Neural Computing and Applications (2021) 33:13723–13743
https://doi.org/10.1007/s00521-021-06009-3

ORIGINAL ARTICLE

Deep neural networks for quantum circuit mapping

Giovanni Acampora[1,2] · Roberto Schiattarella[1]

Received: 7 January 2021 / Accepted: 31 March 2021 / Published online: 9 May 2021

Abstract
Quantum computers have become reality thanks to the effort of some majors in developing innovative technologies that enable the usage of quantum effects in computation, so as to pave the way towards the design of efficient quantum algorithms to use in different applications domains, from finance and chemistry to artificial and computational intelligence. However, there are still some technological limitations that do not allow a correct design of quantum algorithms, compromising the achievement of the so-called quantum advantage. Specifically, a major limitation in the design of a quantum algorithm is related to its proper mapping to a specific quantum processor so that the underlying physical constraints are satisfied. This hard problem, known as circuit mapping, is a critical task to face in quantum world, and it needs to be efficiently addressed to allow quantum computers to work correctly and productively. In order to bridge above gap, this paper introduces a very first circuit mapping approach based on deep neural networks, which opens a completely new scenario in which the correct execution of quantum algorithms is supported by classical machine learning techniques. As shown in experimental section, the proposed approach speeds up current state-of-the-art mapping algorithms when used on 5-qubits IBM Q processors, maintaining suitable mapping accuracy.

Keywords Machine learning for quantum computing · Quantum circuit mapping · Quantum computing

1 Introduction

Quantum computing aims to solve intractable computational problems by leveraging quantum mechanics principles like superposition and entanglement to manipulate information in a different and potentially more efficient way than traditional electronic computers. Indeed, the joint use of quantum superposition and entanglement enables a massive parallelism in computation resulting in the design of quantum algorithms capable of achieving performance never seen on classical computers. Some noteworthy examples are Shor's [1] and Grover's algorithms [2]. Shor's algorithm is a polynomial-time quantum computer algorithm for integer factorization; indeed, its computational complexity is $O((\log n)^2(\log\log n)(\log\log\log n))$, where n is the number to factorize, whereas the computational complexity of the best classical algorithm for integer factorization is $O(\exp(c(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}))$, where c is a constant. Similarly, Grover's algorithm is a quantum algorithm for searching an unsorted database with N entries in $O(\sqrt{N})$ time and using $O(\log N)$ storage space, whereas the complexity of the best classical algorithm that performs the same operation in $O(N)$ steps. Recently, other quantum algorithms have been designed in different application domains, from chemistry [3, 4] to artificial intelligence [5–11]. However, even though theoretical algorithms have already been designed, physical realizations of quantum computers able to run these algorithms have been considered as a kind of utopia for a long time. Nevertheless, this changed in recent years in which quantum computers more and more evolved from an academic idea to an upcoming reality thanks to the effort of main majors acting in information and communication technology in developing so-called Noise Intermediate Scale Quantum (NISQ) devices,

✉ Giovanni Acampora
giovanni.acampora@unina.it; giovanni.acampora@na.infn.it

Roberto Schiattarella
roberto.schiattarella@unina.it

[1] Department of Physics "Ettore Pancini", University of Naples Federico II, Complesso di Monte Sant'Angelo, Via Cintia 21, 80126 Napoli, Italy

[2] Istituto Nazionale di Fisica Nucleare, Sezione di Napoli, 80126 Napoli, Italy

i.e. quantum computers with 50-100 qubits may be able to perform tasks which surpass the capabilities of today's classical digital computers [12]. In this scenario, companies such as IBM[1] and Rigetti[2] are providing access to their quantum computers to a broad audience of researchers and practitioners by means of cloud computing technologies. However, although both advanced techniques for designing quantum algorithms and technologies capable of executing such algorithms are available, there are still some technological limitations, which slow down the race to quantum supremacy and advantage. Specifically, there is a significant gap between the quantum resources required to execute quantum algorithms and the resources available in current NISQ devices. In particular, current quantum processors are characterized by a weakly interconnected coupling map, which limits the interactions among different qubits. As a consequence, a quantum algorithm needs to be adapted to a specific quantum processor's coupling map in order to be correctly and efficiently run. This adaptation requires to add a set of SWAP quantum gates to the original quantum algorithm, increasing the quantum error rate and potentially compromising calculations. By virtue of this, there is a strong emergence for efficient techniques of circuit mapping able to minimize the number of SWAP gates useful to run the quantum algorithm in a correct way. This paper faces this key challenge in quantum technologies by introducing *Neural Layout*, the very first approach for quantum circuit mapping based on machine learning, which uses deep neural networks to improve the performance of current methods based on mathematical solvers and heuristic cost functions. Indeed, as shown in Sect. 5, Neural Layout is able to speed up state-of-the-art circuit mapping algorithms used by IBM Qiskit[3] Transpiler[4] when it is used to perform circuit mapping operations on 5-qubits IBM quantum processors. Moreover, the experiments show that Neural Layout outperforms other well-known machine learning techniques in carrying out quantum circuit mapping. The obtained results show that approaches based on deep neural networks can effectively support the design of quantum devices and open the way towards a completely new research area that blends the foundations of machine learning with those of quantum computing.

The paper is structured as described hereafter. Section 2 presents a collection of approaches currently used to solve quantum circuit mapping, and it provides an overview about the application of machine learning has to the design of classical electronics circuits. Section 3 describes the basic concepts of quantum computing and a formal definition of the mapping quantum circuits problem. Section 4 shows the design of Neural Layout and its capabilities in efficiently performing quantum circuit mapping. In Sect. 5, the performance of Neural Layout is assessed and compared to the performance obtained by other machine learning techniques, and to the performance yielded by state-of-the-art circuit mapping algorithms used in IBM Qiskit. Section 6 provides some insights about future research directions mainly aimed at improving the performance of the proposed approach when applied to perform quantum circuit mapping on larger processors previously introduced.

2 Related works

Limiting the error during the current quantum computation is one of the biggest challenges researchers are facing. Efficiently mapping a quantum circuit to a processor is a fundamental step in this direction. Previous approaches to this problem can be classified into two classes. One is to formulate the circuit mapping problem into an equivalent mathematical problem and then apply well-known solvers [13–23]. However, these methods lack scalability as the number of qubits in the circuits increases, suffering from very long run time. Thus, even though optimal solutions computed by these approaches are theoretically useful, they are impractical to be actually used in real scenarios. For this reason, some research proposes a second set of approaches, which are based on heuristic search for sub-optimal solutions [24–32]. However, most of above methods were developed for ideal 1D/2D lattice model and they are not usable with NISQ devices, which are characterized by more complicated coupling maps. In this scenario, a quantum compiler used in IBM quantum computers, named IBM Transpiler, uses two different, not trivial, heuristic approaches to addressing circuit mapping[5]: *Dense Layout*[6] and *NoiseAdaptiveLayout* [34]. DenseLayout carries out a breadth first search starting from each qubit belonging to the processor, so as to compute a collection of connected subsets of qubits. Successively, this approach selects the subset characterized by the highest connectivity and low noise. Finally, the selected subset is given in input to the *reverse Cuthill–Mckee algorithm* to order the qubits in the selected set in ascending order of degree of connectivity. *NoiseAdaptiveLayout* leverages a qubit mapping techniques that uses the calibration

[1] https://quantum-computing.ibm.com.

[2] https://qcs.rigetti.com/.

[3] In this work, the version of Qiskit meta-package used is the 0.19.6.

[4] https://qiskit.org/documentation/stubs/qiskit.compiler.transpile.html.

[5] At the time of this research, the transpiler did not yet provide the possibility of using SABRE qubit placement algorithm presented in [33].

[6] DenseLayout.

information from the backend devices and evaluates several optimal and heuristic mappings. To the best of our knowledge, there are no previous works about the possibility to use artificial intelligence (AI) and machine learning (ML) to address the quantum circuit mapping problem as proposed in this manuscript.

Although quantum technologies are very recent and machine learning has never been used to support the development of these technologies, there is a strong scientific production where machine learning has been widely used in supporting design, simulation, and optimization of classical circuits, as reported in [35]. Some examples of recent and significant application of machine learning techniques applied to circuit design are shown hereafter. The research presented in [36] proposes a two-layer evolutionary scheme based on genetic programming (GP) and neural network (NN), which uses a divide-and-conquer approach to design analog circuits by especially focusing on how to select component values and topology sizes for a given circuit topology. Several researches, instead, focus their attention on developing electronic design automation (EDA) techniques using machine learning [37, 38]. Furthermore, NNs have been used in applications ranging from circuits optimization [39], replacing empirical modelling solutions or numerical modelling methods limited by their computationally expansive behaviour, to resource optimization for circuit simulation [40] or to circuit partitioning, as shown in [41], where a neural network model was proposed for circuit bipartitioning. In particular, in this research the massive parallelism of a NN has been successfully exploited to reduce the external wiring between the partitions and to balance the partitions of circuit. In [42–46], NNs were also applied to develop fault-diagnostic system for analog electronic circuits.

NNs are not the only machine learning algorithms applied to analog and digital circuits design and optimization. The research in [47] offers a review on different methods of Machine Learning such as K-nearest neighbour (KNN) Logistic regression (LR) or Support vector machines(SVM) and their usage in analog circuits. In particular, in [48] SVM and LR are compared to NN for the task of automated performance-driven placement of analog integrated circuit.

All above studies prove that machine learning algorithms are particularly suitable to address issues in the design, simulation and optimization of classical circuits by providing several benefits both in terms of accuracy of solutions and computational time. These studies were the inspiration for using machine learning to support the design of quantum technologies by efficiently carrying out a critical task such as the quantum circuit mapping, so as to prove that aforementioned benefits can be used to address the limits previously highlighted by current circuit mapping approaches.

3 Basic concepts: quantum computing, quantum circuit mapping, and state-of-the-art methodologies

This section introduces the basic concepts of quantum computing and the details of the problem to be solved, known as quantum circuit mapping. Moreover, some details about current techniques for quantum circuit mapping used in state-of-the-art software tools for quantum computing, such as IBM Qiskit, are provided.

3.1 Quantum computing

Quantum computing is an alternative computational paradigm that uses quantum mechanics effects such as superposition and entanglement to introduce an intrinsic and massive parallelism in computation so as to enable the design of quantum algorithms, which could be more efficient than their classical counterparts. This paradigm uses the qubit as basic unit to store and manage information. Informally speaking, while a classical binary digit (bit) can be in a classical state either 0 or 1, a qubit can be in a quantum state that is a quantum superposition of 0 and 1, before being measured. In a sense, before performing a quantum measurement, a qubit may have simultaneously the values 0 and 1 and, only when it is measured, it "collapses" to one of these two values, corresponding to classical bits. When a qubit is in a superposition of states, it can be said that it has an amplitude associated with each state. Two key aspects are related to the concept of amplitude, two knobs to adjust the configuration of a qubit's superposition:

- The *magnitude* associated with each basic state of a qubit (0 or 1), which is related to the probability that a qubit will collapse to the state 0 or 1 after a quantum measurement;
- The *relative phase* between the different states in the qubit's superposition determines the degree to which different computational paths interfere constructively or destructively. Note that though the probability to measure a certain state in the superposition is related only to the magnitude associated with it, the relative phase takes a key role in several quantum algorithms such as *amplitude amplification*, *quantum Fourier transform* or *phase estimation* in modelling desired magnitude distributions and enabling the design of efficient quantum algorithms in different applications domains.

The magnitude and relative phase are values available for being exploited when computing, and it is worth to think of them as being encoded in a single qubit.

Like classical computation, quantum computing uses logic gates (*quantum gates*) to change the state of qubits and transform input information into a desired output. These quantum gates are *reversible*. This means that, given an output, the corresponding input can be retrieved. Some examples of quantum gates are listed in Table 1.

As in classical computation where single bits can be aggregated together to form classical register, one or more qubits can be aggregated together to form *quantum registers*. A classical register can contain any arbitrary number of bits, say n. A quantum register can hold any superposition of n-qubit quantum states. Therefore, while an n-bit classical register can embody any one of 2^n possible numbers, it can store just one at a time. An n-qubit register, on the other hand, can store any combination of 2^n numbers. Moreover, some of the qubits belonging to a single register or multiple registers can be entangled among them; the entanglement is a non-local property of two or more qubits that allows a set of qubits to express higher correlation than is possible in classical systems.

Formally speaking, a qubit is represented by a unit vector, namely $|\psi\rangle$, of a two-dimensional Hilbert space:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$, and $|0\rangle$ and $|1\rangle$ are the basis states of the Hilbert space:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2}$$

The value $|\alpha|^2$ is to be interpreted as the probability that, after measuring the qubit, it will be found in state $|0\rangle$, whereas $|\beta|^2$ is to be interpreted as the probability that, after measuring the qubit, it will be found in state $|1\rangle$. The notation adopted to represent a qubit ($|\cdot\rangle$) is named *ket notation*, and it is used in quantum mechanics to model quantum state vectors.

There is an alternative representation of a qubit enabling its visualization in a three-dimensional reference system. This representation is named *Bloch sphere*[7], and it uses the following representation of a qubit to work, derived from the polar form of complex numbers [49]:

$$\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{3}$$

By using this representation, only two real numbers, namely θ and ϕ, are necessary to identify a qubit, and consequently it can be represented as an arrow from the origin to the surface of a three-dimensional sphere of $\mathbb{R}^3$ of radius 1, as shown in Fig. 1. According to the general notation presented in (1), θ refers to the magnitude associated with each basis state and ϕ is the relative phase between them.

The evolution of a closed quantum system is described by special linear operators, *unitary operators*[8] U which operate on qubits as follows:

$$U|\psi\rangle = U[\alpha|0\rangle + \beta|1\rangle] = \alpha U|0\rangle + \beta U|1\rangle \tag{4}$$

Therefore, for each of the above quantum gates, there will be a unitary operator capable of formalizing its behaviour. In general, the unit operators perform rotations of the vectors corresponding to the quantum states in a two-dimensional Hilbert space. As an example, let us consider the Pauli gate acting on a single qubit. It is the quantum equivalent of the NOT gate for classical computers and, for this reason, it is sometimes called bit-flip. The unitary matrix associated with the Pauli-X gate is the one reported in Table 1. Let us suppose to have a qubit in a state $|\psi\rangle = 1 \cdot |0\rangle + 0 \cdot |1\rangle$, where $\alpha = 1$ and $\beta = 0$, $|\alpha|^2 = 1$ and $|\beta|^2 = 0$, and compute $|\psi'\rangle = X|\psi\rangle$ (see Fig. 2):

$$|\psi'\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5}$$

In addition to the X gate, other quantum single-qubit gates can be used to change the state of a qubit. Among these, the R_y and R_z gates are of particular interest, because they allow a simple and direct modification of the aforementioned magnitude and phase knobs. As an example, the R_y gate performs a single-qubit rotation through angle θ radians around the y-axis. The R_y rotation mainly acts on magnitude knob of qubit. Thus, this gate modifies the probability that a qubit in a state $|\psi\rangle$ will collapse to 1 or 0, after measuring it.

As useful as single qubits can be, they are much more powerful in groups. Indeed, when a quantum device has access to more than one qubit, it can make use of another powerful quantum phenomenon, entanglement. Formally speaking, a size n quantum register is a quantum system comprising n individual qubits, where each qubit q_i with $i \in \{0, \ldots, n-1\}$ is represented by a unit vector of two-dimensional Hilbert space $\mathcal{H}_i$ with $i \in \{0, \ldots, n-1\}$. Then, the resulting quantum register is represented by a unit vector of n-dimensional Hilbert space:

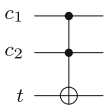
$$\mathcal{H} = \mathcal{H}_{n-1} \otimes \mathcal{H}_{n-2} \otimes \cdots \otimes \mathcal{H}_0$$

where the symbol $\otimes$ computes the tensor product of two vector spaces. Quantum registers evolve by using quantum

[7] Named after the German physicist Felix Bloch.

[8] A linear operator is said to be unitary if $UU^\dagger = U^\dagger U = I$, where $U^\dagger$ denotes the adjoint of the operator U and I the identity matrix.

Table 1 Examples of quantum gates

Symbol	Name	Description	Matrix
	Pauli X (also NOT)	Logical bitwise NOT	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
	Hadamard H	The Hadamard gate is a single-qubit operation creating an equal superposition of the two basis states, typically $\|0\rangle$ and $\|1\rangle$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
	Rotation R_y	The R_y gate is one of the Rotation operators. It is used to modify the magnitude knob of qubits.	$\begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$
	Rotation R_z	The R_z performs a rotation of ϕ around the Z-axis direction	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$
	S Gate[1]	This is an R_z-gate with $\phi = \pi/2$. It does a quarter-turn around the Bloch sphere.	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$
	CNOT	Controlled NOT: if (c) then NOT(t). This gate is used to enable the quantum entanglement between two qubits	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
	SWAP	The SWAP gate performs the swap of two qubit states q0 and q1	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
	Toffoli CCNOT	The Toffoli gate: if (c_1 AND c_2) then NOT(t)	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$
	CSWAP	Conditional SWAP: if (c) then SWAP(t_1,t_2)	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

[1] It is important to note that unlike every gate in the table, the S-gate is not its own inverse. As a result, the $S^{\dagger}$-gate is a different gate that performs a R_z-rotation with $\phi = -\pi/2$

gates acting on two or more qubits. An important example of a quantum gate acting on two qubits is the Controlled NOT (*CNOT*). Precisely, the *CNOT* gate operates on two qubits, a *control* qubit and a *target* qubit, and it works as follows: apply the logical quantum NOT operation to the target qubit, but only if the control qubit has the value 1. The matrix representation for the unitary operator related to the *CNOT* gate is shown in Table 1. The role of *CNOT* gate is particularly relevant when the control qubit is in superposition state because, in this case, it enables the quantum entanglement and for this reason it is called *entangling gate*. Together with superposition, entanglement is another quantum property useful for improving the performance of computing devices. Quantum entanglement often is seen as a key ingredient if quantum computers are to demonstrate an advantage over classical computers. In particular, if a quantum system is not highly entangled, it can often be simulated efficiently on a classical computer. So far, the simplest form of entanglement, *Bell states*, enable tasks such as quantum cryptography, super-dense coding, teleportation, and entanglement swapping [50].

All the above gates enable the design of quantum algorithms by means of circuits composed of collection gates at the same way classical computation uses gates such as *AND*, *OR*, and *NOT* to develop algorithms on classical computers. An example of quantum circuit is in Fig. 8. In particular, classical algorithms can be designed by using a subset of classical gates, such as *NAND* and

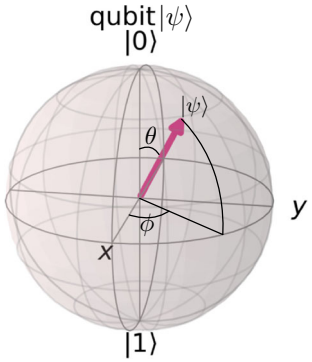


Fig. 1 Bloch sphere

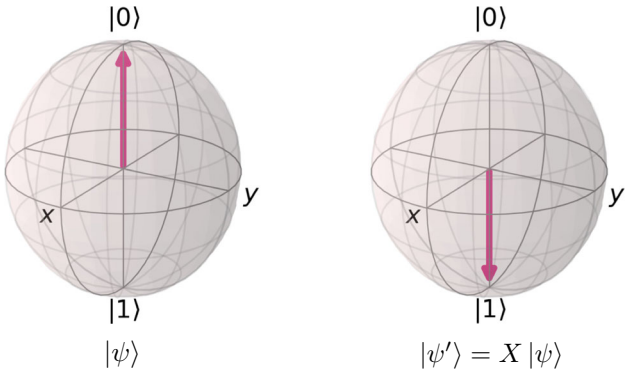
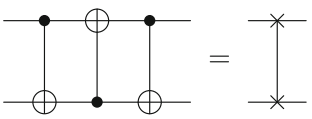


Fig. 2 The X gate applied to a qubit in state $|0\rangle$: $X|0\rangle = |1\rangle$

FANOUT, capable of reproducing the behaviour of any function of the form: $f : \{0,1\}^n \rightarrow \{0,1\}^m$. At the same way, a fundamental theoretical result in quantum computing proves that $\{R_y, R_z, CNOT\}$ is a universal gate set [51]. Any quantum gate can be unrolled in terms of this set of gates. As an example, the SWAP decomposition in terms of CX is the following [52]:



(6)

To conclude this brief introduction to the basic elements of quantum computation, it is important to highlight that all the above single-qubit quantum gates can by view as a special case most general universal gates. In this context, the U_3 gate is the most general of all single-qubit quantum gates because it is able to model the behaviour of all the aforementioned single-qubit gates by means of three different parameters, θ, ϕ, and λ:

$$U_3(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\lambda+i\phi}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{7}$$

Quantum software such as IBM Qiskit provides also U_2 and U_1-gates, which are specific cases of the U_3 gate in which $\theta = \frac{\pi}{2}$, and $\theta = \phi = 0$, respectively:

$$U_2 = U_3\left(\frac{\pi}{2},\phi,\lambda\right) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i\lambda+i\phi} \end{pmatrix} \tag{8}$$

$$U_1 = U_3(0,0,\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix} \tag{9}$$

You will notice that the U_1-gate is equivalent to the R_z-gate.

Even though, every gate in this paper could be specified as $U_3(\theta,\phi,\lambda)$, it is unusual to see U_3 in a circuit diagram, possibly due to the loss of semantics induced by this gate. However, its role is crucial for enabling an efficient execution on quantum circuits on real quantum hardware. Indeed, in order to run a specific quantum circuit on a real IBM quantum hardware, it needs to be compiled and, during this step, all single-qubit operations are converted to an appropriate set of U_1, U_2 and U_3 gates. For this reason, these gates are called *physical gates*. Because physical gates are able to model the behaviour of single-qubit gates, they do not represent a universal gate set, but, however, the union of physical gate and entangling gate, such as CNOT, represents a universal gate set, where sequences of the entangling gates will be used to model the behaviour of every other multi-qubit gate. However, even though an universal gate set enables the execution of any quantum circuit on a given processor by using a quantum compiler, there are some technological challenges to face in order to make a compiler able to convert a circuit and efficiently run it. One of the most important challenges to face in this scenario is the circuit mapping problem.

3.2 Decoherence times and error rates

In the previous section, an overview about quantum computing was provided. However, although in this years quantum devices are growing fast, some hardware limitations still affect the performance they can achieve. One of them is surely related to the short decoherence times of current systems. Quantum decoherence can be viewed as the loss of information from a system into the surrounding environment. Formally speaking, a quantum system is said to be coherent as long as exists a definite phase relation between different states. However, the interaction of a quantum device with the surrounding environment causes

the first to pass from a coherent state to a statistical mixture of states, which no longer contains the quantum information encoded in its states. This loss of information is mainly due to two types of decoherences: transverse relaxation and longitudinal relaxation. The first one is caused by the loss of coherence between the relative phases of the amplitudes of a quantum state. The resulting decoherence time is indicated by T_2. The second one is due to population decay: excited states tend to decay spontaneously at the ground state in a certain typical time T_1. Considering this, to ensure a proper execution of a quantum algorithm it has to be executed in a times that is shorter than the decoherence times of the system. In recent years, incredible efforts have been done in this direction. Currently, execution times for single qubit gates are on the order of nanoseconds, while typical decoherence times vary from tens to hundreds of microseconds. This makes it possible to manipulate qubits using quantum gates.

A further limitation, indeed, is related to the high error rates in the current quantum devices. The main issue is due to the high error rate of the CNOTs and readout operations. Each CNOT gate has a typical error rate order 10^{-2}, while single-qubit gates have error rate order 10^{-4}. The reliability of a quantum circuit is therefore, mainly influenced by the number of CNOTs in it. Similarly, also the measurement operation has an error rate not negligible, order 10^{-2}.

For the IBM devices, decoherence times and error rates are daily measured and provided to users. Lastly, in the current quantum hardware the qubits are not entirely connected each other, but a quantum processor is characterized by a coupling map which limits the interactions of qubits by means of CNOT gates. In the next section, this limitation will be widely discussed, highlighting the importance of optimal mappings of quantum circuits onto quantum processors.

3.3 Circuit mapping on quantum processors

Currently, quantum computing is becoming a popular paradigm in computation thanks to the cloud-based availability of so-called Noisy Intermediate-Scale Quantum (NISQ) devices, i.e. quantum processors equipped with a low number (typically 50–100) of qubits not fully tolerant to quantum noise, which enable researchers and practitioners in developing quantum algorithms [12]. Besides the issues related to their size and noise, another critical problem that characterizes this kind of technology is the low connectivity of their *coupling map*, for which each qubit is connected to a limited number of other qubits. As a consequence, two-qubit gates, such as a CNOT, cannot be placed in a circuit if the target and control qubits are not physically connected in the processor coupling map.

As an example, Fig. 3 shows the coupling map of an IBM Q processor named Burlington composed of $n = 5$ qubits. The IBM Q Burlington coupling map defines a set $\{(0,1),(1,0),(1,2),(2,1),(1,3),(3,1),(3,4),(4,3)\}$ containing the pairs of qubits that can be used as target and control in a CNOT gate; thus, on this processor, only 8 out of 20 ($n^2 - n$) pairs can be used to position a CNOT in a circuit, severely limiting the possibilities offered by quantum devices. However, current quantum technologies use the so-called quantum compilers that are able to solve the above issue. These compilers use a sequence of SWAP operations (Table 1) between adjacent qubits so as to enable the computation of a CNOT gate between two non-adjacent qubits. However, this approach could very negatively affect the execution of a quantum circuit for two fundamental reasons:

- It increases the circuit depth and, consequently, it increases the probability of a decoherence error, with the consequent loss of the information that the circuit carries.
- Each CNOT gate has a typical error rate order 10^{-2}. Thus, the execution of a single SWAP operation involves the execution of three gates with an high error rate, negatively affecting the computation of the whole circuit.

As a consequence, there is a strong emergence for quantum compilers able to identify an optimal initial mapping among circuit qubits and physical qubits, so as to minimize the number of SWAP operations required to execute the compiled circuit. This optimization problem is known as circuit mapping, and it is formalized as described hereafter. Let $C_n = \{c_i\}_{i\in\mathbb{N}}$ be the set of all quantum circuits that can be designed with n qubits belonging to the set $Q_c = \{q_0^c, q_1^c, \ldots, q_n^c\}$, and let P be a NISQ processor composed of $m \geq n$ qubits belonging to the set $Q_p = \{q_0, q_1, \ldots, q_m\}$ and characterized by a coupling map $M_p = \{(q_i, q_j) \mid q_i, q_j \in Q_p \text{ and } i \neq j\}$, then a collection of initial quantum circuit mappings consists in a set of functions $F_M = \{f_k : C_n \rightarrow \wp(Q_C \times Q_P)\}_{k\in\mathbb{N}}$ where each f_k relates the qubits of each circuit in C_n to the qubits of the processor P; the solution of the circuit mapping problem is a function $f^* \in F_M$ able to minimize the number of SWAP operations to add to original circuits and enable its efficient execution on the processor P. In Fig.[9] 4, there is a graphical representation of a circuit mapping function $\bar{f} \in F_M$, which maps a circuit $\bar{c} \in C$ designed with $n = 5$ qubits on a processor P composed of $m = 20$ qubits and a coupling

[9] https://qiskit.org/documentation/apidoc/transpiler.html.

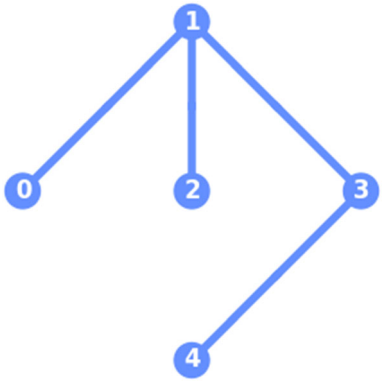


Fig. 3 IBM Q Burlington coupling map

map M_P shown on the right side of the figure. In particular, $\bar{f}(\bar{c}) = \{(q_0^C, q_1), (q_1^C, q_0), (q_2^C, q_5), (q_3^C, q^6), (q_4^C, q_7)\}$.

The circuit mapping problem has been proved to be NP-complete in [53], and, as a consequence, the computation of its exact solution could be not suitable to deal with future generation of quantum processors characterized by thousands or millions of qubits[10]. Thus, there is a strong need of approximate algorithms capable of efficiently computing sub-optimal solutions for this problem and pave the way towards the next generation of compilers for quantum computers. In this paper, this challenge has been faced by introducing the first approach based on machine learning to efficiently address the circuit mapping problem on real IBM Q processors.

4 Neural layout: a deep neural network for quantum circuit mapping

This section introduces Neural Layout, a machine learning-based approach aimed at solving above circuit mapping problem. In particular, Neural Layout acts in three sequential steps: (1) initially, it models the quantum circuit mapping problem as a conventional classification task, (2) successively it trains an appropriate deep neural network aimed at efficiently addressing the classification task for circuit mapping and, finally (3) it uses a refinement step to make neural network predictions compliant with some logical constraints that characterize quantum processors.

4.1 Modelling quantum circuit mapping as a classification task

In general, a classification problem can be defined as the task of estimating a label y from a K-dimensional input vector $\mathbf{x}$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$ and $y \in \mathcal{Y} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_Q\}$. This task is accomplished by using a classification rule implemented by a function $g : \mathcal{X} \rightarrow \mathcal{Y}$ able to predict the label of new patterns, where g is learnt and adjusted by using a training set composed of N points, represented by a set $D = \{(\mathbf{x}_i, y_i), \text{with } i = 1, \ldots, N\}$. In order to model the quantum circuit mapping problem by a classification task, let us to consider a quantum circuit $c \in C_k$ composed of k qubits belonging to the set $Q_C = \{q_1^C, q_2^C, \ldots, q_k^C\}$, and a quantum processor P composed of n qubits belonging to the set $Q_P = \{q_1, q_2, \ldots, q_n\}$ and characterized by a specific coupling map M_P. Then, a classification task for quantum circuit mapping is implemented by a function ϕ that uses an input vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^K$, containing a set of features characterizing both the circuit c and the processor P on which running c, to estimate an array $\mathbf{y}$ composed of n elements, where each elements belongs to the mapping label set $\mathcal{Y} = \{-1\} \cup \{1, \ldots, k\} \subset \mathbb{N}$. The function ϕ is learned and adjusted by using a training set $D_\phi = \{(\mathbf{x}_i, \mathbf{y}_i), \text{with } i = 1, \ldots, N\}$, where $\mathbf{x}_i \in \mathcal{X}$ is a feature set and $\mathbf{y}_i \in \mathcal{Y}^n$ is an array of mapping labels representing an ideal mapping from the circuit c to the processor P. Reasoning in this way, the function ϕ will be able to provide mapping capabilities similar to the best algorithms currently used in quantum circuit mapping, but with a lower computational complexity, as shown in the experimental results section. The aforementioned input vector $\mathbf{x}$ is composed of a set of features characterizing both the circuit c and the processor P. In particular, with respect to the circuit c, the following information has been considered:

- an integer value representing the number of qubits composing the circuit;
- an integer value representing the total number of CNOT gates in the circuit c;
- a matrix of integer values where the item $[i, j]$ contains the number of CNOT gates between the control qubits q_i^C and the target qubit q_j^C of the circuit c. To make the network work with any circuit width less than or equal to the number of processor qubits, this matrix of integers is set of size $n \times n$.

At the same time, with respect to the processor P, the following information has been taken into account:

- an array of real values where each value represents the error rate of a CNOT using q_i as control qubit and q_j as target qubit for each $(q_i, q_j) \in M_P$;
- an array of real values where each value represents the execution time of a CNOT using q_i as control qubit and q_j as target qubit for each $(q_i, q_j) \in M_P$;

[10] IBM Quantum Roadmap 2023.

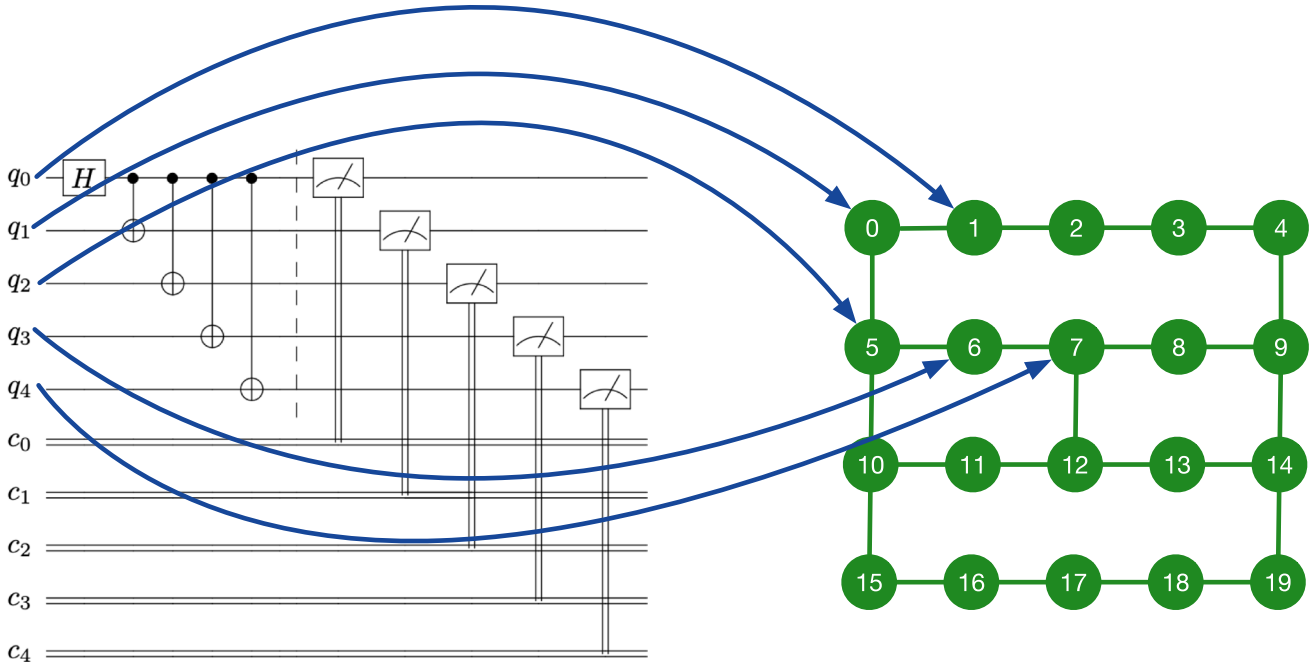


Fig. 4 Graphical representation of a circuit mapping function $\bar{f} \in F_M$

- an array of real values where each value represents the transverse relaxation time (T_2) characterizing a qubit q_i of the processor P
- an array of real values where each value represents the longitudinal relaxation time (T_1) characterizing a qubit q_i of the processor P
- an array of real values where each value represents the readout error characterizing a qubit q_i of the processor P.

A schematic view of the features related to the quantum circuit mapping problem is provided in Table 2. The output array $\mathbf{y} = [y_1, y_2, \ldots, y_k]$ is composed of n items where each $y_i \neq -1$ represents the mapping between the qubits $q^C_{y_i} \in C$ and $q_i \in P$, whereas $y_i = -1$ means that the processor qubit q_i must remain un-mapped. In other words, the collection $\{\mathbf{y}\}$ of all arrays generated by the above function ϕ by applying it to all circuits $c \in C_k$ encodes a function $f' : C \rightarrow \wp(Q_C \times Q_P) \in F_M$, and our approach uses the dataset D_ϕ to attempt to learn a function ϕ capable of approximating the aforementioned function f^* in a proper way.

In order to provide a further clarification about the representation of the circuit mapping problem by a classification task, a practical example is proposed hereafter. Let us to consider a quantum circuit c made up of 5 qubits $Q_C = \{q^C_0, q^C_1, \ldots, q^C_4\}$ and a quantum processor P composed of 5 qubits $Q_P = \{q_0, q_1, \ldots, q_4\}$, and let us suppose that the above classification function ϕ computes an array $\mathbf{y} = [3, 2, 4, 0, 1]$ starting from the set of features modelling the circuit c and the processor P. Then, the array $\mathbf{y} = [3, 2, 4, 0, 1]$ corresponds to the following mapping between circuit and processor qubits:

$$\begin{aligned} &q^C_3 \rightarrow q_0; \\ &q^C_2 \rightarrow q_1; \\ &q^C_4 \rightarrow q_2; \\ &q^C_0 \rightarrow q_3; \\ &q^C_1 \rightarrow q_4. \end{aligned} \tag{10}$$

Once the circuit mapping problem has been defined as a classification task, it is necessary to design an appropriate algorithm based on a deep neural network to learn the function ϕ.

4.2 Designing a deep neural network for circuit mapping

The identification of the aforementioned function ϕ is based on the design of Neural Layout, a deep neural network extended with a constraint satisfaction method. This network is composed of an input layer, a collection of hidden layers, and an output layer. The input layer has been designed by taking into account the size of the vector $\mathbf{x}$ introduced in the definition of circuit mapping as classification task, and containing the features that characterize both a quantum circuit and a quantum processor on which

Table 2 Summary table of the features used by the proposed DNN approach for circuit mapping

Feature name	Feature description	Dimension
Quantum circuit features		
N_{qubits}	Number of qubits in C	1
N_{CNOT}	Number of CNOTs in C	1
$N_{CNOT}^{i,j}$	Total number of CNOT gates between each pair of qubits q_i and q_j in C.	20
Quantum processor features		
$CNOT_{ER}$	CNOT error rate between each pair of qubits connected in the coupling map	8
$CNOT_{ET}$	CNOT execution time between each pair of qubits connected in the coupling map	8
T_2	Transverse relaxation time for each processor qubit	5
T_1	Longitudinal relaxation time for each processor qubit	5
E_{Ro}	Readout error for each processor qubit	5

The first column contains the name of then feature; the column contains the description of each feature; the last column represents the dimensionality of each feature

running the circuit. Let us suppose to have a circuit c composed of k qubits, P a quantum processor composed of m qubits, M_p the coupling map related to P, and l the cardinality of M_p, then the input layer of the proposed network is composed of $2 + (m^2 - m) + 3 \cdot m + 4 \cdot l$ nodes. The first term of this formula is related to two features representing the number of qubits and the number of CNOT gates belonging to the circuit c; the term $(m^2 - m)$ refers to the maximum number of CNOT gates that can be placed between each pair of qubits belonging to a generic circuit computable by the processor P; the term $3 \cdot m$ describes the number of features related to the qubits calibration data of the processor P, namely the decoherence times T_1, T_2, and the readout error; finally, the term $4 \cdot l$ refers to the error rate and execution time of the CNOT gate for each pair of qubits connected in the coupling map M_p. The output layer of the proposed network has been designed to model the vector $\mathbf{y}$ introduced in the definition of circuit mapping as classification task, and containing the set of labels used to map the qubits related to the quantum circuit to the qubits related to quantum processor where the circuit will be run. In particular, the output layer is organized in a collection of m slots, where each slot outputs a single item of the vector $\mathbf{y}$.

More properly, each slot can be considered as a complex neural structure composed of different hidden layers and output layer embodying $m + 1$ neurons and equipped with a *softmax* activation function [54]. Formally speaking, the output layer of the ith slot, named $slot_i$, is an array $P(y_i) = \left[p_0^i, p_1^i, \ldots, p_{m-1}^i, p_{-1}^i\right]$, where p_j^i corresponds to a neuron representing the probability that the ith item of the array $\mathbf{y}$ equals to j, and $\sum_{j=-1}^{m-1} p_j^i = 1$, due to the use of the softmax activation function. By virtue of this modelling scheme, the output vector $\mathbf{y}$ can be obtained by selecting the arg max of each $P(y_i)$ value related to each $slot_i$ with $i = 0, 1, \ldots, m$, as shown in the Eq. (11).

$$\mathbf{y} = [\arg\max(P(y_0)), \arg\max(P(y_1)), \ldots, \arg\max(P(y_m))] \tag{11}$$

However, this approach can compute not feasible solutions for the circuit mapping problem, because it is not certain that $\arg\max(P(y_i)) \neq \arg\max(P(y_j)) \quad \forall i, j, \text{ and } i \neq j$. When that happens, it would lead to map the same circuit qubit q^C onto two different processor qubits q_i and q_j (see the mapping in equation (12) in the case study for a practical demonstration of this issue). In order to address this feasibility issue, our approach introduces and uses a repair operator to move the set of unfeasible solutions generated by the above deep neural network to a feasibility area. This repair operator, named Ξ, can be summarized as follows:

(1) Let $\mathcal{F} = \{P(y_i)\}_{i=0}^{m-1}$ be the set of vectors computed by the different slots of the DNN;
(2) Let $\bar{\mathbf{y}} = [\bar{y_0}, \bar{y_1}, \ldots, \bar{y_m}]$ be a feasible output vector composed of m items and initially initialized as $\bar{\mathbf{y}} = [-1, -1, \ldots, -1]$;
(3) Let $\mathcal{F}' = \{P'(y_i)\}_{i=0}^{m-1}$ be the set of vectors composed by the first k components of each vector in $\mathcal{F}$;
(4) Let $\kappa = 1$ be an iteration variable;
(5) Let y_{jt} be the κ-th maximum among all probability values stored in the vectors $P'(y_i)$, with $i = 0, \ldots, m-1$, belonging to the set $\mathcal{F}'$, located in the t-th position of the vector $P'(y_j)$;
(6) If the value t is not present in $\bar{\mathbf{y}}$, set the value of the j-th item of the vector $\bar{\mathbf{y}}$ to t: $\bar{\mathbf{y}}_j = t$. Go to the step 8);

(7) If t is present in $\bar{\mathbf{y}}$, set $\kappa = \kappa + 1$ and go to the step 5);
(8) Remove $P'(y_j)$ from the collection $\mathcal{F}'$: $\mathcal{F}' = \mathcal{F}' - P'(y_j)$;
(9) If the number items equal to -1 in the vector $\bar{\mathbf{y}}$ is equal to $m - k$ go to the step 10), else go to the step 4) and find a new item for the vector $\bar{\mathbf{y}}$;
(10) End.

In the case study presented in the next section, a step by step application of the repair operator is provided (See Eq. 13).

Once obtained a feasible output vector $\mathbf{y}$ is then possible to decode the circuit mapping that it encodes by following the procedure shown in Sect. 4. The number and type of hidden layers of the proposed network are identified by means of an experimental approach, as shown in Sect. 5. A complete graphical view of the proposed approach for quantum circuit mapping is summarized in Fig. 5.

4.3 A case study for IBM Q Burlington

This section shows the proposed circuit mapping approach at works on a real 5-qubits processor from IBM, named IBM Q Burlington, whose coupling map is shown in Fig. 3. From this map, it appears that $m = 5$ and $l = 4$ and, as a consequence, the required input layer is composed of 53 nodes, whereas the output layer is characterized by $m = 5$ slots, where the i-th slot is a neural structure composed of two hidden dense layers characterized by a relu activation function [54] and an output layer containing $m + 1 = 6$ neurons representing the output vector $P(y_i) = [p_0^i, p_1^i, p_2^i, p_3^i, p_4^i, p_{-1}^i]$ and characterized by a softmax activation function. The structure of the generic slot is shown in Fig. 7.

The set of hidden layers for the whole network is composed of three different layers, where the first two are dense layers, whereas the third layer is a dropout layer that during training time, turns off in a random way some neurons from the previous layer helping to prevent overfitting. The architecture of Neural Layout for circuit mapping on IBM Q Burlington is graphically presented in Fig. 6. In the topology shown in the figure, each block is a neural network layer and the number of neurons for each corresponds to the number of columns of the output tensor.

The behaviour of Neural Layout applied to the IBM Q Burlington has been assessed by considering a random quantum circuit shown in Fig. 8. Moreover, the mapping result computed by Neural Layout has been compared with the so-called *Trivial Mapping Algorithm* provided by IBM Qiskit.

Before applying circuit mapping algorithms, it is necessary to convert the starting circuit into an equivalent circuit containing only elementary gates used by the IBM Q Burlington processor and belonging to the set $\{U_1, U_2, U_3, CNOT, I\}$. This step is named circuit unrolling. Once the circuit is unrolled, its 53 features are encoded in a vector $\mathbf{x}$ which is given input to Neural Layout so as to compute the following set of output vectors:

$$\begin{aligned}
&Slot_0 \rightarrow P(y_0) \rightarrow [0.15, 0.23, 0.12, 0.06, \mathbf{0.4}, 0.04] \rightarrow \arg\max = 4\\
&Slot_1 \rightarrow P(y_1) \rightarrow [0.08, 0.16, \mathbf{0.25}, 0.13, 0.2, 0.18] \rightarrow \arg\max = 2\\
&Slot_2 \rightarrow P(y_2) \rightarrow [0.05, 0.06, 0.31, \mathbf{0.33}, 0.2, 0.05] \rightarrow \arg\max = 3\\
&Slot_3 \rightarrow P(y_3) \rightarrow [\mathbf{0.45}, 0.1, 0.03, 0.15, 0.17, 0.1] \rightarrow \arg\max = 0\\
&Slot_4 \rightarrow P(y_0) \rightarrow [\mathbf{0.27}, 0.2, 0.18, 0.06, 0.18, 0.11] \rightarrow \arg\max = 0
\end{aligned} \tag{12}$$

The outputs related to the different slots of Neural Layout are aggregated together to compute the circuit mapping output vector $\mathbf{y} = [4, 2, 3, 0, 0]$, which unfortunately corresponds to a not feasible solution. However, Ξ operator used by Neural Layout is able to solve this issue by incrementally computing the output vector $\bar{\mathbf{y}}$ as follows:

$$\begin{aligned}
&\mathcal{F}' = \{P'(y_0), P'(y_1), P'(y_2), P'(y_3), P'(y_4)\} \quad \rightarrow y_3^0 \rightarrow \bar{\mathbf{y}} = [-1, -1, -1, 0, -1]\\
&\mathcal{F}' = \{P'(y_0), P'(y_1), P'(y_2), P'(y_4)\} \quad \rightarrow y_0^4 \rightarrow \bar{\mathbf{y}} = [4, -1, -1, 0, -1]\\
&\mathcal{F}' = \{P'(y_1), P'(y_2), P'(y_4)\} \quad \rightarrow y_2^3 \rightarrow \bar{\mathbf{y}} = [4, -1, 3, 0, -1]\\
&\mathcal{F}' = \{P'(y_1), P'(y_4)\} \quad \rightarrow y_1^2 \rightarrow \bar{\mathbf{y}} = [4, 2, 3, 0, -1]\\
&\mathcal{F}' = \{P'(y_4)\} \quad \rightarrow y_4^1 \rightarrow \bar{\mathbf{y}} = [4, 2, 3, 0, 1]
\end{aligned} \tag{13}$$

Thus, the final vector representing the computed quantum circuit mapping is $\bar{\mathbf{y}} = [4, 2, 3, 0, 1]$. The quantum circuit obtained by using the circuit mapping encoded in the vector $\bar{\mathbf{y}}$, shown on the right side of Table 3, is characterized by 10 CNOT gates and a single SWAP gate. Vice versa, a quantum circuit obtained by using the trivial approach provided by IBM Qiskit, shown on the left side of Table 3, is characterized by 16 CNOTs and 3 SWAP gates. In this way, it is possible to state that Neural Layout generates a quantum circuit more tolerant to the effects of

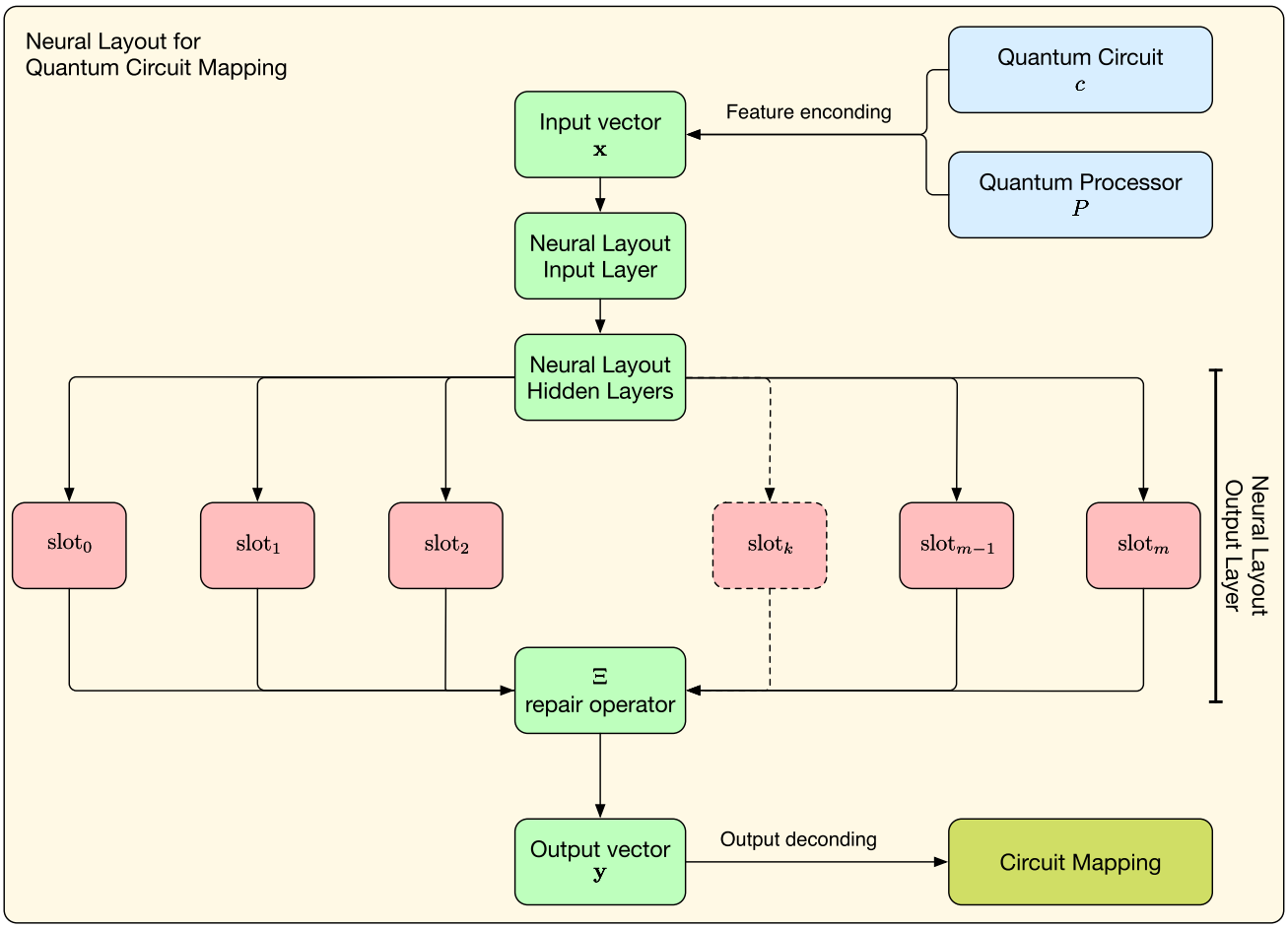


Fig. 5 Circuit mapping workflow via Neural Layout

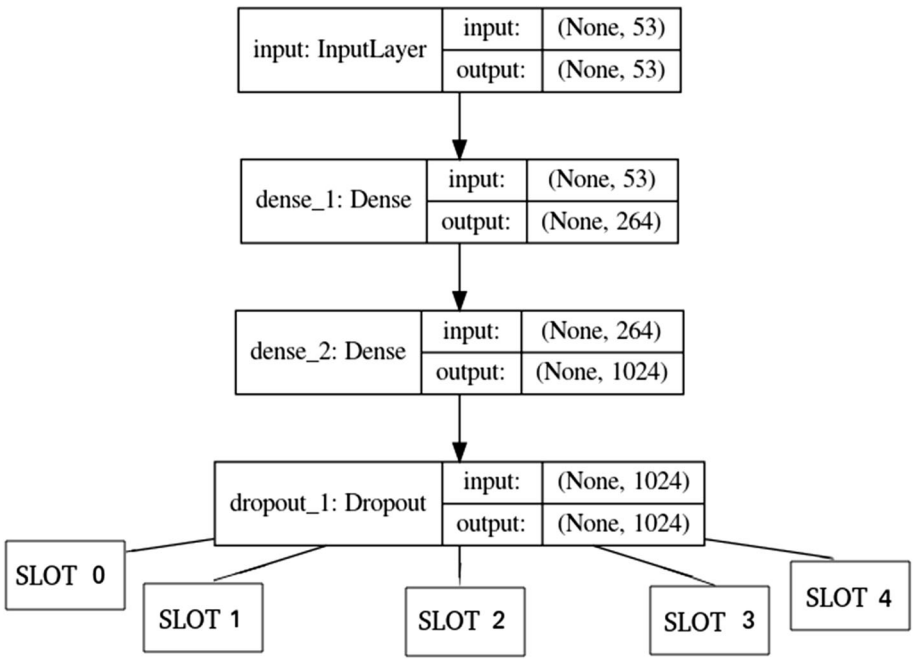


Fig. 6 DNN topology in Neural Layout for mapping quantum circuit on IBM Q Burlington processor

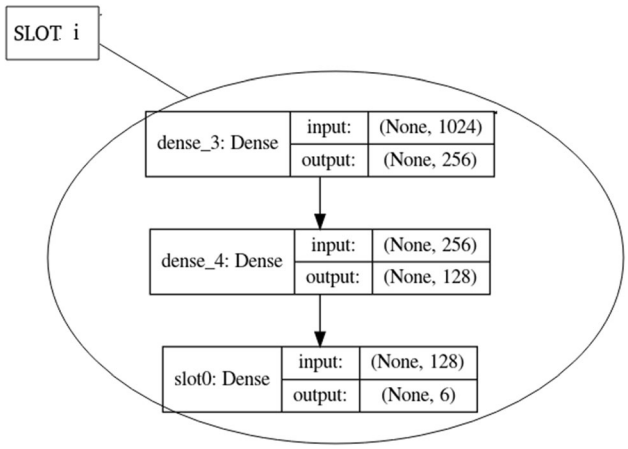


Fig. 7 SLOT topology

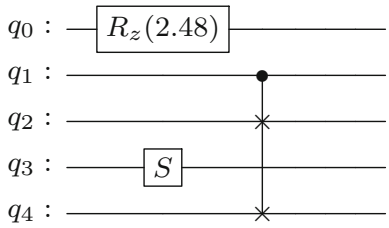


Fig. 8 A case study quantum circuit

quantum decoherence than traditional approaches when used to run the circuit shown in Fig. 8.

5 Experimental results

In this section, the suitability of Neural Layout in performing quantum circuit mapping operations is assessed by comparing its performance with those yielded by other machine learning techniques, and other techniques used in real quantum compilers, such as IBM Qiskit Transpiler. These comparisons were made taking into account both quality of mappings and computational times. The experiments were conducted by using a dataset composed of 5-qubits random circuits mapped on a specific processor provided by the IBM Q Experience, namely IBM Q Burlington.

5.1 Dataset creation for quantum circuit mapping

The dataset used to train and test Neural Layout is composed of 42,039 random unrolled quantum circuits operating on 5 qubits and characterized by a maximum of 10 CNOT gates.

From each circuit, twenty-two features have been extracted (see Sect. 4.1). Moreover, besides circuits' features, the dataset also contains a collection of features related to the processor where above circuits are mapped on, namely IBM Q Burlington, which have been collected by using calibration data provided by IBM[11]. Each instance belonging to the dataset is then related to an output label encoding the best circuit mapping computed by using well-known algorithms available in IBM Qiskit, namely *Dense Layout* and *Noise Adaptive Layout*. In this context, the best mapping of a circuit is the one that generates a new circuit characterized by the smaller number of SWAP gates. The number of SWAP gates in a circuit is computed by means of two IBM Qiskit routing algorithms, named *Look ahead Swap*[12] and *Stochastic Swap*[13].

Therefore, the process of labelling each circuit c randomly generated is outlined as follows:

1. Compute two initial mappings for the circuit c by using both *Dense Layout* and *Noise Adaptive Layout* approaches;
2. For each mapping computed at previous step, compute the number of SWAPs needed to run the circuit c by using both *Look ahead Swap* and *Stochastic Swap* approaches;
3. Choose the mapping requiring the smaller number of SWAP gate executions.

However, above random generation of quantum circuits does not ensure a perfect balance of output classes in the dataset. Indeed, each histogram in Fig. 9 highlights the imbalance in output classes by showing how many times each qubit of the circuit has been mapped to its processor qubit by means of the above approach. Therefore, to develop an efficient predictor, two precautions have been taken. Firstly, a weighted average of the cost function with respect to the target frequencies for each output layer in the model was done. Secondly, appropriate dropout layers have been added to our model to face overfitting due to data imbalance, as shown in Sect. 5.2.

5.2 Analysis of experimental results

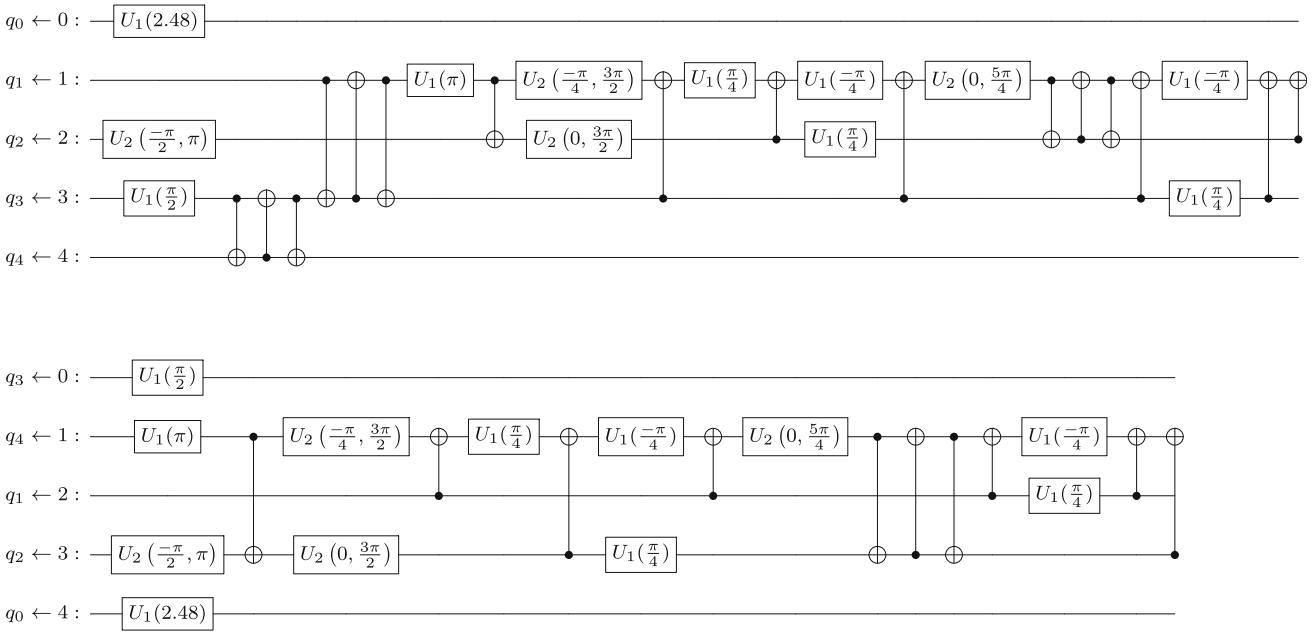
The experimental results section has been organized in three parts. In the first part, a complete experimental setting of the optimal configuration of Neural Layout has been performed so as to identify the right number of hidden layers able to maximize the network accuracy. In the second part, the optimized Neural Layout has been compared

[11] https://qiskit.org/documentation/stubs/qiskit.providers.models.BackendProperties.html.

[12] IBM Qiskit Lookahead Swap Algorithm

[13] IBM Qiskit Stochastic Swap Algorithm.

Table 3 Final quantum circuit obtained using a naive mapping strategy on the left and the Neural Layout mapping approach on the right



The circuits must be read from the bottom. At the beginning of each line, there is the mapping used. The integers refer to circuit qubits, while q_is are processor qubits

to other well-known classifiers such as Random Forest, Support Vector Machine and Logistic Regression Cross-validation, in order to validate its superiority in solving the problem under consideration with respect to other machine learning techniques. In the last part, the performance of the optimized Neural Layout has been compared to the performance yielded by state-of-the-art quantum circuit mapping algorithms provided by IBM Qiskit, both in terms of circuit mapping accuracy and running time, so as to prove that the proposed approach is ready to be used in real quantum compilers.

5.2.1 Experimental setting of neural layout

This experimental session is aimed at identifying the best configuration of Neural Layout. Here, four different configurations, based on the use or not of the Ξ operator and a dropout layer, have been considered:

Configuration 1: Neural Layout composed of an input layer of 53 units, a dense layer of 264 units (D(264)), a dense layer of 1024 units (D(1024)), a dropout layer, and five output slots, without Ξ operator and dropout layer in slot 4;

Configuration 2: Neural Layout composed of an input layer of 53 units, a dense layer of 264 units (D(264)), a dense layer of 1024 units (D(1024)), a dropout layer, and five output slots extended with Ξ operator;

Configuration 3: Neural Layout composed of an input layer of 53 units, a dense layer of 264 units (D(264)), a dense layer of 1024 units (D(1024)), a dropout layer, and five output slots extended with a dropout layer in Slot 4 to face overfitting due to the imbalance of the dataset;;

Configuration 4: Neural Layout composed of an input layer of 53 units, a dense layer of 264 units (D(264)), a dense layer of 1024 units (D(1024)), a dropout layer, and five output slots extended with both Ξ operator and a dropout layer in Slot 4 to face overfitting due to the imbalance of the dataset;

Each of the above configurations has been trained and tested by using the above dataset composed of 42,039 5-qubits random quantum circuits split in training(80%)—

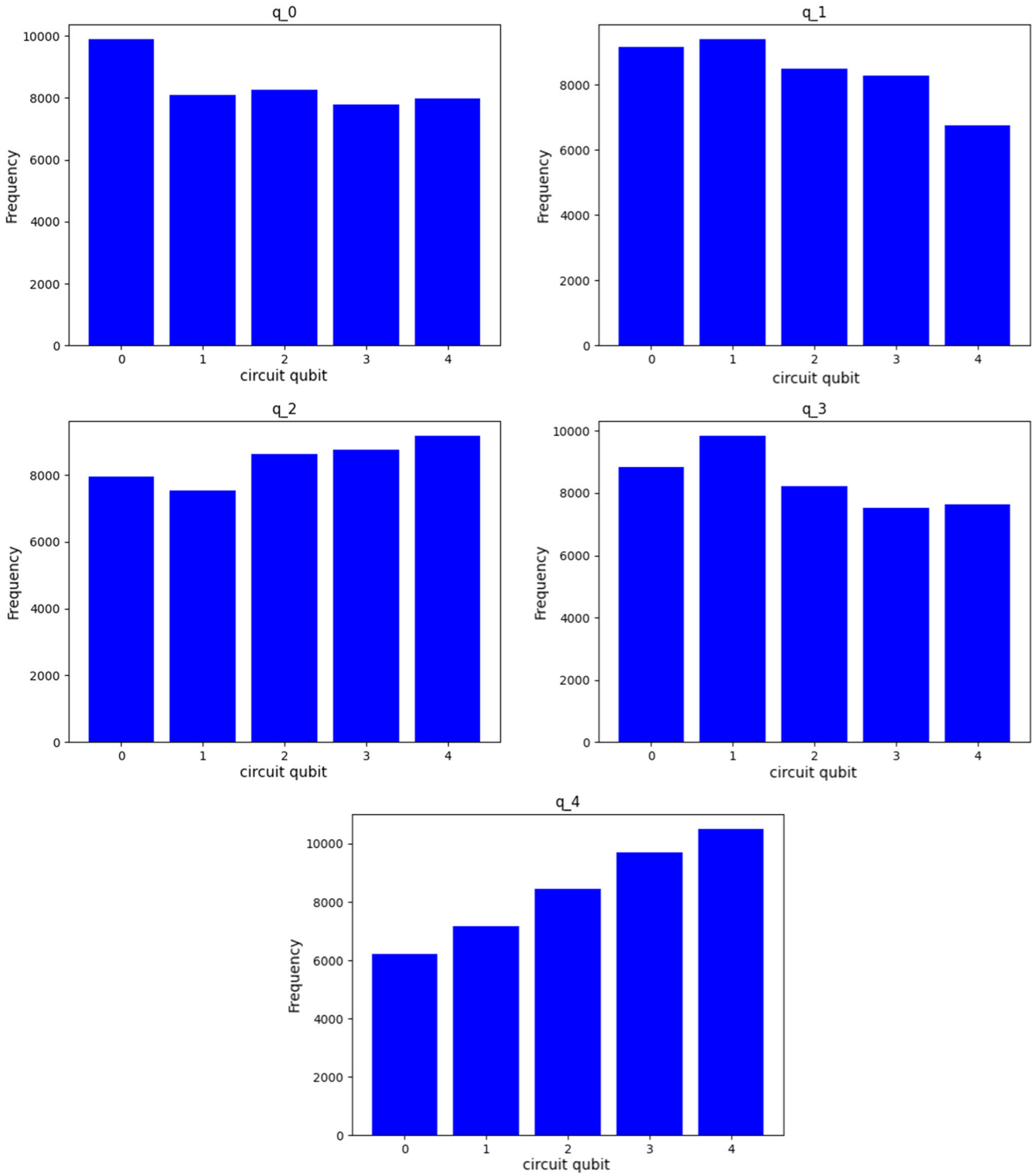


Fig. 9 Histograms representing the target frequencies for each processor qubit of 42039 random quantum circuits used to train and test our model

test(10%)—validation(10%) sets. Each configuration has been trained for 150 epochs using Adam Optimizer with learning rate equals to 0.0005. The experiments were run on a classical computer equipped with an Intel i9 processor and 128 Gb of RAM. The results obtained in terms of accuracy are shown in Table 4. The best accuracy on the test set was obtained by the configuration 4. It should be emphasized that by using the repair operator Ξ accuracy increases significantly, while the exploitation of a dropout layer in slot 4 results in a percentage increase in test

Table 4 Test and training accuracy of the four Neural Layout configurations

Configuration	Test accuracy	Training accuracy
Configuration 1	0.7041	0.8471
Configuration 2	0.7548	0.8867
Configuration 3	0.7188	0.8295
Configuration 4	**0.7645**	0.8752

Bold values reports the maximum value of test accuracy achieved

accuracy and a simultaneous decrease in training accuracy, that can be interpreted as a reduction in overfitting phenomenon.

Once the best configuration has been identified, a further analysis is performed by using confusion matrices to evaluate the performance of this configuration in correctly identifying the i-th qubit of the output vector, with $i \in \{0, 1, 2, 3, 4\}$ (see Fig. 10). Accuracy obtained by the best configuration in identifying the i-th qubit of the output vector is reported in Table 5. Here, the slight difference between the values is due to the aforementioned imbalance presents in the dataset.

In the next sections, the identified best model Neural Layout will be compared with other machine learning classifiers and with other deterministic quantum circuit mapping algorithms already used in real quantum compilers.

5.2.2 Comparing neural layout and machine learning techniques for circuit mapping

In this section, well-known machine learning classifiers, such as Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LR-CV), are compared with Neural Layout with respect to their capabilities in solving the quantum circuit mapping problem modelled by a classification task. These classifiers can be easily adapted to perform the multi-output classification that the proposed mapping approach requires. In order to allow these machine learning techniques to generate feasible solution for the problem, Ξ operator has been applied to their output. For each classifier, different combinations of hyper-parameters have been tried in order to identify a suitable model to solve the quantum circuit mapping problem. All these combinations are summarized in Table 6: Random Forest has been tested with several numbers of estimators and two different splitting criteria for the construction of decision trees, namely Gini and Entropy criterion; furthermore, SVM has been tested with two different kernel functions; LR-CV has been tested by using several values of the regularization parameter c. All these techniques have been trained and tested by using the dataset prepared in Sect. 5.1 and, as shown in Fig. 11, performance yielded by these machine learning techniques in terms of test accuracy is worse than the best configuration of Neural Layout identified in the previous section

As already done for Neural Layout, a further verification of the performance for these classifiers has been made by calculating the test accuracy for the individual output slots. The obtained results are shown in Table 7. This table is a further confirmation of how Neural Layout is far more suitable to perform the operation of mapping quantum circuits than conventional machine learning models. Indeed, the best classifier, Random Forest with Gini criterion, has an average accuracy value for individual slots that is more than 10% lower than Neural Layout, reflecting an overall accuracy value that is more than a 15% lower than that provided by Neural Layout.

5.2.3 Comparing neural layout and IBM Qiskit mapping algorithms

In the last step of the evaluation process, the performance of the optimal configuration of Neural Layout has been compared with two state-of-the-art quantum circuit mapping algorithms, namely Dense Layout and Noise Adaptive Layout, provided by IBM Qiskit, both in terms of quality of the mappings and running times. For this purpose, 1000 random quantum circuits were generated and unrolled in terms of the basic gates. The total number of CNOTs belonging to the unrolled circuits varies between 0 and 50. These unrolled circuits have been mapped on the IBM Q Burlington processor, by using two circuit mappings: the one computed by Neural Layout and the best circuit mapping returned by the execution of the two aforementioned algorithms provided by IBM Qiskit. For each of these two mappings, the depth of the final circuit was calculated by using both routing algorithms made available by Qiskit: Look ahead Swap (LAS) and Stochastic Swap (SS). The plots in Fig. 12 show the results about the quality of the mappings proposed by Neural Layout, as the routing algorithm used varies. On the x axis, there is the number of CNOTs in the unrolled circuit, on the y there is the percentage of final circuits in which the initial mapping provided by Neural Layout returns a circuit characterized by a depth less than or equal to that of the best Qiksit mapping. The plots in blue and red refer to the different routing methods used to obtain the final circuits: in blue the required SWAP insertions were computed with the LAS routing algorithm, while in red they were computed with SS routing algorithm. In green, there is the percentage of random quantum circuits mapped by Neural Layout which have a depth less than or equal to the one obtained by the

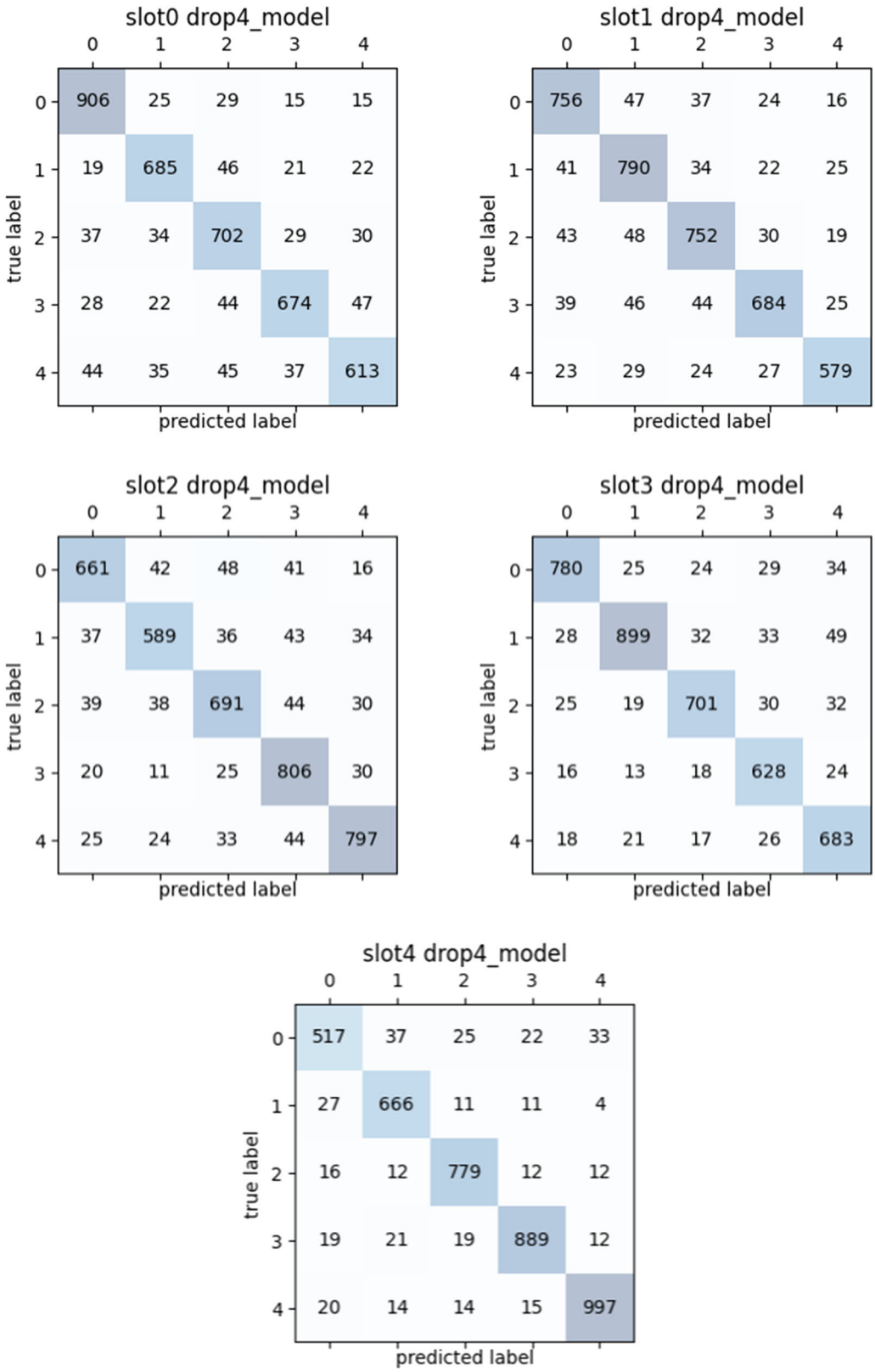


Fig. 10 Confusion matrices related to the different output slots for Neural Layout

Table 5 Accuracy of individual output slots related to Neural Layout extended with Ξ operator and a dropout layer for slot 4

	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4
Test accuracy	0.85	0.85	0.84	0.88	0.92

best mapping provided by Qiskit using at least one of the two routing algorithms. The random quantum circuits generated were 1000, with a number of CNOTs after the unrolling, less or equal to 50. Table 8 shows the average percentage values and relative standard deviation for the three cases above considered. It should be also emphasized that although the network has been trained through unrolled circuits with a maximum of 10 CNOTs, it manages to keep its performance almost constant even as the number of CNOTs within the circuits increases.

As further proof of the suitability of the proposed approach, a comparative analysis of running times of Neural Layout, Dense Layout, and Noise Adaptive Layout has been performed. This comparison was made by generating random quantum circuits of different depths and mapping these on the IBM Q Burlington processor by using the three above algorithms. For each depth value, 10 random circuits were generated and the mean value of the mapping time for each depth has been collected. These values are shown in Fig. 13 according to the mapping algorithm used.

As highlighted in the figure, the computational time taken by Neural Layout remains constant as the depth of the circuits increases. On the contrary, the times taken by Qiskit algorithms grow linearly with the depth. This result, combined with a good accuracy of Neural Layout, encourages the use of this mapping approach for larger circuits and processors. Indeed, the constant execution time means that the mapping of quantum circuits via Neural Layer can be a strong candidate to solve the problem of scalability that occurs in current mapping algorithms when applied large circuits to large processors.

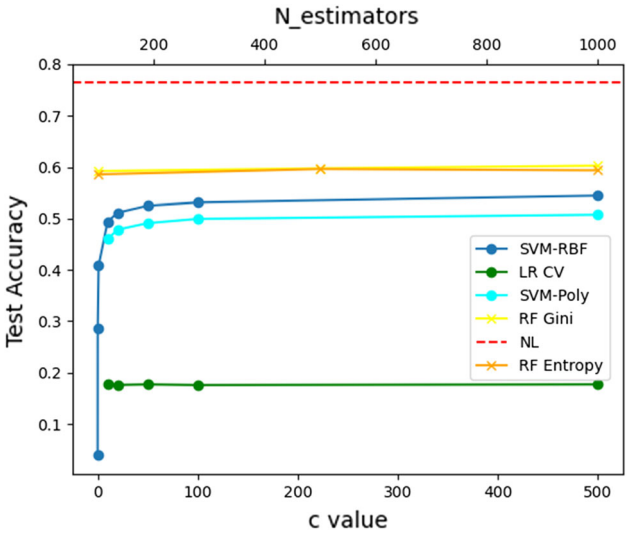


Fig. 11 Comparing test accuracy of Neural Layout (NL) and other machine learning techniques

6 Conclusions

This research introduced Neural Layout, the very first approach aimed at facing the quantum circuit mapping problem by means of machine learning techniques. In order to achieve this goal, the circuit mapping problem has been modelled as a conventional classification task and, successively, a deep neural network, characterized by a proper output layer, has been integrated with a repair operator capable of moving values computed by the neural network towards feasibility regions and allowing the proposed approach to work correctly. The performance yielded by Neural Layout has been compared to that obtained by other well-known machine learning techniques when applied to the circuit mapping problem and, in all cases, Neural Layout proved to be the most efficient one. Moreover, Neural Layout has been compared to two state-of-the-art algorithms for quantum circuit mapping belonging to IBM Qiskit and, in this case, Neural Layout showed similar performance in terms of mapping accuracy, but a considerable speedup in terms of running time, making the

Table 6 Hyperparameter setting for machine learning techniques

Classifier	Hyperparameters	
SVM	*Kernel Function*	*C Value*
	Rbf	[0.01, 0.1, 1.0, 10.0, 20.0, 50.0, 100.0, 500.0]
	Poly	[10.0, 20.0, 50.0, 100.0, 500.0]
LR-CV	*CV*	*C Value*
	5	[10.0, 20.0, 50.0, 100.0, 500.0]
RF	*Criterion*	*N Estimators*
	Gini	[100, 500, 1000]
	Entropy	[100, 500, 1000]

Table 7 Test Accuracy of output Slots for each classic classifier

Test accuracy					
Classifier	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4
SVM (Rbf)	0.69	0.74	0.70	0.73	0.82
SVM (Poly)	0.67	0.72	0.67	0.72	0.80
LR	0.43	0.60	0.30	0.47	0.66
RF (Gini)	0.72	0.76	0.75	0.75	0.84
RF (Entropy)	0.71	0.76	0.75	0.75	0.83

Table 8 Mean values and relative standard deviations of the percentages of 1000 random circuits mapped via DNN resulted in a final circuits less or equal deep to the final circuits obtained mapping the random circuit via Qiskit algorithms depending on the routing algorithm used

Routing algorithm	Mean	SD
LAS	0.582	0.239
SS	0.578	0.195
LAS or SS	0.757	0.168

proposed approach appropriate to be used in real quantum computing environments, such as IBM Q Experience.

Considering the results obtained by Neural Layout both in terms of quality and runtime of the mappings, it can be further developed to significantly impact the current state-of-the-art for the quantum circuit mapping algorithms. Indeed, the proposed DNN model has to be considered as a proof that machine learning can be used to address quantum circuit mapping problem opportunely modelled as a classification task, but further investigations on the optimization of the proposed model will be conducted by the authors in future studies.

This manuscript lays the foundation to a new mapping paradigm, and despite this approach has been tested on *ibmq_burlington* processor, the procedure to build and test models is completely independent from the processor used. Therefore, practically implementation of Neural Layout for other processors requires only the construction of proper datasets using the workflow proposed in Sect. 5.1. Moreover, the described way to collect data can be integrated with further deterministic mapping algorithms such as *SABRE Layout* [33] or better algorithms which will be developed in future, so as to increase the quality of the training data for classification models. The practical application of NL can be further supported by means of *online-training* techniques of the prediction models, which could be re-trained daily thanks to the calibration data provided simultaneously by IBM for each quantum processors.

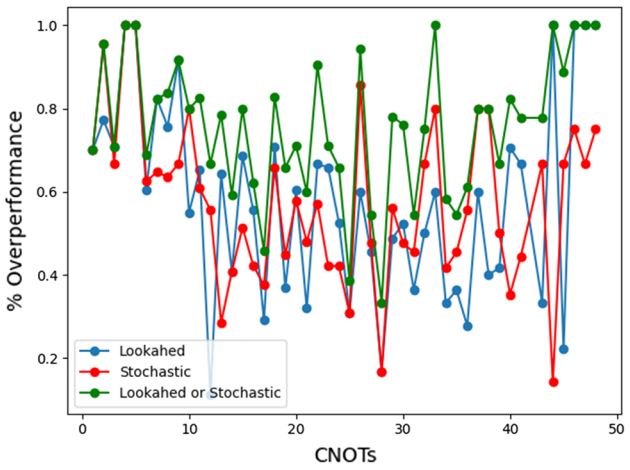


Fig. 12 Percentage of final circuits mapped by DNN resulting in a depth less than or equal to the depth resulted after the Qiskit best map

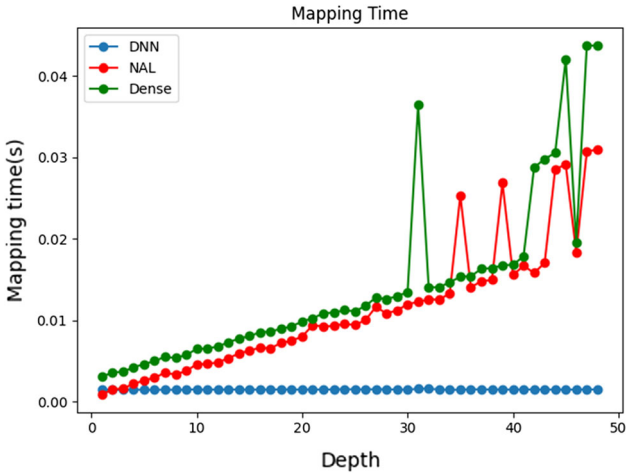


Fig. 13 Mapping time in second of each mapping algorithm as the depth of the circuit to map varies. A point in these plots is the average mapping time value of 10 circuits mapped

In the future, other important studies will be conducted on the possibility to embed NL in a recursive framework, where unsupervised machine learning techniques will be used to partition n-qubit circuits ($n > > 5$) in a collection of equivalent k-qubits circuits ($k \leq 5$) to be mapped to m-qubit quantum processors ($m > > 5$) so as to enable an efficient quantum circuit mapping on future generation of quantum computers.

Acknowledgements This research was conducted in the context of the Canada-Italy Innovation Award obtained by Prof. Giovanni Acampora in 2019.

Funding Open access funding provided by Università degli Studi di Napoli Federico II within the CRUI-CARE Agreement.

Declarations

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

References

1. Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput 26(5):1484–1509
2. Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp 212–219
3. Cao Y, Romero J, Olson JP, Degroote M, Johnson PD, Kieferová M, Kivlichan ID, Menke T, Peropadre B, Sawaya NPD et al (2019) Quantum chemistry in the age of quantum computing. Chem Rev 119(19):10856–10915
4. Lanyon BP, Whitfield JD, Gillett GG, Goggin ME, Almeida MP, Kassal I, Biamonte JD, Mohseni M, Powell BJ, Barbieri M et al (2010) Towards quantum chemistry on a quantum computer. Nat Chem 2(2):106–111
5. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature 549(7671):195–202
6. Schuld M, Sinayskiy I, Petruccione F (2015) An introduction to quantum machine learning. Contemp Phys 56(2):172–185
7. Dunjko V, Briegel HJ (2018) Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Rep Prog Phys 81(7):074001
8. Acampora G (2019) Quantum machine intelligence. Quantum Mach Intell 1(1):1–3
9. Venturelli D, Kondratyev A (2019) Reverse quantum annealing approach to portfolio optimization problems. Quantum Mach Intell 1(1):17–30
10. Zhao Z, Pozas-Kerstjens A, Rebentrost P, Wittek P (2019) Bayesian deep learning on a quantum computer. Quantum Mach Intell 1(1):41–51
11. Mengoni R, Di Pierro A (2019) Kernel methods in quantum machine learning. Quantum Mach Intell 1(3):65–71
12. Preskill J (2018) Quantum computing in the nisq era and beyond. Quantum 2:79
13. Maslov D, Falconer SM, Mosca M (2008) Quantum circuit placement. IEEE Trans Comput Aid Des Integr Circuits Syst 27(4):752–763
14. Chakrabarti A, Sur-Kolay S, Chaudhury A (2011) Linear nearest neighbor synthesis of reversible circuits by graph partitioning. arXiv:1112.0564
15. Shafaei A, Saeedi M, Pedram M (2013) Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In: 2013 50th ACM/EDAC/IEEE design automation conference (DAC). IEEE, pp 1–6
16. Shafaei A, Saeedi M, Pedram M (2014) Qubit placement to minimize communication overhead in 2d quantum architectures. In: 2014 19th Asia and South Pacific design automation conference (ASP-DAC). IEEE, pp 495–500
17. Wille R, Lye A, Drechsler R (2014) Optimal swap gate insertion for nearest neighbor quantum circuits. In: 2014 19th Asia and South Pacific design automation conference (ASP-DAC). IEEE, pp 489–494
18. Lye A, Wille R, Drechsler R (2015) Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In: The 20th Asia and South Pacific design automation conference. IEEE, pp 178–183
19. Venturelli D, Do M, Rieffel EG, Frank J (2017) Temporal planning for compilation of quantum approximate optimization circuits. In: IJCAI, pp 4440–4446
20. Venturelli D, Do M, Rieffel E, Frank J (2018) Compiling quantum circuits to realistic hardware architectures using temporal planners. Quantum Sci Technol 3(2):025004
21. Booth KEC, Do M, Beck JC, Rieffel E, Venturelli D, Frank J (2018) Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. arXiv:1803.06775
22. Oddi A, Rasconi R (2018) Greedy randomized search for scalable compilation of quantum circuits. In: International conference on the integration of constraint programming, artificial intelligence, and operations research. Springer, pp 446–461
23. Bhattacharjee D, Chattopadhyay A (2017) Depth-optimal quantum circuit placement for arbitrary topologies. arXiv:1703.08540
24. AlFailakawi M, Ahmad I, Hamdan S (2014) Lnn reversible circuit realization using fast harmony search based heuristic. In: Asia-Pacific conference on computer science and electrical engineering
25. Saeedi M, Wille R, Drechsler R (2011) Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Inf Process 10(3):355–377
26. Lin C-C, Sur-Kolay S, Jha NK (2014) Paqcs physical design-aware fault-tolerant quantum circuit synthesis. IEEE Trans Very Large Scale Integr VLSI Syst 23(7):1221–1234
27. Wille R, Keszocze O, Walter M, Rohrs P, Chattopadhyay A, Drechsler R (2016) Look-ahead schemes for nearest neighbor optimization of 1d and 2d quantum circuits. In: 2016 21st Asia and South Pacific design automation conference (ASP-DAC). IEEE, pp 292–297
28. Shrivastwa RR, Datta K, Sengupta I (2015) Fast qubit placement in 2d architecture using nearest neighbor realization. In: 2015 ieee international symposium on nanoelectronic and information systems. IEEE, pp 95–100
29. Kole A, Datta K, Sengupta I (2016) A heuristic for linear nearest neighbor realization of quantum circuits by swap gate insertion using *n*-gate lookahead. IEEE J Emerg Sel Topics Circuits Syst 6(1):62–72
30. Kole A, Datta K, Sengupta I (2017) A new heuristic for *n*-dimensional nearest neighbor realization of a quantum circuit. IEEE Trans Comput Aided Des Integr Circuits Syst 37(1):182–192
31. Bhattacharjee A, Bandyopadhyay C, Wille R, Drechsler R, Rahaman H (2018) A novel approach for nearest neighbor realization of 2d quantum circuits. In: 2018 IEEE computer society annual symposium on VLSI (ISVLSI). IEEE, pp 305–310
32. Paler A (2019) On the influence of initial qubit placement during nisq circuit compilation. In: International workshop on quantum technology and optimization problems. Springer, pp 207–217
33. Li G, Ding Y, Xie Y (2019) Tackling the qubit mapping problem for nisq-era quantum devices. In: Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems, pp 1001–1014

34. Murali P, Baker JM, Javadi-Abhari A, Chong FT, Martonosi M (2019) Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In: Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems, pp 1015–1029
35. Rosa JPS, Daniel, Guerra JD, Horta NCG, Martins R, Lourenco NCC (2020) Using artificial neural networks for analog integrated circuit design automation. Springer, Berlin
36. Wang F, Li Y-X (2006) Analog circuit design automation using neural network-based two-level genetic programming. In: 2006 international conference on machine learning and cybernetics. IEEE, pp 2087–2092
37. Li B, Franzon PD (2016) Machine learning in physical design. In: 2016 IEEE 25th conference on electrical performance of electronic packaging and systems (EPEPS), pp 147–150
38. Qi W, et al. (2017) IC design analysis, optimization and reuse via machine learning. North Carolina State University
39. Chakraborty M (2012) Artificial neural network for performance modeling and optimization of cmos analog circuits. arXiv:1212.0215
40. Chakma G, Awasthi S (2018) Resource optimization for circuit simulation using machine learning. In: 2018 IEEE international conference on big data (big data). IEEE, pp 4900–4905
41. Yih J-S, Mazumder P (1990) A neural network design for circuit partitioning. IEEE Trans Comput Aid Des Integr Circuits Syst 9(12):1265–1271
42. Aminian M, Aminian F (2007) A modular fault-diagnostic system for analog electronic circuits using neural networks with wavelet transform as a preprocessor. IEEE Trans Instrum Meas 56(5):1546–1554
43. Aminian F, Aminian M (2001) Fault diagnosis of analog circuits using Bayesian neural networks with wavelet transform as preprocessor. J Electron Test 17(1):29–36
44. Spence H (1996) Automatic analog fault simulation. In: Conference record. AUTOTESTCON'96. IEEE, pp 17–22
45. Catelani M, Gori M (1996) On the application of neural networks to fault diagnosis of electronic analog circuits. Measurement 17(2):73–80
46. Aminian F, Aminian M, Collins HW (2002) Analog fault diagnosis of actual circuits using neural networks. IEEE Trans Instrum Meas 51(3):544–550
47. Patel N (2020) Review on machine learning for analog circuit design. Int J Eng Tech Res 9:1024–1028
48. Li Y, Lin Y, Madhusudan M, Sharma A, Xu W, Sapatnekar S, Harjani R, Hu J (2020) Exploring a machine learning approach to performance driven analog ic placement. In: 2020 IEEE computer society annual symposium on VLSI (ISVLSI), pp 24–29
49. Kaye P, Laflamme R, Mosca M et al (2007) An introduction to quantum computing. Oxford University Press, Oxford
50. Mooney GJ, Hill CD, Hollenberg LCL (2019) Entanglement in a 20-qubit superconducting quantum computer. Sci Rep 9(1):1–8
51. Nielsen MA, Chuang I (2002) Quantum computation and quantum information: 10th anniversary edition. Cambridge University Press, USA. ISBN: 1107002176
52. Garcia-Escartin JC, Chamorro-Posada P (2011) Equivalent quantum circuits. arXiv:1110.2998
53. Botea A, Kishimoto A, Marinescu R (2018) On the complexity of quantum circuit compilation. In: Eleventh annual symposium on combinatorial search
54. Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: comparison of trends in practice and research for deep learning. arXiv:1811.03378