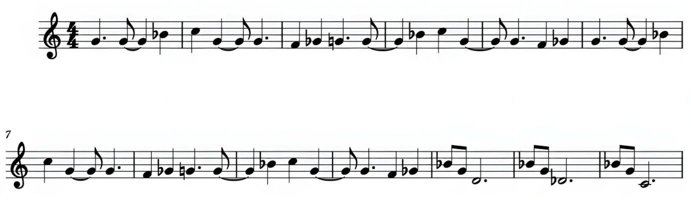

Figure 7 • Excerpt from the *Mission: Impossible* theme, notated by the authors. (The complete Λ is not shown.)

The training data were created with 100 samples of input–output pairs picked from random places in the tune, each with a sequence length of 32 and a washout period of 35%. We aimed to capture the long-term dependencies of the tune. The washout period of 35% made the system ignore the first 11 notes to make the response of the reservoir less dependent on its initial state.

$$\begin{aligned}\Lambda = [&(67,1.5),(67,1.5),(70,1.0),(72,1.0),(67,1.5),(67,1.5),\\&(65,1.0),(66,1.0),(67,1.5),(67,0.5),(67,1.0),(70,1.0),\\&(72,1.0),(67,1.0),(67,0.5),(67,1.5),(65,1.0),(66,1.0),\\&(67,1.5),(67,1.5),(70,1.0),(72,1.0),(67,1.5),(67,1.5),\\&(65,1.0),(66,1.0),(67,1.5),(67,0.5),(67,1.0),(70,1.0),\\&(72,1.0),(67,1.0),(67,0.5),(67,1.5),(65,1.0),(66,1.0),\\&(70,0.5),(67,0.5),(62,3.0),(70,0.5),(67,0.5),(61,3.0),\\&(70,0.5),(67,0.5),(60,3.0),(58,0.5),(60,1.0),(58,0.5),\\&(55,0.5),(66,3.0),(58,0.5),(55,0.5),(65,3.0),(58,0.5),\\&(55,0.5),(64,3.0),(63,0.5),(62,1.0),(75,0.5),(72,0.5),\\&(67,3.0),(75,0.5),(72,0.5),(66,3.0),(75,0.5),(72,0.5),\\&(65,3.0),(63,0.5),(65,1.0),(70,0.5),(67,0.5),(62,3.0),\\&(70,0.5),(67,0.5),(61,3.0),(70,0.5),(67,0.5),(60,3.0),\\&(58,0.5),(60,1.0),(67,1.5),(67,1.5),(70,1.0),(72,1.0),\\&(67,1.5),(67,1.5),(65,1.0),(66,1.0),(67,1.5),(67,0.5),\\&(67,1.0),(70,1.0),(72,1.0),(67,1.0),(67,0.5),(67,1.5),\\&(65,1.0),(66,1.0),(67,1.5),(67,1.5),(70,1.0),(72,1.0),\\&(67,1.5),(67,1.5),(65,1.0),(66,1.0),(67,1.5),(67,0.5),\\&(67,1.0),(70,1.0),(72,1.0),(67,1.0),(67,0.5),(67,1.5),\\&(65,1.0),(66,1.0),(67,4.0)]\end{aligned} \tag{11}$$

3.2. Input layer

The representation of input data plays a significant role in the performance of neural network architectures, and RC is no exception. For high-level information, such as natural language and music, it is a common practice to map the discrete elements onto a multi-dimensional vector that ensures enough separability, so a neural network can effectively process it [54]. These continuous vector representations can be used to capture relationships between classes and help a machine learning model with better generalization.

For our demonstration, the mono-dimensional music sequence is projected into a multi-dimensional vector space using a fixed random input layer, which the quantum reservoir will follow. The input layer weights W_{in} are randomly generated using a uniform distribution as shown in Equation (12).

$$W_{in} = [w_1, w_2, w_3, ..., w_N]^T; \qquad w_t \in [-\sigma, \sigma] \tag{12}$$

In Equation (12), N represents the dimension of the input layer and σ is the input-scale coefficient [55]. To ensure separation of the music events, we worked with N set to an arbitrarily high value equal to 32 and σ equal to 0.5. Hence, the input weights contain 32 random values ranging from -0.5 to 0.5. If the input note at any given time t is given as x_t, then the projection at the input will be a scalar multiplication of $W_{in} \times x_t$.

A visualization of the input layer projection of each unique event in the tune is shown in **Figure 8**. The visualization is made possible using Principal Component Analysis (PCA) to reduce the values from a 32-dimensional space to a 3-dimensional space while preserving maximum variance [56].

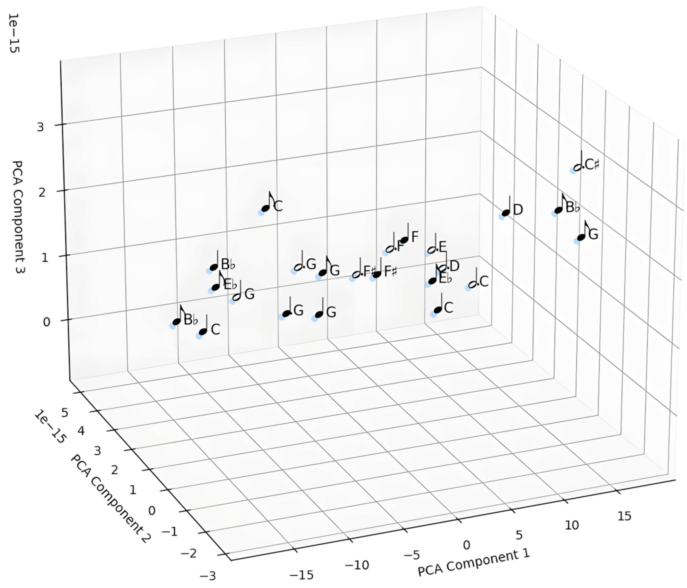

Figure 8 • Example projection of musical notes onto a 32-dimensional space, but reduced to a 3-dimensional representation.

Because of the random projection, and how unique events were represented (as an aggregate of pitch and duration), it can be noted that events with the same pitch, but different durations, are spread out. In a potential use of the *transfer learning* technique, where the input layer weights trained from a larger neural network can be borrowed, similar notes would be clustered together. This would be an efficient strategy for Natural Language Processing (NLP) tasks as proven with other architectures like Transformer-based models [57]. But it is not a matter of concern in the case of this demonstration.

3.3. Circuit preparation

After passing through the input layer, each unique event was transformed into a unique vector, which was used to parameterise and prepare the state of the quantum circuit set to five qubits. The input vector was used to parameterise the second Unitary block in **Figure 4**.

The first Unitary block was parameterised using a random initial state following the Dirichlet distribution [58], which ensured that the probability distribution added to 1. Then, this was updated sequentially with the measurement outcomes h (**Figure 4** and **Figure 9**) of the previous time-step. The full circuit layout per time-step involving the parameterised blocks of state and input followed by the fixed reservoir circuit is depicted in **Figure 9**.

As seen in **Figure 9**, keeping the reservoir circuit minimal can also lead to keeping the input-driven dynamics more prominent than the internal dynamics of the reservoir.

3.4. Measurements

We ran the circuits on an IQM machine sporting five qubits. We also ran experiments on the Qiskit simulator [59]. As we were using only five qubits here, we did not notice much difference between using the real machine and the simulator.

We performed 4000 shots per circuit run. Since quantum computing systems are inherently probabilistic, a single execution of a quantum circuit does not yield a definitive result. Instead, it provides a probabilistic outcome. To obtain useful information, multiple measurements, or shots, are often performed. Each shot involves running the same quantum circuit and making measurements, which allows for statistical analysis of the results. For example, if one is interested in measuring the probability distribution of a quantum state, one would run the circuit multiple times, each time performing a measurement, and then tallying the results over all the shots to estimate the probabilities of different outcomes.

Executing the circuits at each time-step created a quantum state dependent on the current unique event and the fading memory of previous unique events. Since the quantum state contributed by the reservoir circuit was fixed, the input-driven dynamics arising from the varying parameterisation circuits act upon it to create new states, sequentially distorting the initial state. This is analogous to perturbing the surface of water by sequentially applying a mechanical disturbance. An equivalent visualisation can be observed in QRC using the plot of measurement outcomes from the 4000 shots at six time steps, where each event perturbs the quantum state sequentially, as shown in **Figure 10**.

3.5. Training

The following approach was used during the training stage to ensure process separation. In the case of the present example, the QRC model is not directly connected to the output readout layer. At each time step, an event (represented as a token index) passes through the input and reservoir layers, generating 32 measured features. Before the next time step, the internal state of the reservoir is updated using feedback. This process repeats for a given sequence length, e.g., 8, resulting in an array with a shape equal to [8, 32].

This procedure is then repeated for multiple samples, producing an array of the following shape: [number of samples, sequence length, number of measured features]. This array is used as input to the linear output layer separately. The output layer is configured to work with variable sequence lengths.

During the generation/inference stage, QRC is established with a direct connection between the input and output readout layer. The trained output layer is reconfigured to operate with a sequence length equal to 1, allowing events to be generated one at a time. In this stage, two types of feedback occur: internal feedback, which updates the reservoir's memory, and external feedback, which passes the current output as input for the next step.

Thus, the readout layer at the output maps the measured states to one of the 23 unique events in the vocabulary. This was the only optimisation performed for the demonstration. The output layer used a linear activation function. Essentially, it performed a multi-class classification task to predict the next unique event.

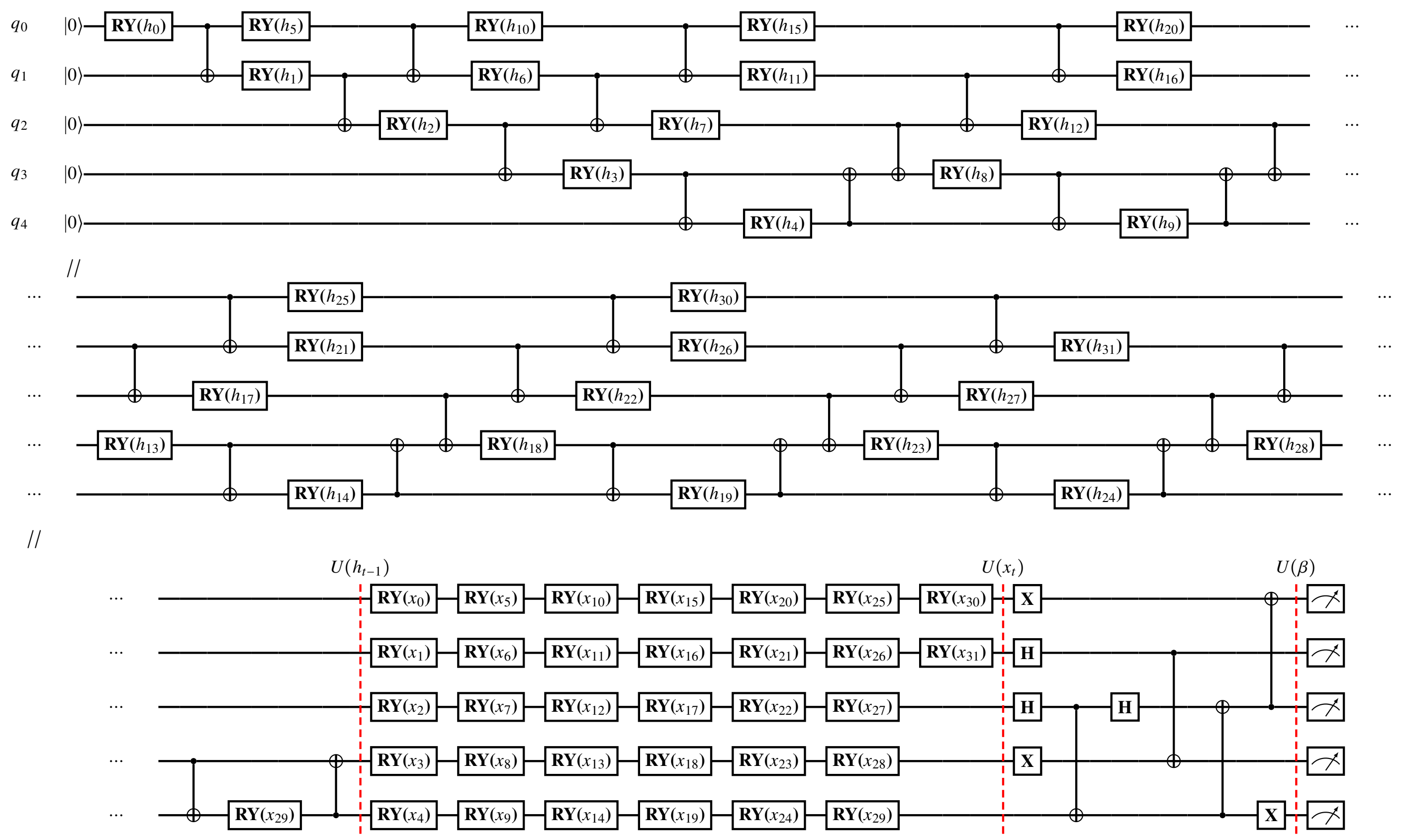

Figure 9 • An example of the complete circuit layout at a time step. The dashed line delineates the three stages shown in **Figure 4**.

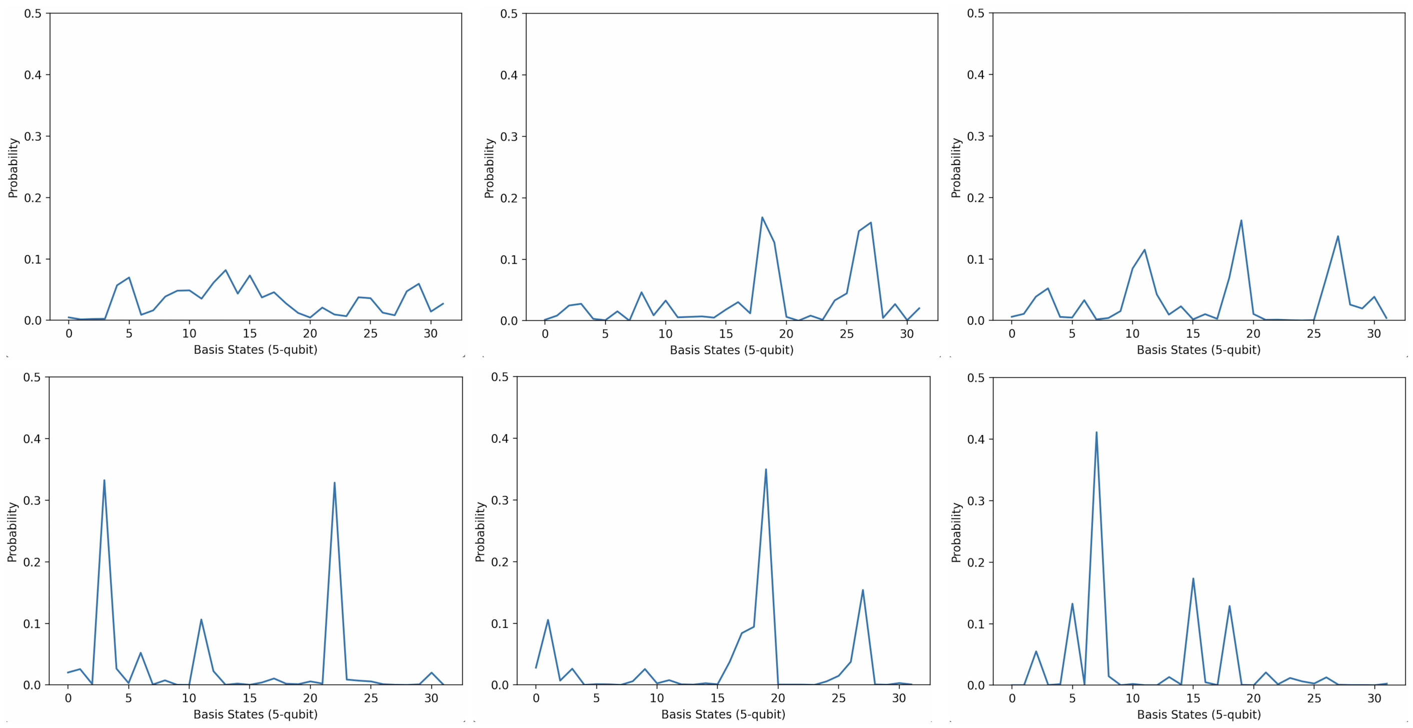

Figure 10 • Perturbations of quantum states in response to sequential musical events, starting at $t = 0$ at the top left and continuing from the left to the right, towards the bottom plot, with $t = 5$ at the bottom right.

For 100 samples each with a sequence length of 32, a total of 3200 measurements were required. This could be configured optimally based on the required task and execution time of quantum hardware. Taking the washout period into account, the rest of the measurement was fed into the output layer and a cross-categorical loss function was used to compare the predicted event with the target event. With a batch size equal to 2 and a learning rate of 0.005, the Adam optimiser [60] was utilized for training throughout 50 epochs, representing a complete iteration through the dataset. The learning converged within a few epochs, indicating that the model had linearly fit the measured quantum states with music events as shown in **Figure 11**.

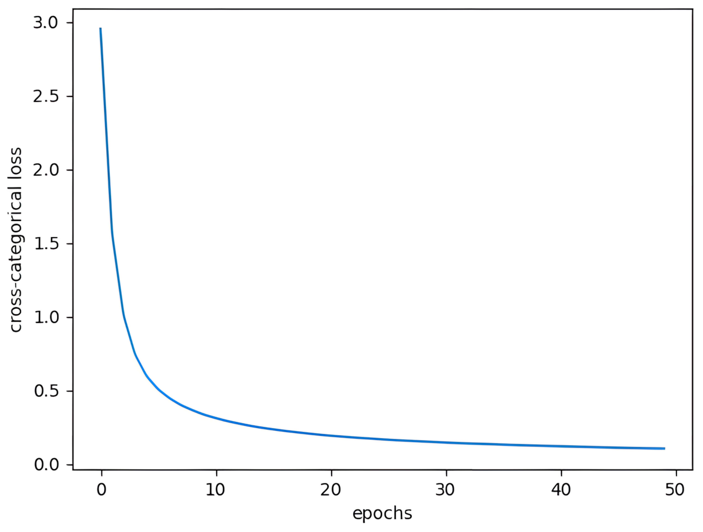

Figure 11 • Optimisation of loss function.

Here, the quantum circuit measurements were collected independently and used to fit the linear output layer. In such a case, the training is much faster than a traditional RNN for sequence modelling. Once the readout weights are learnt, the whole model can be fixed and used for autonomous music generation of any length. The remainder of this paper discusses the results of such a generative process.

3.6. Generated music

The transition matrix of the whole length of the *Mission: Impossible* tune used for the training is shown in **Figure 12**. It displays the overall frequency of each unique event following another unique event in the form of a heat map, independent of the temporal order.

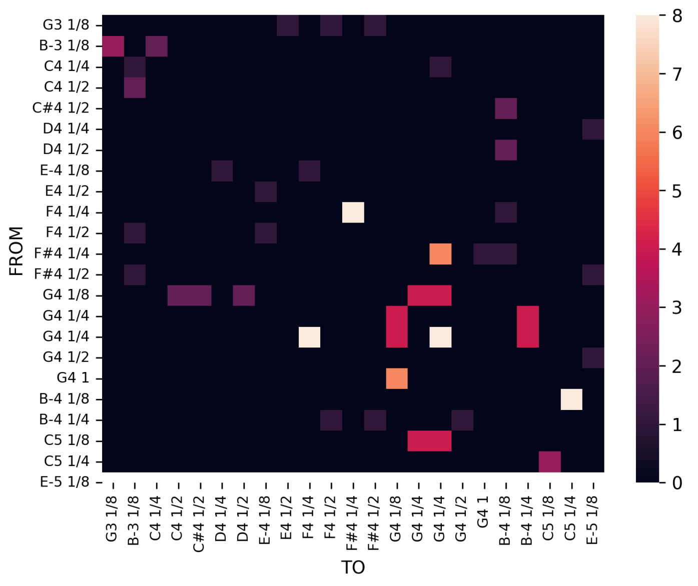

Figure 12 • A transition matrix of the *Mission: Impossible* theme.

Then, we set the system to generate a new tune with a total of 500 events to examine how the transition matrix develops and settles over time. The result from a fully trained model with a low cross-categorical loss equal to 0.26 is displayed in **Figure 13**.

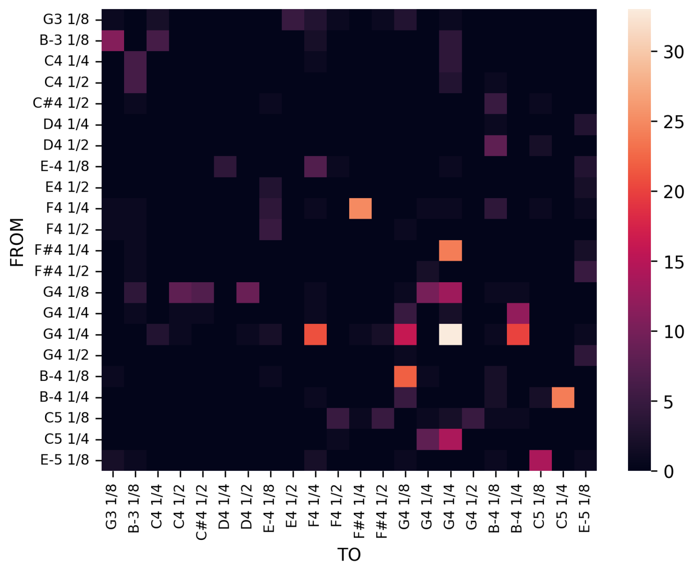

Figure 13 • Transition matrix of recreated tune.

Comparing **Figure 12** and **Figure 13**, it can be observed that the pattern of the musical piece is learnt and recreated by the quantum reservoir with a simple linear readout. The difference in frequency of the heat map is expected as the original tune is much shorter than the generated one. An excerpt from the resulting tune is shown in **Figure 14**.

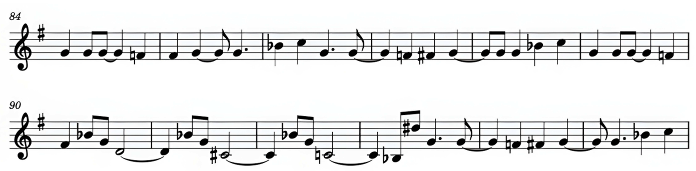

Figure 14 • An excerpt from a generated tune.

3.7. A creativity potentiometer

The tune in **Figure 14** resulted from an optimal low cross-categorical loss equal to 0.26. This tune clearly resembles the original *Mission: Impossible* tune (**Figure 7**).

However, the cross-categorical loss allows a straightforward way of exploring the system creatively. One may not necessarily wish the system to create tunes that imitate the original style faithfully. Higher values will result in tunes bearing little resemblance to the original.

We generated different tunes with different loss values, which were captured by simply stopping the training earlier to observe the respective transition matrices.

The simplest untrained readout layer with a high loss equal to 3.13 is shown in **Figure 15**. In this case, the spread of the probabilities is wider in this transition matrix than in the matrix in **Figure 13**, both of which were obtained by generating 500 notes. An excerpt from a tune generated with this matrix is given in **Figure 16**.

The matrix developed clearer patterns of transition as it is optimised with increasingly lower loss values equal to 2.07 and 0.84, as shown in **Figure 17** and **Figure 18**. Excerpts from the tunes produced with these matrices are shown in **Figure 19** and **Figure 20**, respectively.

The experiments above hint at one of the possible ways of controlling the inventiveness of the system. Think of a ‘creativity potentiometer’, which can control the degree to which the system should follow the rules learnt by the system to compose new tunes. We can devise many such creativity potentiometers associated with parameters like the number of shots and qubit rotations.

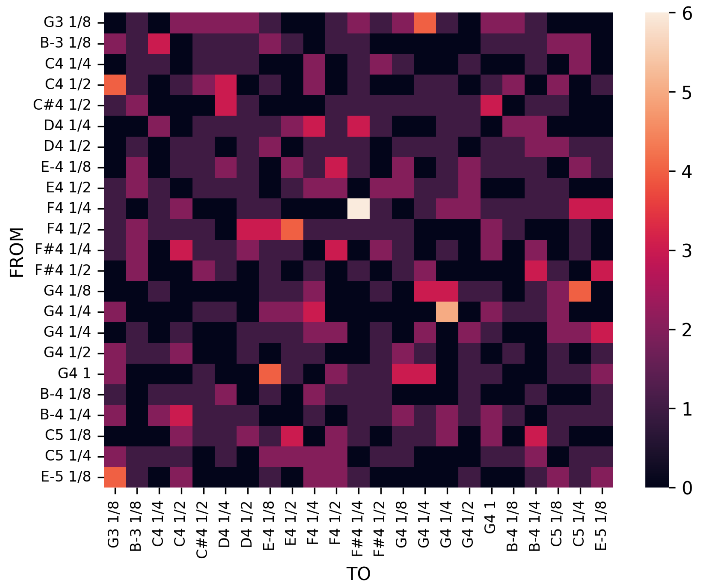

Figure 15 • Transition matrix generated with cross-categorical loss equal to 3.13.

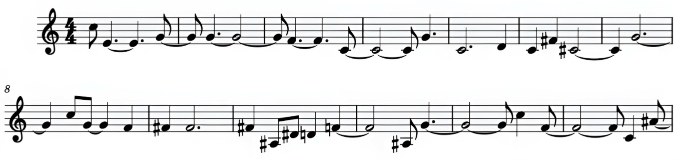

Figure 16 • An excerpt from a recreated tune with a cross-categorical loss equal to 3.13.

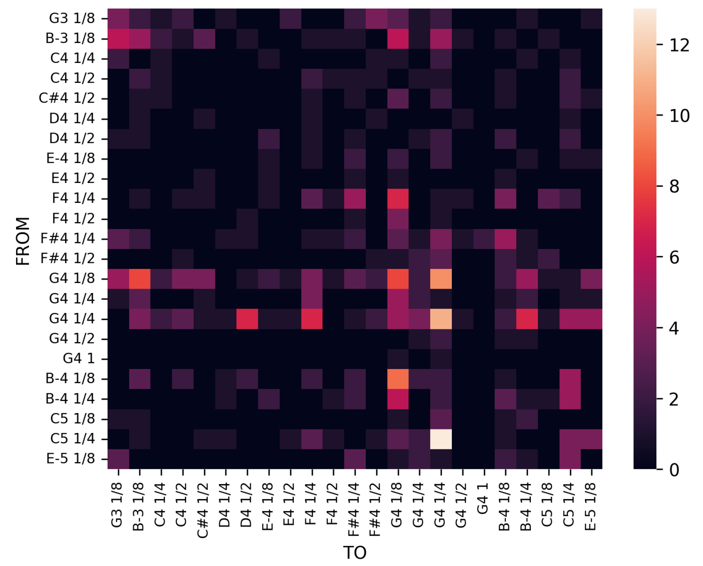

Figure 17 • Transition matrix generated with cross-categorical loss equal to 2.07.

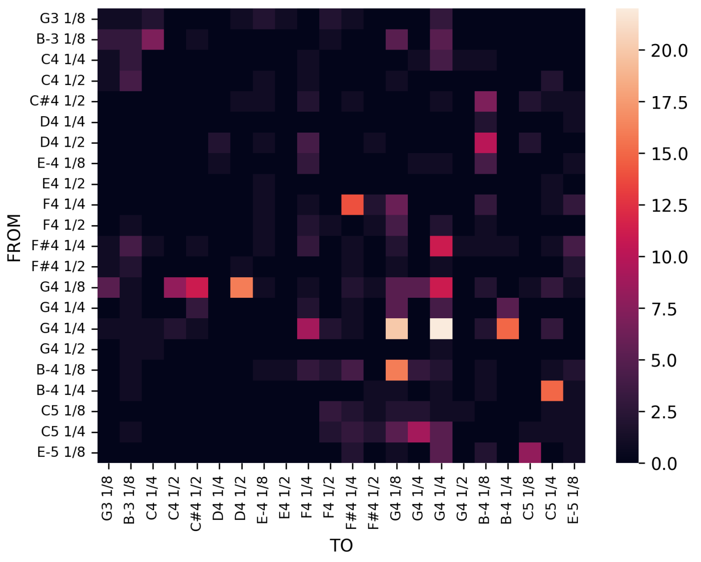

Figure 18 • Transition matrix generated with cross-categorical loss equal to 0.84.

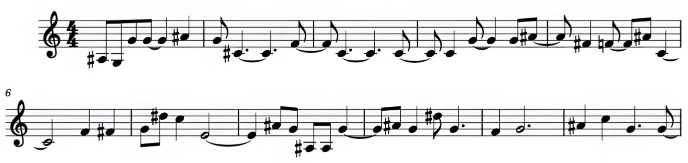

Figure 19 • An excerpt from a recreated tune with a cross-categorical loss equal to 2.07.

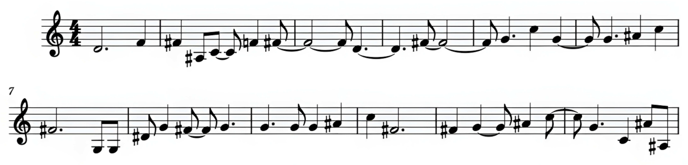

Figure 20 • An excerpt from a recreated tune with a cross-categorical loss equal to 0.84.

3.7.1. Versatility

The abovementioned experiments also highlight the reusability and versatility of the quantum reservoir approach over other classic machine learning techniques.

The system needs to run only once to collect the measurements. The measured states can then be re-used multiple times with such readout layers each having its unique profile, which is simple to train and not limited to mapping just one feature. Hence, significant computational resources are saved compared to classic machine learning methods. This is analogous to the concept of ‘transfer learning’ in machine learning, which re-uses the input layer. In reservoir computing, the core computational layer can be reused. This is particularly relevant in cases where the usage of quantum hardware incurs financial costs or is technically limited due to time-sharing with other users, or both.

3.7.2. Interactivity

Considering the fading memory of a reservoir, it was observed that a model with a fully trained readout can re-create the original trained music if the initial notes fed into the model are a copy of the original music. Conversely, a different sequence of initial notes affects the fading memory and the generation of subsequent notes.

We conducted an experiment whereby the order of initial events was shuffled using seed values before passing them to the same model. Two selected examples of results are shown in **Figure 21** and **Figure 22**. These matrices hold unique patterns, which embody musical structure. The most frequent transitions appear at different places than on a matrix using the original music. An important aspect of this is the possibility of determinism and control over music generation.

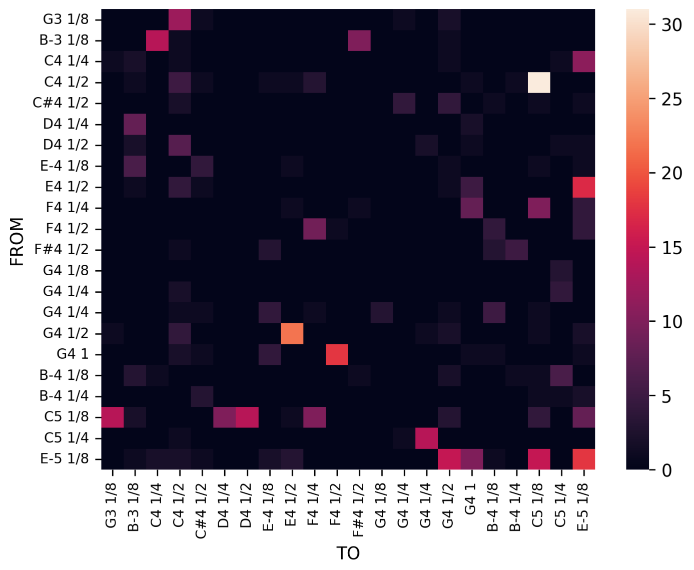

Figure 21 • Transition matrix produced with random seed value = 0.

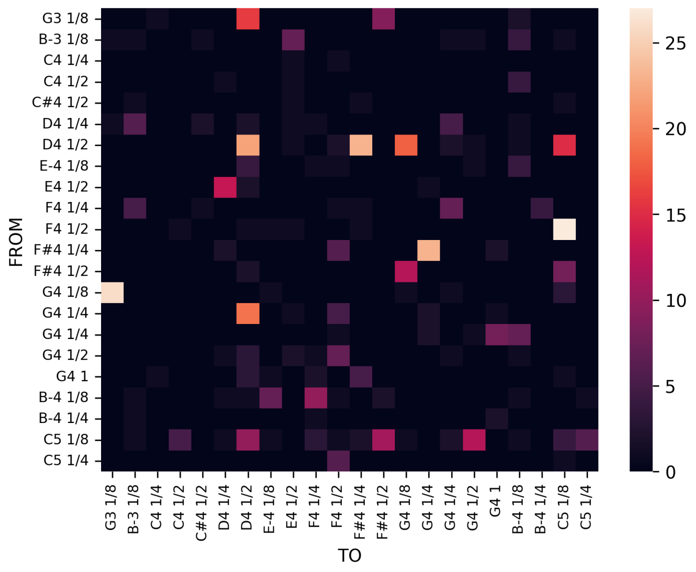

Figure 22 • Transition matrix produced with random seed value = 42.

Re-running the experiment with the same seed value results in the same transition matrix over time. This indicates that all different combinations of initial events of different lengths can trigger the model to generate unique music with clear hierarchies and low variability, dependent on that specific initial sequence along with the training data. Out of other possible parameters that control the music, the generation in direct response to user input can be a desirable feature for musical interactivity.

Figure 23 and **Figure 24** give excerpts of the tunes generated with seed values equal to 0 and 42, respectively.

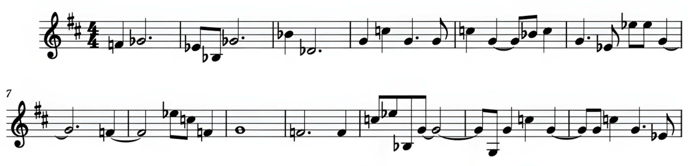

Figure 23 • An excerpt from a tune generated with the matrix in **Figure 1**.

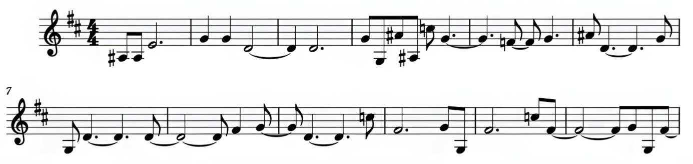

Figure 24 • An excerpt from a tune generated with the matrix in **Figure 22**.

4. Discussion

4.1. Comparison with long short-term memory network

In this section, we compare our QRC method with the Long Short-Term Memory (LSTM) network method. The latter is a Deep Learning type of method, which is often the first choice for learning sequences [61]. It is a special type of RNN that bears close resemblance to the QRC method [26], as discussed in Section 1.3. Both address the *soi-disant* 'vanishing gradient problem'. This problem takes place when the gradients used to update the weights of a neural network become increasingly small as they are propagated back through many time steps. This makes it difficult to train the network effectively.

For the comparison, we trained the systems with the same tune used in the demonstration presented in Section 3. The vocabulary is a sequence of 117 MIDI notes, comprising 23 unique events. The circuit for the QRC is identical to the one discussed in Section 3.3. The preparation of data is also identical.

The task is to build a generative music model whereby given an initial tune, the model must generate a continuation in the style of the *Mission: Impossible* tune.

4.2. Parameters

The following parameters were set for both systems, QRC and LSTM.

Training Data:

- `Amount of samples` $= 16$;
- `Sequence length` $= 8$;
- `Input dimension` $= 32$;
- `Number of units` $= 5$;
- `Batch size` $= 2$;
- `Epoch` $= 50$;
- `Learning rate` $= 0.005$.

The `amount of samples` parameter specifies the number of data points for the training stage. That is the number of input and output sequence pairs extracted from the original sequence. In this case, the output is the continuation of an input sequence that is to be learnt. The `sequence length` parameter is the length of each input and output sequence used for training.

The parameter `input dimension` specifies the size of the input embedding vector and the `number of units` corresponds to the size of the hidden layer. This would be the equivalent of the number of neurons in a typical artificial neural network hidden layer.

The `batch size` parameter is the number of sample batches considered at one iteration of training. An `epoch` is one complete cycle through the entire training dataset and the `Learning rate` is a hyperparameter, which controls the step size during gradient descent in the back-propagation process [52, 61].

The training was performed with TensorFlow https://www.tensorflow.org/. For both models, Adam was used as the optimiser (https://ml-explained.com/blog/adam-explained (Adaptive Moment Estimation)), and the Sparse Categorical Cross-entropy loss function (https://www.shiksha.com/online-courses/articles/cross-entropy-loss-function/) was applied.

4.3. Analyses

The efficiency of the methods was assessed through the number of trainable parameters in the system. Their performance was measured according to the convergence of their loss function (Sparse Categorical Cross-entropy).

For the LSTM networks, the `number of units` corresponds to the total number of artificial neurons in the hidden layer. However, only five neurons do not form a sufficient feature space for the task at hand. Conversely, only five qubits for the QRC provide a good feature space, with 32 basis states that can be measured.

Whereas QRC yields 759 trainable parameters, corresponding to the output layer, the LSTM network yields 1634 trainable parameters, corresponding to input, hidden, and output layers. In this sense, QRC is more economical than LSTM.

As for the convergence of the loss function, the final loss value for LSTM after training was equal to 1.20. In contrast, the final loss value for QRC was equal to 0.156, a considerable difference. The plot in **Figure 25** shows the normalised loss curves. QRC sports a faster convergence curve than the LSTM network.

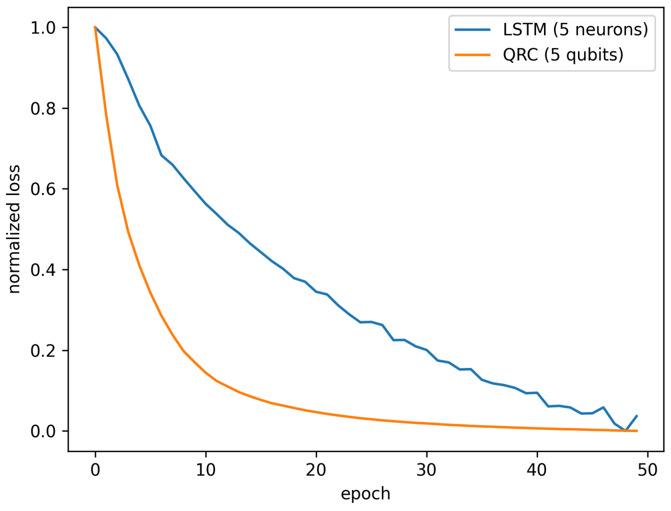

Figure 25 • A QRC system shows a faster convergence curve than the LSTM one.

In a second experiment, we extended the `number of units` for the LSTM network to 32. The QRC remained equal to five. This increased the number of LSTM-trainable parameters to 9815. In this case, the LSTM loss value after training improved to 0.22. But still, with only five qubits, QRC sports comparable performance while using only 7.72% of the number of trainable parameters of that of LSTM. This is a notable benefit. The plot in **Figure 26** shows the normalised loss curves. QRC continued to show a faster convergence curve than the LSTM network.

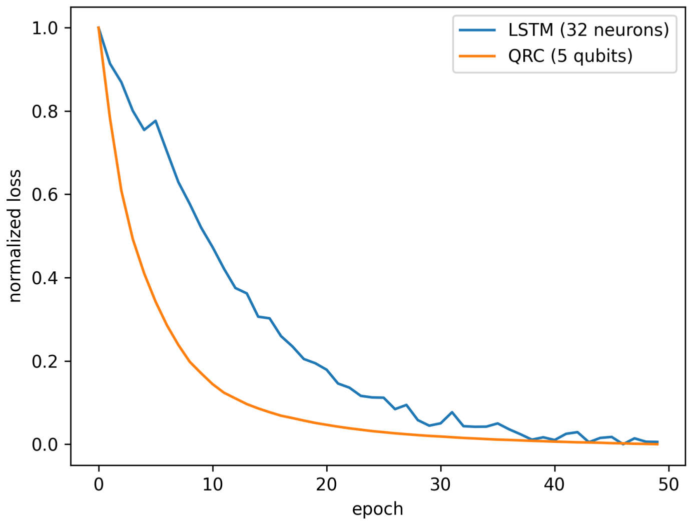

Figure 26 • A QRC system with only five qubits shows a faster convergence curve than the LSTM model with 32 neurons.

5. Conclusions

We proposed Quantum Reservoir Computing (QRC) as an addition to the growing arsenal of machine learning technologies for the development of computer-aided music composition (CAMC) systems.

The takeaways from this paper are as follows. First, QRC works well on currently available NISQ machines. QRC does not rely on quantum algorithms doomed by qubit decoherence, or noise, such as Grover's-like search algorithms [62]. Rather, QRC harnesses quantum mechanical dynamics to function as a fixed neural network layer [63]. However, more research is needed to establish the extent to which QRC is immune to noise and whether or not QRC would still perform other quantum machine learning methods on FTQC technology.

Second, QRC has demonstrable advantages in scalability and flexibility over classical RNN approaches, as discussed in Section 3.7.1. The number of trainable parameters is reduced considerably, as they only correspond to the output layer [64]. The modest system presented above mapped 32 quantum states as input to the probability distribution of a vocabulary of 23 distinct elements. The total trainable parameters in this case was 759. An equivalent output dimension of 32 from the hidden layer of a classical RNN would require nearly 10,000 trainable parameters (see Section 4). For better performance with small-scale tasks, just over 100 RNN cells can amount to 1,00,000 parameters. With QRC, we only needed five qubits for this without compromising the performance. Thus, QRC provides an efficient alternative in a resource-intensive generative AI landscape by largely reducing the number of trainable parameters.

Admittedly, the music example discussed in this paper is simple. We purposely decided to use a simple example for didactic reasons, as the objective of this paper is to introduce our QRC method. The unique events were encoded at the note level as pitches and durations (Equation (11)). Moreover, we dealt with monophonic music only. Yet, music is often polyphonic, involving more than one instrument or track. However, this is a problem of music representation and not machine learning per se.

Currently, we are developing a method for encoding unique events in terms of horizontal and vertical aggregates of musical notes, such as melodies, riffs, and chords. In addition to pitches and durations, the new method will include information such as loudness and instrumentation. This new representation will afford large vocabularies. This will require output layers of much higher dimensions. The QRC will need more qubits for this. Nevertheless, as we head towards higher counts of fault-tolerant logical qubits soon, we are confident that we will be able to scale up our system to process more complex music representations in larger amounts when these machines are available. Moreover, we have been conducting preliminary experiments where we observed that increasing the number of qubits can lead to an increase in memory capacity, that is, how much the model can keep track of past events.

One important caveat of QRC is that it may require the optimisation of several factors such as circuit design, sequence length, washout period, learning rate of the optimiser, and batch size used in training. We have not fully experimented with those parameters [65]. This is ongoing research.

Acknowledgments

The authors thank Karl Jansen at Desy, Zeuthen, Germany, for enabling access to IBM Quantum hardware.

Funding

This research received no external funding.

Author contributions

Conceptualization, E.R.M.; methodology, E.R.M; software, H.V.S.; validation, E.R.M. and H.V.S.; formal analysis, E.R.M. and H.V.S.; investigation, E.R.M. and H.V.S.; resources, E.R.M.; data curation, H.V.S.; writing—original draft preparation, E.R.M. and H.V.S.; writing—review and editing, E.R.M.; visualization, H.V.S..; supervision, E.R.M.; project administration, E.R.M. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflicts of interests.

Data availability statement

Data supporting these findings are available within the article, at https://doi.org/10.20935/AcadQuant7699, or upon request.

Institutional review board statement

Not applicable.

Informed consent statement

Not applicable.

Additional information

Received: 2025-01-29

Accepted: 2025-04-28

Published: 2025-05-26

Academia Quantum papers should be cited as *Academia Quantum 2025*, ISSN 3064-979X, https://doi.org/10.20935/AcadQuant7699. The journal's official abbreviation is *Acad. Quant.*

Publisher's note

Copyright

References

1. Akkerman M. From computer-aided to automatic compositions? A closer look at generative music systems. Computational and Digital Approaches to Music Scholarship. Online publication. 2024 [cited 2025 Mar 31]. Available from: https://codamus.pubpub.org/pub/2023-akkermann/release/2
2. Bouche D, Nika J, Chechile A, Bresson J. Computer-aided composition of musical processes. J New Music Res. 2017;46(1):1–15. doi:10.1080/09298215.2016.1230136
3. Miranda ER. Composing Music with Computers. Oxford: Elsevier, Focal Press; 2002. ISBN 0240515676
4. Sandred Ö. The Musical Fundamentals of Computer Assisted Composition. Winnipeg (MB): Audiospective Media; 2014. ISBN 9781775000013.
5. OpenMusic GitHub Repository. [cited on 2024 Dec 26]. Available from: https://github.com/openmusic-project/

6. Opusmodus Webpage. [cited on 2024 Dec 26]. Available from: https://opusmodus.com/index.html

7. Ban in a Box Website. [cited on 2024 Dec 26]. Available from: https://www.pgmusic.com/

8. Cope D. Experiments in Musical Intelligence. 2nd edition. Middleton (WI): A-R Editions; 2014. ISBN 9780895793379.

9. Miranda ER, editor. Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity. Cham: Springer Nature; 2021. doi: 10.1007/978-3-030-72116-9

10. Miranda ER, editor. Readings in Music and Artificial Intelligence. London : Routlegde, Taylor and Francis Group; 2000. ISBN 9789057550942.

11. Balaban M, Ebcioglu K, Laske OE, editors. Understanding Music with AI Perspectives on Music Cognition. Cambridge (MA): The MIT Press; 1992.

12. Roads C. The Computer Music Tutorial. 2nd ed. Cambridge (MA): The MIT Press; 2023. ISBN 9780262044912.

13. Lerdahl F, Jackendoff R. A Generative Theory of Tonal Music. Cambridge (MA): The MIT Press; 1983. ISBN 9780262620499.

14. Nierhaus G. Algorithmic Composition: Paradigms of Automated Music Generation. Cham: Springer; 2008. ISBN 9783211755396.

15. Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge (MA): The MIT Press; 2016. ISBN 9780262035613.

16. Tyagi AK, Abraham A. Recurrent Neural Networks: Concepts and Applications. Boca Raton (FL): CRC Press; 2023. ISBN: 9781032081649.

17. Kamath U, Graham K, Emara W. Transformers for Machine Learning: A Deep Dive. Boca Raton (FL): CRC Press; 2022. ISBN: 9780367767341.

18. OpenAI MusiNet Website. [cited on 2024 Mar 28]. Available from: https://openai.com/research/musenet

19. Magenta Music Transformer Website. [cited on 2024 Mar 28]. Available from: https://magenta.tensorflow.org/music-transformer

20. OpenAI Blog. Planning for AGI and beyond. [cited on 2024 Dec 28]. Available from: https://openai.com/index/planning-for-agi-and-beyond/

21. Miranda ER, editor. Quantum Computer Music. Cham: Springer Nature; 2022. ISBN 9783031139086.

22. Cho A. No room for error. Science. 2020;370(6523):1420–1. doi:10.1126/science.abd7332.

23. Brooks M. Quantum computing is taking on its biggest challenge: Noise. MIT Technology Review, The innovation issue January/February 2024. 2024 [cited on 2024 Mar 30]. Available from: https://www.technologyreview.com/2024/01/04/1084783/quantum-computing-noise-google-ibm-microsoft/

24. Preskill J. Quantum Computing in the NISQ era and beyond. arXiv. 2018. preprint arXiv: 1801.00862. doi: 10.48550/arXiv.1801.00862

25. Domingo L, Carlo G, Borondo F. Taking advantage of noise in quantum reservoir computing. arXiv. 2023. preprint arXiv: 2301.06814. doi: 10.48550/arXiv.2301.06814

26. Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Comput. 1997;9(8):1735–1780. doi: 10.1162/neco.1997.9.8.1735

27. Cucchi M, Abreu S, Ciccone G, Brunner D, Kleemann H. Hands-on reservoir computing: A tutorial for practical imple-mentation. Neuromorphic Comput Eng. 2022;2(3):032002. doi: 10.1088/2634-4386/ac7db7

28. Dominey PF. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol Cybern. 1995; 73(3):265–74. doi: 10.1007/BF00201428

29. Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning with Neural Networks. arXiv. 2014. preprint arXiv: 1409.3215. doi: 10.48550/arXiv.1409.3215

30. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv. 2016. preprint arXiv: 1603.04467. doi: 10.48550/arXiv.1603.04467

31. Miranda ER, Shaji H. Developing Quantum Reservoir Computing as Machine Learning of Music. Proceedings of the 2nd International Symposium on Quantum Computing and Musical Creativity (2nd ISQCMC); 2023 Oct 5–6; Berlin, Germany; 2024. doi: 10.5281/zenodo.10206712

32. Jaeger H. Foreword. In: Nakajima K, Fischer I, editors. Reservoir Computing. Natural Computing Series. Singapore: Springer; 2021. doi: 10.1007/978-981-13-1687-6

33. Nakajima K, Fischer I, editors. Reservoir Computing: Theory, Physical Implementations, and Applications. Natu-ral Computing Series. Singapore: Springer; 2021. doi: 10.1007/978-981-13-1687-6

34. Tanaka G, Yamane T, Héroux JB, Nakane R, Kanazawa N, Takeda S, et al. Recent advances in physical reservoir computing: A review. Neural Netw. 2019;115:100–23. doi: 10.1016/j.neunet.2019.03.005

35. Schuman CD, Kulkarni SR, Parsa M, Mitchell JP, Date P, Kay B. Opportunities for neuromorphic computing algorithms and applications. Nat Comput Sci. 2022;2:10–9. doi: 10.1038/s43588-021-00184-y

36. Nielsen MA, Chuang IL. Quantum Computation and Quantum Information. 10th Anniversary ed. Cambridge: Cambridge University Press; 2010. doi: 10.1017/CBO9780511976667

37. Cindrak S, Donvil B, Lüdge K, Jaurigue L. Enhancing the performance of quantum reservoir computing and solving the time-complexity problem by artificial memory restriction. Phys Rev Res. 2024;6:013051. doi: 10.1103/PhysRevRe-search.6.013051

38. Fujii K, Nakajima K. Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning. Phys Rev Appl. 2017;8(2):024030. doi: 10.1103/physrevapplied.8.024030

39. Zeguendry A, Jarir Z, Quafafou M. Quantum Machine Learn-ing: A Review and Case Studies. Entropy. 2023;25(2):287. doi: 10.3390/e25020287

40. Bausch J. Recurrent Quantum Neural Networks. arXiv 2020. preprint arXiv: 2006.14619v1. doi: 10.48550/arXiv.2006.14619

41. McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H. Barren plateaus in quantum neural network train-ing landscapes. textitNat Commun. 2018;9(1):4812. doi: 10.1038/s41467-018-07090-4

42. Verstraete F, Wolf MM, Cirac JI. Quantum computation and quantum-state engineering driven by dissipation. Nat Phys. 2009;5(9):633–6. doi: 10.1038/nphys1342

43. Suzuki Y, Gao Q, Pradel KC, Yasuoka K, Yamamoto N. Natural Quantum Reservoir Computing for Tempo-ral Information Processing. Sci Rep. 2022;12(1):1353. doi: 10.1038/s41598-022-05061-w

44. Bravo RA, Najafi K, Gao X, Yelin SF. Quantum reservoir computing using arrays of Rydberg atoms. arXiv 2021. preprint arXiv: 2111.10956. doi: 10.48550/arXiv.2111.10956

45. Chen J, Nurdin HI, Yamamoto N. Temporal information processing on noisy quantum computers. Phys Rev Appl. 2020;14(2):024065. doi: 10.1103/PhysRevApplied.14.024065

46. Domingo L, Carlo G, Borondo F. Optimal quantum reservoir computing for the noisy intermediate-scale quantum era. Phys Rev. 2022;106(4):L043301. doi: 10.1103/physreve.106.l043301

47. Singer W. The Cerebral Cortex: A Delay-Coupled Recur-rent Oscillator Network?. In: Nakajima K, Fischer I, editors. Reservoir Computing. Natural Computing Series. Singapore: Springer; 2021. doi: 10.1007/978-981-13-1687-6_1

48. Schuld M. Supervised quantum machine learning models are kernel methods. arXiv. 2021. preprint arXiv: 2101.11020. doi: 10.48550/arXiv.2101.11020

49. Havlicek V, Córcoles A, Temme K, Harrow A, Kandala A, Chow J, et al. Supervised learning with quantum enhanced feature spaces. arXiv. 2018. preprint arXiv: 1804.11326. doi: 10.48550/arXiv.1804.11326

50. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics; 2021 Nov 5; Fort Lauderdale, FL, USA; 2011. pp. 315–23.

51. Cucchi M, Gruener C, Petrauskas L, Steiner P, Tseng H, Fischer A, et al. Reservoir computing with biocompat-ible organic electrochemical networks for brain-inspired biosignal classification. Sci Adv. 2021;7(34):eabh0693. doi: 10.1126/sciadv.abh0693

52. Mujal P, Martínez-Peña R, Giorgi GL, Soriano MC, Zambrini R. Time-series quantum reservoir computing with weak and projective measurements. npj Quantum Inf. 2023;9(1):1–10. doi: 10.1038/s41534-023-00682-z

53. Pfeffer P, Heyder F, Schumacher J. Hybrid Quantum-Classical Reservoir Computing of Thermal Convection Flow. Phys Rev Res. 2022;4(3): 033176. doi: 10.1103/physrevresearch.4.033176

54. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. arXiv. 2013. preprint arXiv: 1301.3781. doi: 10.48550/arXiv.1301.3781

55. Nakajima K. RC-Tutorial 2023. Physical Intelligence Lab. The University of Tokyo. 2003 [cited on 2024 Apr 02]. Available from: https://colab.research.google.com/drive/1SRLLu7jcnR6Fi1sbQh_IEdF2JAwHMTQg?usp=sharing

56. Jolliffe IT, Cadima J. Principal Component Analysis: A Review and Recent Developments. Philos Trans R Soc Math Phys Eng Sci. 2016;374(2065):20150202. doi: 10.1098/rsta.2015.0202

57. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv. 2019. preprint arXiv: 1810.04805. doi: 10.48550/arXiv.1810.04805

58. Sinharay S. Continuous Probability Distributions. In: Peterson P, Baker E, McGaw B, editors. International Encyclopedia of Education. Amsterdam: Elsevier; 2010. doi: 10.1016/B978-0-08-044894-7.01720-6.

59. IBM. Qiskit Aer 0.13.3. [cited on 2024 Mar 27]. Available from: https://qiskit.github.io/qiskit-aer/tutorials/1_aer_provider.html#Introduction

60. Brownlee J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Machine Learning Mastery. 2021 [cited on 2024 Mar 26]. Available from: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

61. Tabejamaat M, Mohammadzade H, Negin F, Bremond F. EEG classification with limited data: A deep clustering approach. Pattern Recognit. 2024;157(C):110934. doi: 10.1016/j.patcog.2024.110934

62. Szablowski PJ. Understanding the mathematics of Grover's algorithm. Quantum Inf Process. 2021;20:191. doi: 10.1007/s11128-021-03125-w

63. Sornsaeng A, Dangniam N, Chotibut T. Quantum Next Generation Reservoir Computing: An Efficient Quantum Algorithm for Forecasting Quantum Dynamics. Quantum Mach Intell. 2023;6:57. doi: 10.1007/s42484-024-00188-7

64. Xia W, Zou J, Qiu X, Chen F, Zhu B, Li C, et al. Configured quantum reservoir computing for multi-task machine learning. Sci Bull. 2023;68(20):2321–2329. doi: 10.1016/j.scib.2023.08.040

65. Abbas AH, Maksymov IS. Reservoir Computing Using Measurement-Controlled Quantum Dynamics. Electronics. 2024;13(6):1164. doi: 10.3390/electronics13061164