communications physics

ARTICLE

https://doi.org/10.1038/s42005-021-00684-3 OPEN

Quantum compiling by deep reinforcement learning

Lorenzo Moro[1,2], Matteo G. A. Paris[3], Marcello Restelli[1] & Enrico Prati[2✉]

The general problem of quantum compiling is to approximate any unitary transformation that describes the quantum computation as a sequence of elements selected from a finite base of universal quantum gates. The Solovay-Kitaev theorem guarantees the existence of such an approximating sequence. Though, the solutions to the quantum compiling problem suffer from a tradeoff between the length of the sequences, the precompilation time, and the execution time. Traditional approaches are time-consuming, unsuitable to be employed during computation. Here, we propose a deep reinforcement learning method as an alternative strategy, which requires a single precompilation procedure to learn a general strategy to approximate single-qubit unitaries. We show that this approach reduces the overall execution time, improving the tradeoff between the length of the sequence and execution time, potentially allowing real-time operations.

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy. [2] Istituto di Fotonica e Nanotecnologie, Consiglio Nazionale delle Ricerche, Milano, Italy. [3] Quantum Technology Lab, Dipartimento di Fisica Aldo Pontremoli, Università degli Studi di Milano, Milano, Italy. ✉email: enrico.prati@cnr.it

Quantum computation takes place at its lowest level by means of physical operations described by unitary matrices acting on the state of qubits. However, gate-model quantum computers may in practice provide just a limited set of transformations according to the constraints in their architecture[1–4]. Therefore, the computation is achieved as circuits of quantum gates, which are ordered sequences of unitary operators, acting on a few qubits at once[5]. Although the Solovay Kitaev theorem[6] ensures that any computations can be approximated, within an arbitrary tolerance, as a circuit based on a finite set of operators, there is no optimal strategy to establish how to compute such a sequence. The problem is known as quantum compiling and the algorithms to compute suitable approximating circuits as quantum compilers.

Every quantum compiler has its own trade-off between the length of the sequences, which should be as short as possible, the precompilation time, i.e., the time taken by the algorithm to be ready for use, and finally the execution time, i.e, the time the algorithm takes to return the sequence[7].

Previous works[7–10], mostly based on the Solovay–Kiteav theorem, addressed the problem by providing algorithms that return the approximating sequence with lengths and execution times that scale polylogarithmic as $\mathcal{O}(\log^{c}(1/\delta))$, where δ is the accuracy and c is a constant between 3 and 4. For instance, the Dawson–Nielsen (DNSK) formulation[11] provides sequences of length $\mathcal{O}(\log^{3.97}(1/\delta))$ in a time of $\mathcal{O}(\log^{2.71}(1/\delta))$. Additional performance gains can be achieved by selecting unique sets of quantum gates[7], reaching lengths that scale as $\mathcal{O}(\log^{\log(3)/\log(2)}(1/\delta))$ at the cost of increasing the precompilation time. Hybrid approaches involving a planning algorithm[12], in some cases boosted by deep neural networks[13], could achieve better performance. However, the planning algorithm raises the execution time, which could scale suboptimally for high accuracy. Despite the strategy considered, no algorithm can return the sequence using less than $\mathcal{O}(\log(1/\delta))$ gates, as shown in[9] by a geometrical proof. While existing quantum compilers are characterized by high execution and precompilation times[7,11], which make them impractical to compute during online operations, deep learning suggests an alternative approach.

Deep reinforcement learning is a subset of machine learning that exploits deep neural networks to learn optimal policies in order to achieve specific goals in decision-making problems[14–16]. Such techniques can be effective in high-dimensional control tasks and to address problems where limited or no prior knowledge of the configuration space of the system is available.

The fundamental assumptions and concepts in the reinforcement learning theory are built upon the idea of continuous interactions between a decision-maker called agent and a controlled system named environment, typically defined in the form of a Markov decision process[17] (MDP). According to a policy function that fully determines its behavior, the former interacts with the latter at discrete time steps, performing an action based on an observation related to the current state of the environment. Therefore, the environment evolves changing its state and returning a reward signal that can be interpreted as a measure of the adequateness of the action the agent has performed. The only purpose of the agent is to learn a policy to maximize the reward over time. The learning procedure can be a highly time-consuming task, but it has to be performed once. Then, it is possible to exploit the policy encoded in the deep neural network, with low computational resources in minimal time.

Recently, deep learning has been successfully applied to physics[18–21], where unprecedented advancements have been achieved by combining reinforcement learning[22] with deep neural networks into deep reinforcement learning (DRL). DRL, thanks to its ability to identify strategies for achieving a goal in complex configuration spaces without prior knowledge of the system[23–28], has recently been proposed for the control of quantum systems[15,18,29–33]. In this context, some of us previously applied deep reinforcement learning to control and initialize qubits by continuous pulse sequences[34,35] for coherent transport by adiabatic passage (CTAP)[36] and by digital pulse sequences for stimulated Raman passage (STIRAP)[37,38], respectively. Furthermore, it has proven effective as a control framework for optimizing the speed and fidelity of quantum computation[39] and in control of quantum gates[40].

In this work, we propose an approach to quantum compiling, exploiting deep reinforcement learning to approximate single-qubit unitary operators as circuits made by an arbitrary initial set of elementary quantum gates. As examples, we show how to steer quantum compiling for small rotations of $\pi/128$ around the three-axis of the Bloch sphere and for the Harrow–Recht–Chuang efficiently universal gates (HRC)[9], by employing two alternative DRL algorithms, depending on the nature of the base. After training, agents can generate single-qubit logic circuits within a tolerance of 0.99 average-gate fidelity (AGF). The adopted strategy for training the agent consists of generating a uniform distribution of single-qubit unitary matrices, where to sample the training targets. The agents are not told how to approximate such targets, but instead, they are asked to establish a suitable policy to complete the task. The agents' final performance is then measured using a validation set of unitary operators not previously seen by the agent.

To summarize, the DRL agents learn a policy to approximate single-qubit unitary transformation at the cost of a precompilation procedure, which is done only once. The method is effective for both sets of small-angle rotations and sparse sets of unitary operators. Average gate fidelity achieves $\varepsilon = 0.9999$ in the best cases for small rotations, for which the execution time empirically scales as $\mathcal{O}(\log^{1.25}(1/(1-\varepsilon)))$. Although the method does not guarantee finding the solution, it has the advantage of operating independently from the specific hardware and that its speed would enable real-time computing.

Results and discussion

Deep reinforcement learning as quantum compiler. The quantum compilation is a fundamental problem in the quantum computation theory, consisting of approximating any unitary transformation as a finite sequence of unitary operators A_j is chosen from a universal set of gates $\mathcal{B}$.

In this work, we ask the agent to approximate any single-qubit unitary matrix $\mathcal{U}$, within a fixed tolerance ε. Therefore, the goal of the agent is to find a unitary matrix $U_n = \prod_{j=1}^{n} A_j$, resulting from the composition of the elements in the sequence, that is sufficiently close to $\mathcal{U}$. Although the DRL framework allows exploiting any distance between matrices to evaluate the accuracy of the solutions, the average gate fidelity is wildly used for the purpose, mainly due to the modest computational demands needed to compute it. Alternative choices are possible, such as the diamond norm[41,42].

In the framework of quantum compiling, the environment consists of a quantum circuit that starts as the identity at the beginning of each episode. It is built incrementally at each time step by the agent, choosing a gate from $\mathcal{B}$ according to the policy π encoded in the deep neural network, as shown in Fig. 1. Therefore, the available actions that the agent can perform correspond to the gates in the base $\mathcal{B}$.

The observation used as input at time step n corresponds to the vector of the real and imaginary parts of the elements of the matrix O_n, where $\mathcal{U} = U_n \cdot O_n$. Such representation encodes all the information needed by the agent to build a suitable approximating sequence of gates, i.e., the current composition

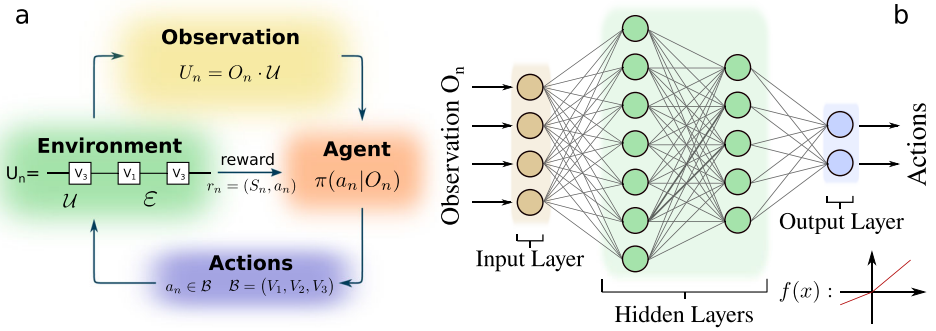

Fig. 1 The deep reinforcement learning (DRL) architecture. a The DRL environment can be described as a quantum circuit modeled by the approximating sequence U_n, the fixed tolerance ε, and the unitary target to approximate $\mathcal{U}$, that generally changes at each episode. At each time step n, the agent receives the current observation O_n and based on that information, it chooses from the base $\mathcal{B}$ the next gate an to apply on the quantum circuit. Therefore, the environment returns the real-valued reward r_n to the agent, which is a function of the state S_n and the action a_n. **b** The policy π of the agent is encoded in a deep neural network (DNN). The policy of the agent is encoded in a deep neural network. At each time-step, the DNN receives as input a vector made by the real and imaginary parts of the observation O_n. Such information is processed by the hidden layers and returned through the output layer. The neurons in the output layer are associated with the action the agent will perform in the next time step. In the bottom-right corner is reported an example of the nonlinear activation function, i.e., the rectified linear unit function RELU.

of gates and the unitary target to approximate. No information on the tolerance is given to the agent since it is fixed and thus it can be learned indirectly during the training.

Designing a suitable reward function is challenging, potentially leading to unexpected or unwanted behavior if not defined accurately. Therefore, two reward functions have been designed, depending on the different characteristics of the gate base considered, which can be identified as quasi-continuous-like sets of small rotations and discrete sets. The former are inspired by gates available on superconductive and trapped ions architecture[1,2,4], where the latter is the standard set of logic gates, typically used to write quantum algorithms, e.g., the Clifford+T library[43,44]. Both reward functions are negative at each time step, so that the agent will prefer shorter episodes.

In this work, we exploit Deep Q-Learning (DQL)[45] and Proximal Policy Optimization (PPO)[46] algorithms to train the agents, depending on the reward function. Such algorithms differ in many aspects, as described in Supplementary Note 1. The former is mandatory for the case of sparse reward, since such reward requires off-policy methods to be exploited, while the latter has been chosen for its robustness and tunability. More details on the rewards are given in Supplementary Note 2.

Training neural networks for approximating a single-qubit gate. To demonstrate the exploitation of DRL as a quantum compiler, we first considered the problem of decomposing a single-qubit gate $\mathcal{U}$, into a circuit of unitary transformations that can be implemented directly on quantum hardware. The base of gates corresponds to six small rotations of $\pi/128$ around the three-axis of the Bloch sphere, i.e., $\mathcal{B} = (R_{\hat{x}}(\pm\frac{\pi}{128}),\ R_{\hat{y}}(\pm\frac{\pi}{128}),\ R_{\hat{z}}(\pm\frac{\pi}{128}))$.

It is essential to choose the tolerance ε and the fixed target accurately to appreciate the learning procedure. The former should be small enough and the latter sufficiently far from the identity not to be solved by chance. However, if the target is too difficult to approximate, the agent will fail and no learning occurs. To be sure that at least one solution does exist, we build $\mathcal{U}$ as a composition of 87 elements selected from $\mathcal{B}$. The resulting unitary target is

$$\mathcal{U} = \begin{pmatrix} 0.76749896 - 0.43959894\mathrm{i} & -0.09607122 + 0.45658344\mathrm{i} \\ 0.09607122 + 0.45658344\mathrm{i} & 0.76749896 + 0.43959894\mathrm{i} \end{pmatrix}. \quad (1)$$

Table 1 List of the hyperparameters and their values used in the fixed-target problem. We used the scaled exponential linear units (SELU) as nonlinear activation functions in the hidden layers of the neural network.

Area	Hyperparameter	Value
Neural network	# hidden layers	128, 128
	activations	SELU, SELU, linear
	initializers	lecun, lecun, glorot
Training	optimizer	Adam
	learning rate	0.0005
	batch size	10^3
	training frequency	every one episode
Algorithm	epsilon decay	0.99976
	memory size	10^4 experiences
	max length episode	130

We tested a nonlearning agent that acts randomly to ensure not to deal with a trivial task, setting the tolerance at 0.99 average gate fidelity and limiting the maximum length of the episode at 130. No solution was found after 10^4 episodes. Then, the problem was addressed by exploiting a DQL agent, using the same thresholds for the tolerance and the length of the episode. We exploited the dense reward function

$$r(S_n, a_n) = \begin{cases} (L - n) + 1 & \text{if } d(U_n, \mathcal{U}) < \varepsilon \\ -d(U_n, \mathcal{U})/L & \text{otherwise} \end{cases} \quad (2)$$

where L is the maximum length of the episode, a_n, S_n, and $d(U_n, \mathcal{U})$ are the action performed, the state of the environment, and the distance between the target and the approximating sequence at time n, respectively. Such reward performs adequately if small rotations are used as base only.

Table 1 reports some additional information about the network architecture and the hyperparameter set, while Fig. 2 shows the performance and the solutions found by the agent during the training time. The agent learns how to approximate the target after about 10^4 episodes, while improving the solution over time. At the end of the learning, the agent discovered an approximating circuit made by 76 gates only, within the target tolerance.

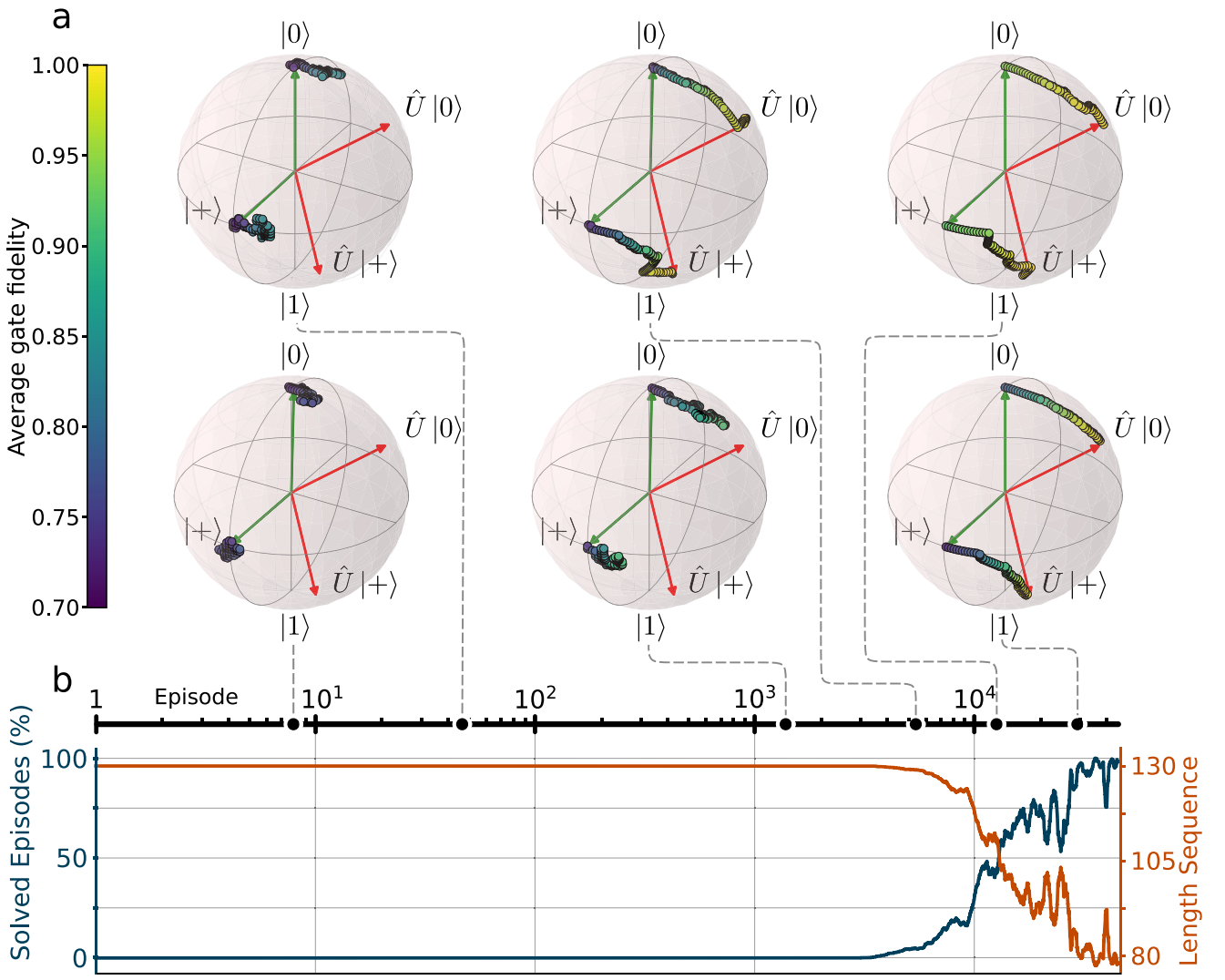

Fig. 2 The deep reinforcement learning agent learns how to approximate a single-qubit gate. a Best sequences of gates discovered by the agent during the training at different epochs. The dashed lines connecting the Bloch spheres to the Episode axis indicate the episode at which the sequences were found for the first time. Each approximating sequence is represented by two trajectories of states (colored points) on the Bloch sphere. They are obtained by applying the unitary transformations associated with the circuit at the time step n on two representative states, namely $|0\rangle$ and $|+\rangle$ respectively. The agent is asked to transform the starting state (green arrows) in the corresponding ending state (red arrows), i.e., $|0\rangle$ to $\mathcal{U}|0\rangle$ and $|+\rangle$ to $\mathcal{U}|+\rangle$ respectively, where $\mathcal{U}$ corresponds to the unitary target. **b** Performance of the agent during training. The plot represents the percentage of episodes for which the agent was able to find a solution (blue line) and the average number of the sequence of gates (orange line). The agent learns how to approximate the target after about 104 episodes and then improves the solution over time.

Quantum compiling by rotation operators. The DRL approach can be generalized to the quantum compilation of a larger class of unitary transformations. Instead of limiting to approximating one matrix only, we aim at exploiting the knowledge of a trained agent to approximate any single-qubit unitary transformation, without requiring additional training. Therefore, we used as training targets Haar unitary matrices, since they form an unbiased and a general data set which is ideal to train neural networks, as described in the "Methods" section. If additional information on the type and distribution of targets is known, it is possible to choose a different set of gates for training, potentially increasing the performance of the agent as described in Supplementary Note 3.1.

Such task is tougher to solve compared with the fixed-target problem. Therefore, we exploited the Proximal Policy Optimization algorithm (PPO)[46] being more robust and easy to tune than DQL. We fixed the tolerance ε at 0.99 AGF and limited the maximum length of the approximating circuits (time step per episode) at 300 gates, as reported in Table 2. Figure 3b shows the performance of the agent during the training time (blue lines). The agent starts to approximate unitaries after 10^5 episodes, but it requires much more time to achieve satisfactory performance.

We tested the performance of the agents at the end of the learning, using a validation set of 10^6 Haar unitary targets. The agent is able to approximate more than 96% of the targets within the tolerance requested. Complete results are reported in Table 3.

Quantum compiling by the HRC efficiently universal base of gates. In order to fully exploit the power of DRL, we now turn to the problem of compiling single-qubit unitary matrices using a base of discrete gates. The agent can perform the set of HRC efficiently universal base of gates[9]:

$$V_1 = \frac{1}{\sqrt{5}}\begin{pmatrix} 1 & 2\mathrm{i} \\ 2\mathrm{i} & 1 \end{pmatrix} V_2 = \frac{1}{\sqrt{5}}\begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} V_3 = \frac{1}{\sqrt{5}}\begin{pmatrix} 1+2\mathrm{i} & 0 \\ 0 & 1-2\mathrm{i} \end{pmatrix} \quad (3)$$

Such unitary matrices implement quantum transformations that are very different from the ones performed by small rotations. The agent has to learn how to navigate in the high-dimensional space of unitary matrices, exploiting counterintuitive movements that could lead it close to the target at the last time step of the episode only. Therefore, the dense reward function (2) is no longer useful to guide the agent toward the targets. We exploited a "sparse" reward (binary reward):

$$r(S_n, a_n) = \begin{cases} 0 & \text{if } d(U_n, \mathcal{U}) < \varepsilon \\ -1/L & \text{otherwise.} \end{cases} \quad (4)$$

Such function lowers the reward of the agent equally for every action it takes, bringing no information to the agent on how to find the solution. Therefore, it requires advanced generalization techniques to be effective, such as Hindsight Experience Replay (HER)[47]. Since HER requires an off-policy reinforcement learning algorithm, we chose the DQL agent to address the problem using the hyperparameters reported in Table 4.

We fixed the tolerance ε at 0.99 AGF and limited the maximum length of the approximating circuits at 130 gates. Figure 3 shows the performance of the agent during the training time (orange lines) and the length distribution of the solved sequences obtained using a validation set of 10^6 Haar random unitaries. Although the agent receives a noninformative reward signal at each time-step, it surprisingly succeeds to solve roughly more than 95% of the targets, using on average less than 36 gates as reported in Table 3. It is worth noting that the agent can build significantly shorter circuits, compared with the case of rotation matrices. It is not unexpected, since the HRC base allows to explore the space of unitary matrices quite efficiently.

Table 2 List of the hyperparameters and their values used in the problem of quantum compiling by rotations operators. The proximal policy optimization agent (PPO) exploits scaled exponential linear units (SELU) as nonlinear activation functions in the hidden layers of the neural network.

Area	Hyperparameter	Value
Neural network	# hidden layers	128, 128
	activations	SELU, SELU
Training	learning rate	0.0001
	batch size	128
	# agents	40
Algorithm	max length episode	300

Performances of the DRL quantum compiler. Our results show that deep reinforcement learning-based quantum compilers can approximate single-qubit gates by a set of quantum gates without prior knowledge. We now turn to the evaluation of the performances demonstrated by our method.

We point out that our method differs from existing quantum compilers for its flexibility since it can be applied to any basis. Indeed, Y–Z–Y gate decomposition can only manage a basis consisting of y and z rotations, while KAK decomposition[48] is limited to two-qubits and CNOT and y and z rotations. Machine-learning methods based on A*[49] algorithms could suffer from high execution time that could scale suboptimally for high accuracy[12,13]. Instead of designing a tailored quantum compiling algorithm, we exploited a DRL agent to learn a general strategy to approximate single-qubit unitary matrices and store it within an artificial neural network.

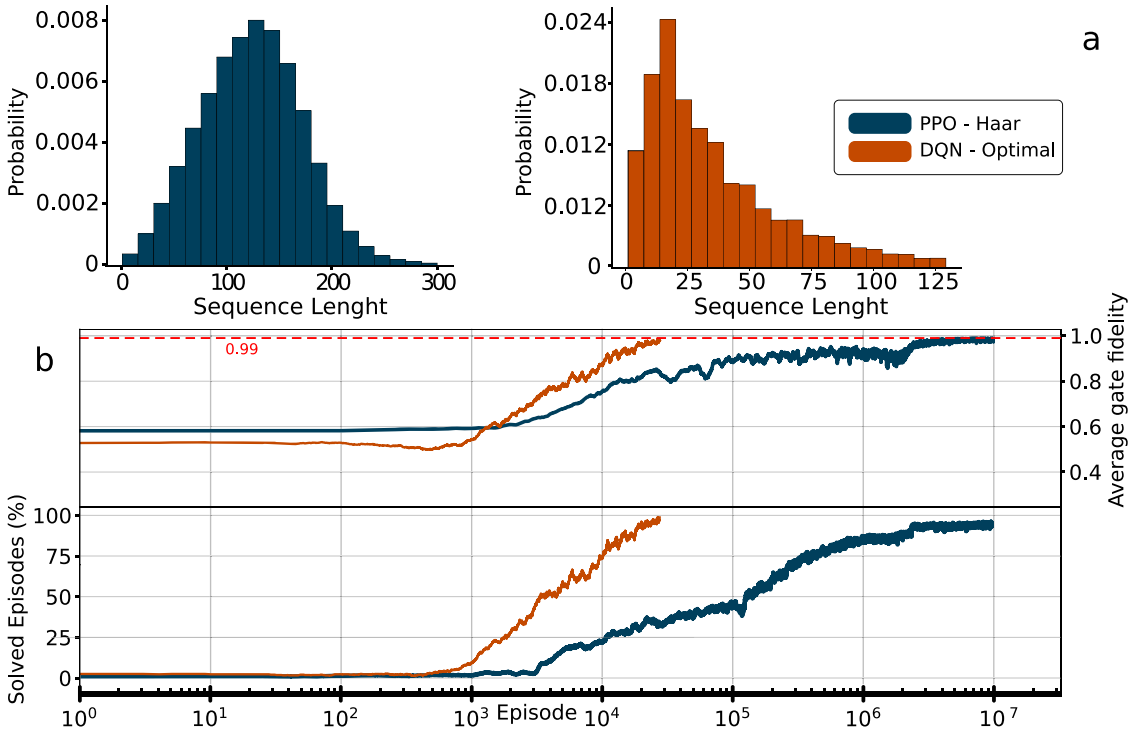

Fig. 3 Deep reinforcement learning agents learn how to approximate single-qubit unitaries using different base of gates. A proximal policy-optimization agent (PPO) (blue color) and a deep Q-learning hindsight-experience replay agent DQL+HER (orange color) were trained to approximate single-qubit unitaries using two different bases of gates, i.e., six small rotations of $\pi/128$ around the three-axis of the Bloch sphere and the Harrow–Recht–Chuang efficient base of gates (HRC), respectively. The tolerance was fixed to 0.99 average gate fidelity. **a** The length distributions of the gates sequences discovered by the agents at the end of the learning. The HRC base generates shorter circuits as expected. **b** Performance of the agent during training on the tasks.

Table 3 Performance of the proximal policy optimization (PPO) and deep Q-learning (DQL) agents in approximating Haar unitary matrices. The performances are measured after the training procedures over a validation set of 10^6 targets. We exploit the Harrow-Recht-Chuang efficient base of gates (HRC) and the rotations gates. The 95th and 95th columns refer to the 95th and 95th percentile of the distribution of the length of the solved sequences.

Base	Solved (%)	Mean length	95th percentile	99th percentile
HRC	95.0	35	94	120
Rotations	96.4	124	204	245

Table 4 List of the hyperparameters and their values used in the Harrow-Recht-Chuang efficient base of gates (HRC) problem. The deep Q-learning agent employs scaled exponential linear units (SELU) as nonlinear activation functions in the hidden layers of the neural network.

Area	Hyperparameter	Value
Neural network	# hidden layers	128, 128
	activations	SELU, SELU, linear
	initializers	lecun, lecun, glorot
Training	optimizer	Adam
	learning rate	0.0001
	batch size	200
	training frequency	every one episode
Algorithm	epsilon decay	0.99931
	memory size	$5 \cdot 10^5$ experiences
	max length episode	130

One of the critical questions to consider when measuring the performance of a quantum compiler is how a classical computer can efficiently return the sequence of gates. Inefficient strategies[50] can neutralize any quantum advantage over classical counterparts if the execution time and the length of the sequence scale suboptimally with the accuracy.

Our DRL quantum compiler can produce sequences that length scales as $\mathcal{O}(\log^{1.25}(1/\delta))$, as demonstrated by empirically measuring the performance on a specific task and shown in Fig. 4. Although such an approach has no guarantee to return a suitable solution, DRL quantum compilers can return solutions after a precompilation procedure to be performed once, improving the execution time and potentially enabling online quantum compilation. In fact, by writing the policy into a deep neural network, the execution time depends on the complexity of the network and the episode length only. Therefore, it scales proportionally to the sequence length, i.e., as $\mathcal{O}(\log^{1.25}(1/\delta))$. At the end of the training procedure, the agent returns the whole approximating sequence in a fraction of a second ($5.4 \cdot 10^{-4}s$ per time step) on a single CPU core. The speed-up gain could be further enhanced by reducing the size of the neural networks and being easily parallelizable by exploiting specialized hardware to run them, such as GPUs or Tensor Processing Unit[51].

As examples, we trained and employed two deep reinforcement learning quantum compilers to build quantum circuits within a final tolerance of 0.99 AGF, using two different sets of quantum logic gates. We accounted for the diverse characteristics of the bases designing a dense and a sparse reward function. We addressed Haar distributed unitary matrices as targets to be as general as possible, but if additional information on the targets is available, it is possible to achieve higher tolerance without fine-tuning neural network architectures or the RL hyperparameters, as shown in Supplementary Note 3.1.

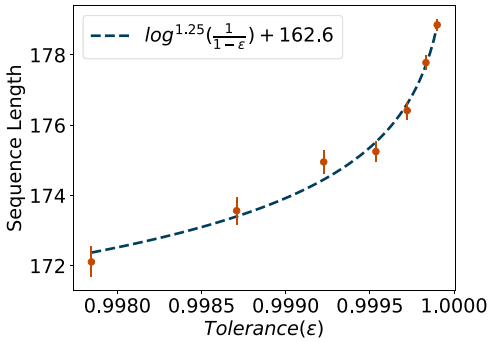

Fig. 4 Relation between sequence length and tolerance. Each data point is obtained by averaging the length of the approximating sequence of gates found by a trained agent using a validation set of 10^7 unitary targets. The error bars report the standard deviation. The agent was trained to achieve a final tolerance of 0.9999 average gate fidelity (AGF). The targets are built as compositions of small rotations around the three axes of the Bloch sphere, as described in Supplementary Note 3.1. The data are fitted by a polylogarithmic function (dashed blue line) with $R^2 = 0.986$ and $RMSE = 0.26$.

Our method could be employed in larger qubit spaces, as shown by an early prototype in Supplementary Note 3.2. The DRL compiler can approximate two-qubit logic gates with consistent performance compared with the one-qubit gates of 0.99 AGF. Supplementary Note 4 reports examples of single and two-qubit circuits discovered by the DRL quantum compilers.

As a concluding remark, we observe that this approach can be specialized, taking into account any hardware constraints that limit operations, integrating them directly into the environments. These tests, as well as the extension of this approach to n-qubits, will be the objective of future work.

Methods

Generation of Haar random unitary matrices. The strategy used to generate the training data set should be chosen opportunely, depending on the particular set of gates of interest, since deep neural networks are very susceptible both to the range and the distribution of the inputs. Therefore, Haar random unitaries have been used as training targets. Pictorially, picking a Haar unitary matrix from the space of unitaries can be thought as choosing a random number from a uniform distribution[52]. More precisely, the probability of selecting a particular unitary matrix from some region in the space of all unitary matrices is directly proportional to the volume of the region itself. Such matrices form an unbiased data set that is ideal to train neural networks.

Learning by HER. Many RL problems may be efficiently addressed employing sparse rewards only, since engineering efficient and well-shaped reward functions can be extremely challenging. Such rewards are binary, i.e., the agent receives a constant signal until it achieves the goal. However, if the agent gets the same reward almost every time, it cannot learn any relationships of cause and effect that its actions have on the environment. Therefore, it might take an extremely long time to learn something, if anything at all.

HER is a technique introduced by OPENAI that allows to mitigate the sparse-reward problem[47]. The basic idea of HER is to exploit the ability that humans have to learn from failure. Specifically, even if the agent always failed to solve the task, it can reach different objectives. Exploiting this information, it is possible to train the agent to reach different targets. Although the agent receives a reward signal to achieve a distinct goal from the original one, this procedure, if iterated, can help the agent to learn how to generalize the policy to reach the primary task we want to solve.

The implementation of HER in the Q-learning algorithm is straightforward. After an entire episode is completed, the experiences associated with that episode are modified selecting a new goal. Then, the q-function is updated as usual. There are several strategies to choose the goals[47]. We designed a strategy to select the new goals, consisting in randomly selecting k-percent of the states that come from the same episode.

Average gate fidelity. The average fidelity $\bar{F}(\mathcal{U}, U)$ between two gates $\mathcal{U}$ and U is defined by

$$\bar{F}(\mathcal{U}, U) = \int \langle\psi|\mathcal{U}^\dagger U|\psi\rangle\langle\psi|U^\dagger\mathcal{U}|\psi\rangle d\psi, \qquad \int d\psi = 1 \tag{5}$$

where the integral is over all the state spaces using a Haar measure.

Neural network architectures. The architecture of the deep neural network directly affects the performance of the DRL agent. However, choosing the optimal architecture is a trial-and-error task and can be an exceptionally time-consuming procedure, since it depends on the specific problem the agent is addressing. Therefore, in this work, we did not focus on the optimization, but on finding the smallest neural network architecture that can lead to satisfactory performance. We started with one hidden layer only and a few neurons, gradually increasing the depth and the width of the network. We found that a relatively small architecture made by two hidden layers of 128 neurons is sufficient to achieve the tasks.

Software and hardware. All the code in this work was developed using Python language. The Stable Baseline[53] library has been employed for the implementation of PPO agent only. Most of the simulation has been run by using GNU parallel[54] on an Intel Xeon W-2195 and a Nvidia GV100.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Code availability

The code and the algorithm used in this study are available from the corresponding author upon reasonable request.

Received: 22 February 2021; Accepted: 20 July 2021;
Published online: 06 August 2021

References

1. Linke, N. M. et al. Experimental comparison of two quantum computing architectures. *Proc. Natl Acad. Sci. USA* **114**, 3305–3310 (2017).
2. Maslov, D. Basic circuit compilation techniques for an ion-trap quantum machine. *New J. Phys.* **19**, 023035 (2017).
3. Leibfried, D., Knill, E., Ospelkaus, C. & Wineland, D. J. Transport quantum logic gates for trapped ions. *Phys. Rev. A* **76**, 032324 (2007).
4. Debnath, S. et al. Demonstration of a small programmable quantum computer with atomic qubits. *Nature* **536**, 63 (2016).
5. Maronese, M. & Prati, E. A continuous rosenblatt quantum perceptron. *Int. J. Quantum Inf.* https://doi.org/10.1142/S0219749921400025 (2021).
6. Kitaev, A. Y. Quantum computations: algorithms and error correction. *Russian Math. Surv.* **52**, 1191–1249 (1997).
7. Zhiyenbayev, Y., Akulin, V. & Mandilara, A. Quantum compiling with diffusive sets of gates. *Phys. Rev. A* **98**, 012325 (2018).
8. Barenco, A. et al. Elementary gates for quantum computation. *Phys. Rev. A* **52**, 3457 (1995).
9. Harrow, A. W., Recht, B. & Chuang, I. L. Efficient discrete approximations of quantum gates. *J. Math. Phys.* **43**, 4445–4451 (2002).
10. Kitaev, A. Y., Shen, A., Vyalyi, M. N. & Vyalyi, M. N. *Classical and quantum computation*. 47 (American Mathematical Soc., 2002).
11. Dawson, C. M. & Nielsen, M. A. The solovay-kitaev algorithm. *Quantum Info. Comput.* **6**, 81–95 (2006).
12. Davis, M. G. et al. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 223–234 (IEEE, 2020).
13. Zhang, Y.-H., Zheng, P.-L., Zhang, Y. & Deng, D.-L. Topological quantum compiling with reinforcement learning. *Phys. Rev. Lett.* **125**, 170501 (2020).
14. Tognetti, S., Savaresi, S. M., Spelta, C. & Restelli, M. In *2009 IEEE Control Applications,(CCA) & Intelligent Control (ISIC)*, 582–587 (IEEE, 2009).
15. Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. Universal quantum control through deep reinforcement learning. *npj Quantum Inf.* **5**, 1–8 (2019).
16. Castelletti, A., Pianosi, F. & Restelli, M. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resour. Res.* **49**, 3476–3486 (2013).
17. Sutton, R. S., Barto, A. G. et al. *Introduction to reinforcement learning*, vol. 135 (MIT press Cambridge, 1998).
18. Fösel, T., Tighineanu, P., Weiss, T. & Marquardt, F. Reinforcement learning with neural networks for quantum feedback. *Phys. Rev. X* **8**, 031084 (2018).
19. Dunjko, V. & Briegel, H. J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Rep. Prog. Phys.* **81**, 074001 (2018).
20. Sarma, S., Deng, D.-L. & Duan, L.-M. Machine learning meets quantum physics. *Phys. Today* **72**, 48–54 (2019).
21. Carleo, G. et al. Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
22. Sutton, R. S., Barto, A. G. et al. *Reinforcement learning: An introduction* (MIT press, 1998).
23. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).
24. Melnikov, A. A. et al. Active learning machine learns to create new quantum experiments. *Proc. Natl Acad. Sci.* **115**, 1221–1226 (2018).
25. Nautrup, H. P., Delfosse, N., Dunjko, V., Briegel, H. J. & Friis, N. Optimizing quantum error correction codes withreinforcement learning. *Quantum* **3**, 215 (2019).
26. Sweke, R., Kesselring, M. S., van Nieuwenburg, E. P. & Eisert, J. Reinforcement learning decoders for fault-tolerantquantum computation. *Mach. Learn. Sci. Technol.* **2**, 025005 (2020).
27. Reddy, G., Celani, A., Sejnowski, T. J. & Vergassola, M. Learning to soar in turbulent environments. *Proc. Natl Acad. Sci. USA* **113**, E4877–E4884 (2016).
28. Colabrese, S., Gustavsson, K., Celani, A. & Biferale, L. Flow navigation by smart microswimmers via reinforcement learning. *Phys. Rev. Lett.* **118**, 158004 (2017).
29. August, M. & Hernández-Lobato, J. M. Taking gradients through experiments: Lstms and memory proximal policy optimization for black-box quantum control. In *International Conference on High Performance Computing*, 591–613 (Springer, 2018).
30. Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. Universal quantum control through deep reinforcement learning. *npj Quantum Inf.*, **5**, 33 (2019).
31. Albarrán-Arriagada, F., Retamal, J. C., Solano, E. & Lamata, L. Measurement-based adaptation protocol with quantum reinforcement learning. *Phys. Rev. A* **98**, 042315 (2018).
32. Andreasson, P., Johansson, J., Liljestrand, S. & Granath, M. Quantum error correction for the toric code using deep reinforcement learning. *Quantum* **3**, 183 (2019).
33. Prati, E. Quantum neuromorphic hardware for quantum artificial intelligence. *J. Phys. Conf. Ser*, **880**, 012018 (2017).
34. Porotti, R., Tamascelli, D., Restelli, M. & Prati, E. Coherent transport of quantum states by deep reinforcement learning. *Commun. Phys.* **2**, 61 (2019).
35. Porotti, R., Tamascelli, D., Restelli, M. & Prati, E. Reinforcement learning based control of coherent transport by adiabatic passage of spin qubits. *J. Phys. Conf. Ser.* **1275**, 012019 (2019).
36. Ferraro, E., De Michielis, M., Fanciulli, M. & Prati, E. Coherent tunneling by adiabatic passage of an exchange-only spin qubit in a double quantum dot chain. *Phys. Rev. B* **91**, 075435 (2015).
37. Paparelle, I., Moro, L. & Prati, E. Digitally stimulated Raman passage by deep reinforcement learning. *Phys. Lett. A* 384, 126266 (2020).
38. Moro, L., Paparelle, I. & Prati, E. Using deep learning for digitally controlled STIRAP. *Int. J. Quantum Inf.* https://doi.org/10.1142/S0219749921410021 (2021).
39. Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. Universal quantum control through deep reinforcement learning. *npj Quantum Inf.* **5**, 1–8 (2019).
40. An, Z. & Zhou, D. Deep reinforcement learning for quantum gate control. *EPL* **126**, 60002 (2019).
41. Aharonov, D., Kitaev, A. & Nisan, N. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, 20–30 (Association for Computing Machinery, 1998).
42. Watrous, J. Semidefinite programs for completely bounded norms. *Theory Comput.* **5**, 217–238 (2009).
43. Nielsen, M. & Chuang, I. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2002). https://books.google.it/books?id=xnI9PgAACAAJ.
44. Tolar, J. In *Journal of Physics: Conference Series*, vol. 1071, 012022 (IOP Publishing, 2018).
45. Watkins, C. J. & Dayan, P. Q-learning. *Mach. Learn.* **8**, 279–292 (1992).
46. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. CoRRabs/1707.06347 (2017).
47. Andrychowicz, M. et al. In *Advances in Neural Information Processing Systems*, 5048–5058 (NIPS, 2017).
48. Vatan, F. & Williams, C. Optimal quantum circuits for general two-qubit gates. *Phys. Rev. A* **69**, 032315 (2004).
49. Hart, P. E., Nilsson, N. J. & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **4**, 100–107 (1968).
50. Lloyd, S. Almost any quantum logic gate is universal. *Phys. Rev. Lett.* **75**, 346 (1995).

51. Dean, J. & Hölzle, U. Build and train machine learning models on our new google cloud TPUs, 2017. https://www.blog.google/topics/google-cloud/google-cloud-offer-tpus-machine-learning (2017).
52. Russell, N. J., Chakhmakhchyan, L., O'Brien, J. L. & Laing, A. Direct dialling of haar random unitary matrices. *New J. Phys.* **19**, 033007 (2017).
53. Hill, A. et al. Stable baselines. https://github.com/hill-a/stable-baselines (2018).
54. Tange, O. Gnu parallel-the command-line power tool. *The USENIX Magazine* **36**, 42–47 (2011).

Acknowledgements

L.M. and E.P. gratefully thank Vista Technology SRL for having partially supported this research. E.P. gratefully acknowledges the support of NVIDIA Corporation for the donation of the Titan Xp GPU used for this research.

Author contributions

L.M. wrote all the codes and performed the experiments, M.P. contributed to the quantum mechanical environment of the RL agent, M.R. contributed to the development of the RL agents, and E.P. conceived and coordinated this research. All the authors contributed to discuss the results and to the writing of the paper.

Competing interests

Lorenzo Moro, Enrico Prati, Marcello Restelli, have submitted an application for patent of a computed implemented method for real time quantum compiling based on artificial intelligence. Application number \num{102021000006179}, 16 March 2021 Italy. Matteo G.A. Paris declare declares that he has no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-021-00684-3.

Correspondence and requests for materials should be addressed to E.P.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.