Quantum circuit synthesis with diffusion models

Anonymous Authors[1]

Abstract

Quantum computing has recently emerged as a transformative technology. Yet, its promised advantages rely on efficiently translating quantum operations into viable physical realizations. In this work, we use generative machine learning models, specifically denoising diffusion models (DMs), to facilitate this transformation. Leveraging text-conditioning, we steer the model to produce desired quantum operations within gate-based quantum circuits. Notably, DMs allow to sidestep during training the exponential overhead inherent in the classical simulation of quantum dynamics—a consistent bottleneck in preceding ML techniques. We demonstrate the model's capabilities across two tasks: entanglement generation and unitary compilation. The model excels at generating new circuits and supports typical DM extensions such as masking and editing to, for instance, align the circuit generation to the constraints of the targeted quantum device. Given their flexibility and generalization abilities, we envision DMs as pivotal in quantum circuit synthesis, enhancing both practical applications but also insights into theoretical quantum computation.

Quantum computing is considered as a groundbreaking technology, holding promise across various domains, from fundamental research in physics (Feynman et al., 2018) or chemistry (McArdle et al., 2020), to machine learning (Cerezo et al., 2022) and optimization (Farhi et al., 2014). While the technological realization of quantum processors is advancing, the current noisy intermediate-scale quantum (NISQ) era (Preskill, 2018) presents challenges due to device limitations, making it essential to design efficient quantum circuits tailored to the available resources and gate sets, requiring further research and expertise in finding optimal gate ansätze and methods for circuit construction.

This task, commonly known as quantum circuit synthesis, involves the following: given a target quantum state that the quantum processor aims to produce as output, one must design a sequence of basic elements, called quantum gates, that generate that quantum state from a fiducial / standard input state. Similarly, instead of producing a desired quantum state, the target could be to implement a quantum algorithm or more generally a unitary operation. In particular, researchers have lately demonstrated significant advancements in circuit synthesis by harnessing machine learning techniques, ranging from reinforcement learning to generative models, as for instance in quantum state preparation (Arrazola et al., 2019; Bolens & Heyl, 2021; Melnikov et al., 2018), ansätze prediction (He et al., 2023; Shen, 2023; Zhang et al., 2021), circuit optimization (Fösel et al., 2021; Ostaszewski et al., 2021) or unitary compilation (Zhang et al., 2020; Moro et al., 2021; Sarra et al., 2023; Preti et al., 2023). In most of these applications, ML methods rely on minimizing a cost function that compares the output of the resulting quantum computation to the desired one. This metric is generally classically hard to simulate (Khatri et al., 2019), hindering the trainability of the models. Additionally, these methods are typically trained on specific configurations defined by their training data and struggle to adapt to novel, uncharted scenarios.

To address these challenges, we use denoising diffusion models (DM) (Sohl-Dickstein et al., 2015), the current state-of-the-art machine learning generative method (Rombach et al., 2022). Given a text prompt specifying a certain task, the model generates the desired quantum architecture. While we primarily focus on qubit-based quantum circuits, the model is straightforwardly extendable to other platforms such as measurement-based quantum computers (Raussendorf & Briegel, 2001) or fermionic processors (González-Cuadra et al., 2023). As any DM, the architecture is trained to *denoise* corrupted samples from the training set, conditioned to the given text prompt. Hence, contrary to previous approaches, one avoids comparing the output of the generated quantum circuits to the target result. This ensures that no classical simulation of the circuit is needed and thus prevents the computation of intractable cost functions (Khatri et al., 2019). Once trained, the method excels at generating correct, never seen circuits, allowing for instance the discovery of novel circuits decompositions, replacements, or compressions. The architecture is highly flexible, allowing easy control of circuit characteristics such as length and number of qubits but also to account for the physical constraints of the target physical device, such as

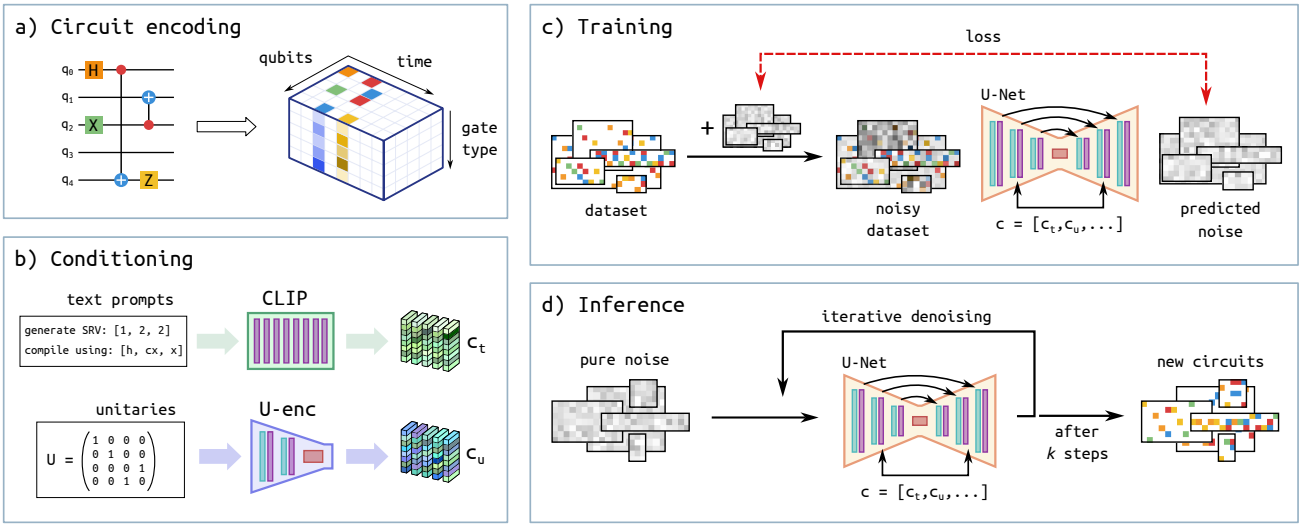


Figure 1. **Quantum circuit generation pipeline summary. (a)** Quantum circuits are encoded in a three dimensional tensor, where each gate is encoded as a continuous vector of certain length (vertical direction), schematically represented here as a color (upper plane). **(b)** Creation of the diffusion model's conditioning. Text is transformed into a continuous representation by means of a pre-trained CLIP encoder. In other cases, as for unitary compilation, an encoder is trained together with the DM to create the encoding of an input unitary. **(c-d)** Schematic representation of the training of the diffusion model and the posterior inference from the trained model. See text for details.

qubit connectivity. Additionally, in situations where generating training sets is costly, trained models can be updated with a handful of samples to acquire new skills. We demonstrate the versatility of the method in two applications: First, we train the DM to generate circuits with varying degrees of entanglement, and use this task to benchmark the model's capabilities; Second, we train the DM to perform unitary compilation of unitaries with restricted sets of gates.

Circuit encoding, model and training pipeline

In this section we present a comprehensive description of the proposed pipeline, namely, the encoding used to transform circuits into trainable tensors, the architecture of the denoising diffusion model (DM) and some key aspects of the model's training. All details necessary to reproduce the results presented here are available in the appendices and the accompanying code repository (Fürrutter & Muñoz-Gil, 2023), which also includes a series of examples that illustrate how diffusion models operate.

Circuit encoding To train the DM, we take inspiration from (Fösel et al., 2021) and represent the circuits as three dimensional tensors. The first dimension encodes the qubit index, the second the *time step* of gate placement (constraining to one gate per time step) and the third represents each gate (Figure 1a). We consider here a continuous gate representation of arbitrary length, fixed before training. While in this work we assign random orthogonal high-dimensional representations to each gate, in more complex scenarios, for example optical devices in photonic setups (Melnikov et al., 2018), such representations could be learned prior to the DM training. To decode circuits from the tensor representation, we calculate the cosine similarity between the described gate and our existing gate set, selecting then the gate with the largest similarity. Further details are given in Methods.

Diffusion model DMs are a class of generative models aimed at learning a diffusion process that generates the probability distribution of a given training dataset (Sohl-Dickstein et al., 2015). Practically, such process involves iteratively denoising an initially noisy sample to produce a high-quality one. Led by the emergence of latent diffusion (Rombach et al., 2022), and the ability to guide / condition their generation at will (Ho & Salimans, 2022), DMs have become the go-to method for generative applications in fields like image generation (Podell et al., 2023), audio synthesis (Kong et al., 2020), video generation (Singer et al., 2022), and protein structure prediction (Watson et al., 2023). In this work, we construct a pipeline similar to that of Ref. (Rombach et al., 2022). We first create the condition that will guide the generation of the DM (Figure 1b). Text prompts are transformed into continuous representation by means of a pre-trained CLIP encoder, a neural network trained to associate visual content and textual descriptions, whose acronym relates to its training procedure (Contrastive Language-Image Pre-training) (Radford et al., 2021). More-

over, depending on the task at hand, further conditioning may be necessary. For instance, in unitary compilation, an encoded representation of unitary is appended to the encoded text representation. Here, the unitary encoder is trained alongside the DM (see Methods).

To train the DM itself, we corrupt the samples in the training set (i.e. the encoded circuits) with different levels of Gaussian noise (Figure 1c). The model then learns to predict the noise present in each sample which, subtracted from the input samples, effectively denoises them. We adapt the architecture from (Rombach et al., 2022) for the noise prediction model. The architecture is a typical U-Net consisting of an encoder-decoder layout with skip connections in between. We construct it by combining residual convolution layers and cross-attention layers (Chen et al., 2021). The latter implements the text conditioning naturally, as done by transformers in natural language processing (Vaswani et al., 2017). For our use-case, we propose some adjustments to the model to: i) account for the non-locality of qubit-connections. ii) allow us to feed circuits of any size both in terms of the number of qubits and its length (see Methods). After training, one feeds a completely noisy tensor into the model which, guided by the chosen condition, iteratively denoises it, generating after some steps a high-quality sample (Figure 1d).

Results

We now illustrate the capabilities of the method by applying it to two distinct problems: entanglement generation and unitary compilation. The former serves as a benchmark for showcasing the method's potential in different scenarios. The latter is a critical problem in the field, and will allow us to demonstrate the full capabilities of the method. Importantly, in both cases, the model is trained using the same denoising loss function, with different tasks achieved by modifying the training samples and their conditioning.

Entanglement generation The automated design of experiments to generate new types of entanglement has been an exciting recent development in quantum physics (Krenn et al., 2023; Melnikov et al., 2018; Krenn et al., 2016). Here, our aim is to generate quantum circuits that produce states with specific entanglement, as described by their Schmidt rank vector (SRV), a numerical vector of dimension equal to the number of qubits that contains the rank of the reduced-density matrix for each subsystem (Huber & de Vicente, 2013). As we here deal with qubits, each element of the SRV is either 1 (non-entangled qubit) or 2 (entangled qubit). To construct the training set, we randomly generate circuits made of H and $CNOT$ gates with varying numbers of qubits (from 3 to 8) and gates (from 2 to 52). We then calculate their corresponding SRV, which can in this case take

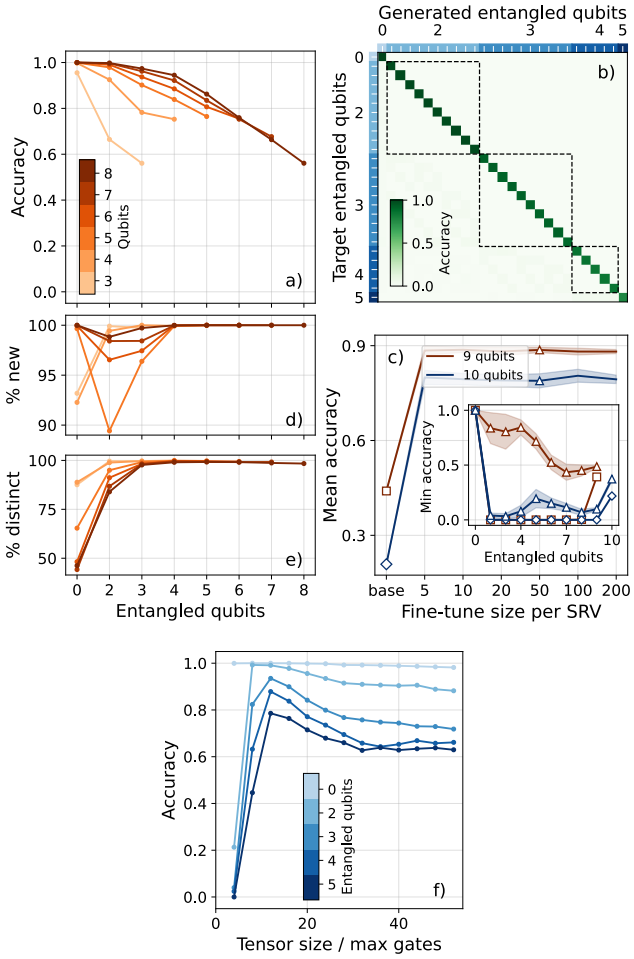


Figure 2. **Entanglement generation. (a)** Model accuracy vs. the number of entangled qubits, for circuits of different qubits number. **(b)** Confusion matrix comparing, in each row and column, the input and generated SRVs for circuits of 5-qubits. For clarity, SRVs are grouped by their respective number of entangled qubits. **(c)** Mean accuracy over all SRVs vs. the number of samples used for fine-tuning the model in (a), for 9 and 10-qubit circuits. "*base*" denotes the model's predictions before fine-tuning. Inset: minimum accuracy for the base (squares / rhombus) and fine-tuned model (triangles) on 50 circuits per SRV. Solid line and shaded area represent the mean and standard deviation over 10 fine-tune runs for both plots, respectively. **(d-e)** Percentage of new (not in training set) and distinct (not repeated in the generated sample) generated circuits. **(f)** Model accuracy vs. the input tensor size (which determines the maximum gate count), for 5-qubit circuits and different numbers of entangled qubits. Details on the data used in this figure given in Methods

values $\in [1, 2]$, and use it as text conditioning (see Figure 1). We found that balancing the dataset both in terms of SRV but also w.r.t. to the circuit length greatly improved the training performance. Once the model is trained, we generate new circuits for different SRV to assess its performance.

We first observe that almost all continuous tensors generated by the DM can be faithfully transformed into circuits (99.6% conversion rate). This highlights the model's ability to learn gate encoding and proper gate placement within the circuit, ensuring that only one gate is placed at each time step. In Figure 2a, we present the accuracy of the DM, defined as the percentage of circuits generated from a given SRV-conditioning that indeed produce the desired SRV. To enhance readability, we average across SRV with the same number of entangled qubits, as the accuracy is usually constant within these. Figure 2b provides a detailed accuracy analysis for circuits of 5 qubits. The majority of the circuits produce the correct entanglement, with the accuracy slightly dropping for larger numbers of entangled qubits. This decrease arises because more entangled qubits demand a larger number of gates, thus presenting a more challenging problem. The variations in accuracy across different qubit numbers mainly reflect their representation in the training set. Nevertheless, as sampling from the DM is a cheap operation, one strategy is to sample a few circuits with the desired property and select successful ones. While this may entail computationally costly operations, as one needs to simulate the quantum circuit and measure its properties, it needs to be computed only for the few circuits sampled.

In many cases, especially when increasing the number of qubits, creating a sufficiently large dataset to train the model becomes prohibitively expensive. In such scenarios, one can first train a model with simpler examples and then *fine-tune* the model with few expensive ones. We demonstrate such feature by fine-tuning the model above to generate circuits of 9 and 10 qubits. As depicted in Figure 2c, the base model can already generate with a reasonable average accuracy such larger circuits. However, a deeper examination shows that, while the method excels for certain SRVs, it completely fails in others (see e.g. the minimum accuracy in inset, square markers) . By fine-tuning the model with few circuits of 9 and 10 qubits per SRV (compared to the tens of thousands used for the original training), we greatly improve its accuracy, and most importantly banish the number of unattained target SRVs (inset, triangle markers).

Furthermore, the model exhibits an exceptional capability to discover new circuits that achieve the target SRV. Figure 2d-e show that, across all SRVs and qubit sizes, the generated circuits are both new (not contained in the training set) and unique (not repeated within the generated set). The model is hence able to generalize beyond the circuits given in the training set, providing a systematic and cost-effective method for discovering novel solutions to the task at hand. Lastly, the DM input tensor can be resized at will to account for different qubit numbers (as done above) and varying circuit lengths. Figure 2f shows that the accuracy of the model is almost constant no matter what the input tensor's size is. The sudden drop for small sizes is due to the impossibility of generating entanglement with few gates.

Masking and editing circuits In addition to resizing the input tensor, one can also fix parts of it, as proposed in Ref. (Lugmayr et al., 2022) for image editing. This enables two operations: one prevents the model to place gates somewhere in the circuit, the other imposes the presence of certain gates. Importantly, one does not require to retrain the model in any form, and the results presented in this section use the same model as employed above.

The first operation, *masking*, offers a means to introduce specific structural constraints into the circuits. For instance, current quantum devices exhibit limited non-local qubit connectivity. Performing operations between distant qubits, such as a two-qubit gate, must be broken into sequences of intermediate gates that swap the operation through the circuit (Bäumer et al., 2023), ultimately reducing gate fidelity. However, such a constraint is not naturally represented in the circuit picture, where seemingly any qubit connectivity is feasible. As illustrated in Figure 3a, to enforce the physical limitation of a known device onto the circuit, we mask desired parts of the model's input tensor preventing any gate placement in the marked region. Then, this masked tensor together with the desired text prompt, is input into the model to generate the suitable circuit. This approach also grants control over the number of qubits and circuit length e.g., by masking entire qubit rows, although the accuracy of the generated circuits generally falls short compared to employing tensors of varying sizes, as done in the previous sections.

The second operation, *editing*, allows us to fix specific gates throughout the circuit prior to the diffusion process. From a practical perspective, we consider here that a quantum state of interest, in the form of a circuit, is given as an initial state in which further computations need to be performed (Figure 3b). For instance, starting with an input circuit of a certain SRV_i, we aim to transform it into a target SRV_t (Figure 3c). We allow for this task circuits of maximum 20 gates, where at most the first 6 generate the initial SRV_i. Notably, the model successfully handles tasks that involve increasing entanglement (elements above the diagonal), but encounters challenges when tasked with reducing entanglement (elements below the diagonal). This becomes particularly evident when a fully entangled state is given (lowest row). While on average the model produces accurate edits, for some circuits where all or most of the entanglement had to be destroyed, no correct editing was found. We note that the

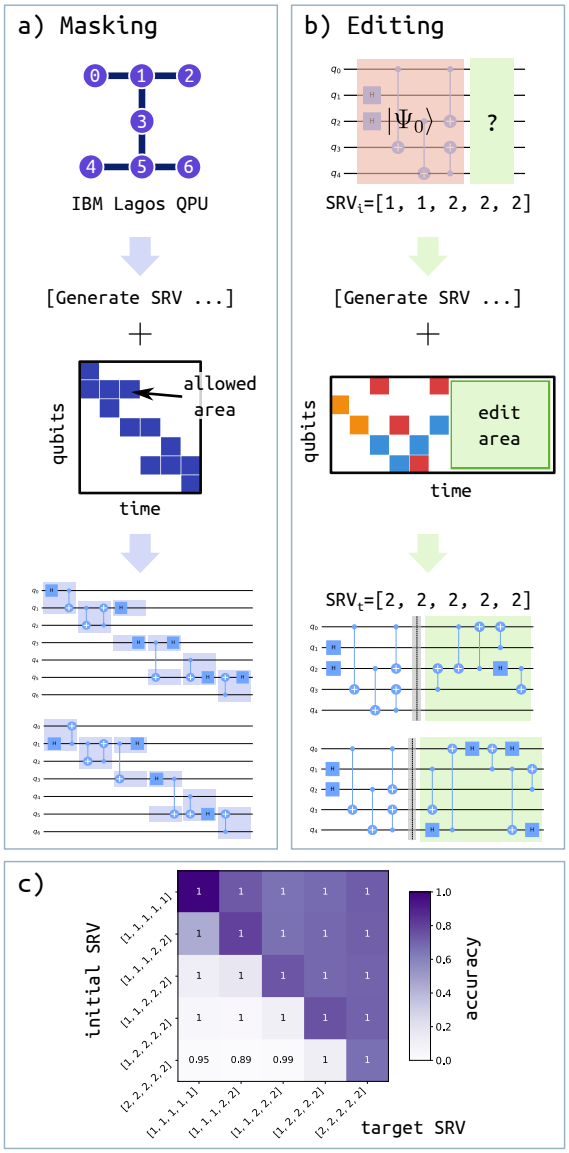


Figure 3. **Masking and editing circuits. (a)** Masking: the layout of a quantum processor can be embedded as a mask that prevents the model from placing gates at specific parts of the input tensor (white area). **(b)** Editing: parts of the circuit can be fixed to given gates prior to generation, for example to account for an initial input quantum state on which a desired quantum computation is to be performed. **(c)** Accuracy when editing circuits from an input SRV to a target SRV. Numbers highlight the fraction of initial circuits (256) where at least one solution was found within a sample of 1024 generated circuits.

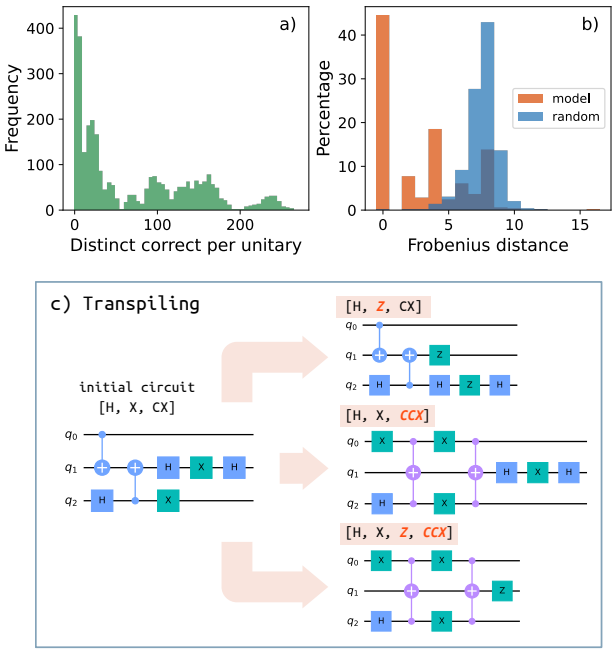


Figure 4. **Unitary compilation. (a)** Number of exact and distinct circuit compilations, over 1024 generated circuits, for 3100 input unitaries. At least one exact circuit was found for 92.6% of the unitaries. **(b)** Frobenius distance between the same 3100 target unitaries and circuits generated randomly (blue) or by the diffusion model (orange). **(c)** Given an input circuit, its variations can be produced by first calculating its unitary and then conditioning the model with this unitary plus different gate sets.

latter is also the most challenging scenario, as disentangling highly entangled states requires the largest number of gates. Moreover, our experiments (see Methods) suggest that the current model slightly correlates SRVs with circuit lengths, favoring shorter circuits for low entanglement. As editing towards such low entanglement is preceded by many gates for initial highly entangled states, the model has difficulties creating such target SRVs. Nonetheless, in the worst case scenario the model is still capable of correctly editing 89.1% of the input circuits.

Unitary compilation We now train the model in one of the most crucial tasks in quantum circuit synthesis: compiling given unitaries into circuits, constrained to a certain set of gates. To do so, we generate a variety of circuits with differing lengths and gates, subsequently calculating their corresponding unitaries. In previous tasks, generation was only conditioned to the information contained in the text prompt. Here, we extend such conditioning by appending to the CLIP encoded text prompt, which now encodes the set of gates to be used, a learned encoding of the unitary (see Figure 1). Just as before, the model is trained to denoise noisy input tensors, avoiding the exponential cost of comput-

ing distances between generated and target unitaries. After training, new unitaries are fed into the model to evaluate its generative compilation capabilities.

We restrict here to 3 qubit circuits and unitaries computed from randomly generated circuits, ensuring that at least one exact implementation exists under the given constraints. Further, we only pick random unitaries that are not within the training set and new to the model. For each input unitary, we generate 1024 circuits. Remarkably, the model successfully identifies the correct exact unitary for 92.6 % of the 3100 tested unitaries. In most instances the model finds multiple valid circuits for the same unitary (Figure 4a), allowing the prospective user to choose according to its needs. To gain further insights into the model's capabilities, we calculate the Frobenius norm based distance $\frac{1}{2}||U_t - U_g||_F^2$ between the target (U_t) and generated (U_g) unitaries (Figure 4b). As anticipated from the previous result, the majority of target unitaries can be compiled with norm zero. When this is not achieved, the norm attained is smaller than what would be expected from a randomly generated circuit, suggesting that the model, while not capable of succeeding in the task, is heading in the right direction. For those few failed cases, one can rely on further sampling or leverage the generated circuit as a starting point for other methods (Weiden et al., 2023). In that sense, the masking and editing operations presented above can be performed in exactly equal manner for this task.

Interestingly, one can explore the compilation of the same unitary conditioned to multiple gate sets to uncover optimized solutions or identify unknown gate decompositions or compressions (Figure 4c).

Discussion

In this work, we have demonstrated the potential use of diffusion models for quantum circuit synthesis. As the development of quantum hardware progresses, there is an increasing need for efficient and versatile methods that convert desired quantum operations, as needed, e.g., in quantum algorithms (Dalzell et al., 2023) or quantum simulation (Daley et al., 2022), into practical physical implementations under given hardware constraints. The approach introduced here addresses several persistent challenges in circuit synthesis: i) through the model's denoising loss, the method avoids the exponential overhead of classically computing cost functions comparing generated and target quantum circuits; ii) the model's flexibility enables adjusting text prompts and input tensors to produce circuits with specific lengths and topologies. This is especially useful for addressing physical device constraints or other boundary conditions and desiderata; iii) Fine-tuning the model with a small number of samples from pre-trained models enables its training in situations where obtaining a full training set is prohibitively expensive. We demonstrated these capabilities in two tasks: quantum state preparation, specifically entanglement generation, and exact unitary compilation. The model not only tackles these tasks but also uncovers multiple novel solutions, offering further insights into the problem at hand.

The current work offers multiple extensions. On the one hand, science greatly benefits from and needs interpretable machine learning solutions – models that not only provide predictions but also offer insights into how these solutions were found. We believe that certain features of our proposed architecture can serve this interpretative purpose. For instance, as done in the context of image generations (Niu et al., 2021; Wiegreffe & Pinter, 2019; Hertz et al., 2022; Li et al., 2023), the U-Net's cross-attention maps can be used to reveal which parts of the circuit are deemed important by the machine for different text prompts. On the other hand, perhaps the most direct benefit is the extension of the current method to continuous, parametrized gates, which can seamlessly be integrated into the current circuit encoding and training framework. Moreover, the model can be extended to different platforms and applications, as e.g., the transpilation of quantum-gate-based operations to measurement-based quantum computations (Raussendorf & Briegel, 2001; Raussendorf et al., 2003; Briegel et al., 2009), discovering quantum experiments (Melnikov et al., 2018; Krenn et al., 2021) or finding native compilations in fermionic (González-Cuadra et al., 2023) or anyonic (Zhang et al., 2020) processors. For each scenario, it is essential for the user to establish a suitable tensor encoding for its device's computational units, such as optical elements in a photonic processor or native gates in a fermionic processor. With a specific encoding set defined, a similar or even the same model presented here becomes readily adaptable to fulfill distinct user needs.

A key factor for the widespread application of the presented method is its scalability. From a ML perspective, the diffusion model used here can be easily enlarged to reach the current state-of-the-art industrial models, given necessary computational power. Indeed, a model on the scale of Stable Diffusion XL (Podell et al., 2023) could potentially generate circuits of 1024 qubits and 1024 gates. However, as in any circuit synthesis method, the main bottleneck arises from the quantum nature of the problem. For instance, assembling a suitable dataset for such a large-scale model poses a considerable challenge, as it requires the classical computation of the unitary corresponding to each circuit or the SRV of its output. Fine-tuning smaller, more manageable models for larger sizes, as demonstrated in Figure 2c, may be practical for systems with tens of qubits. Yet, alternative strategies may be necessary for handling hundreds or thousands of qubits. Moreover, performing unitary compilation involves conditioning the model with a unitary matrix which grows exponentially with the number of qubits. Future iterations of

the model here presented should explore more efficient conditionings, for example, based on Hamiltonians generating the unitary evolution of the system.

References

Arrazola, J. M., Bromley, T. R., Izaac, J., Myers, C. R., Brádler, K., and Killoran, N. Machine learning method for state preparation and gate synthesis on photonic quantum computers. *Quantum Science and Technology*, 4(2): 024004, 2019. doi: 10.1088/2058-9565/aaf59e.

Bäumer, E., Tripathi, V., Wang, D. S., Rall, P., Chen, E. H., Majumder, S., Seif, A., and Minev, Z. K. Efficient long-range entanglement using dynamic circuits. *arXiv preprint arXiv:2308.13065*, 2023. doi: 10.48550/arXiv.2308.13065.

Bolens, A. and Heyl, M. Reinforcement learning for digital quantum simulation. *Physical Review Letters*, 127(11): 110502, 2021. doi: 10.1103/PhysRevLett.127.110502.

Briegel, H. J., Browne, D. E., Dür, W., Raussendorf, R., and Van den Nest, M. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009. doi: 10.1038/nphys1157.

Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L., and Coles, P. J. Challenges and opportunities in quantum machine learning. *Nature Computational Science*, 2(9):567–576, 2022. doi: 10.1038/s43588-022-00311-3.

Chen, C.-F. R., Fan, Q., and Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 357–366, 2021. doi: 10.48550/arXiv.2103.14899.

Daley, A. J., Bloch, I., Kokail, C., Flannigan, S., Pearson, N., Troyer, M., and Zoller, P. Practical quantum advantage in quantum simulation. *Nature*, 607(7920):667–676, 2022. doi: 10.1038/s41586-022-04940-6.

Dalzell, A. M., McArdle, S., Berta, M., Bienias, P., Chen, C.-F., Gilyén, A., Hann, C. T., Kastoryano, M. J., Khabiboulline, E. T., Kubica, A., et al. Quantum algorithms: A survey of applications and end-to-end complexities. *arXiv preprint arXiv:2310.03011*, 2023. doi: 10.48550/arXiv.2310.03011.

Farhi, E., Goldstone, J., and Gutmann, S. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014. doi: 10.48550/arXiv.1411.4028.

Feynman, R. P. et al. Simulating physics with computers. *Int. j. Theor. phys*, 21(6/7), 2018. doi: 10.1007/BF02650179.

Fösel, T., Niu, M. Y., Marquardt, F., and Li, L. Quantum circuit optimization with deep reinforcement learning. *arXiv preprint arXiv:2103.07585*, 2021. doi: 10.48550/arXiv.2103.07585.

Fürrutter, F. and Muñoz-Gil, G. Genqc github repository, 2023. URL https://florianfuerrutter.github.io/genQC/.

González-Cuadra, D., Bluvstein, D., Kalinowski, M., Kaubruegger, R., Maskara, N., Naldesi, P., Zache, T. V., Kaufman, A. M., Lukin, M. D., Pichler, H., Vermersch, B., Ye, J., and Zoller, P. Fermionic quantum processing with programmable neutral atom arrays. *Proceedings of the National Academy of Sciences*, 120(35):e2304294120, 2023. doi: 10.1073/pnas.2304294120.

He, Z., Zhang, X., Chen, C., Huang, Z., Zhou, Y., and Situ, H. A gnn-based predictor for quantum architecture search. *Quantum Information Processing*, 22(2):128, 2023. doi: 10.1007/s11128-023-03881-x.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-Or, D. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. doi: 10.48550/arXiv.2208.01626.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. doi: 10.48550/arXiv.2207.12598.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020.

Huber, M. and de Vicente, J. I. Structure of multidimensional entanglement in multipartite systems. *Phys. Rev. Lett.*, 110:030501, Jan 2013. doi: 10.1103/PhysRevLett.110.030501. URL https://link.aps.org/doi/10.1103/PhysRevLett.110.030501.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. Openclip, October 2023. URL https://doi.org/10.5281/zenodo.10037810.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers, 2023.

Khatri, S., LaRose, R., Poremba, A., Cincio, L., Sornborger, A. T., and Coles, P. J. Quantum-assisted quantum compiling. *Quantum*, 3:140, 2019. doi: 10.22331/q-2019-05-13-140.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. doi: 10.48550/arXiv.1412.6980.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020. doi: 10.48550/arXiv.2009.09761.

Krenn, M., Malik, M., Fickler, R., Lapkiewicz, R., and Zeilinger, A. Automated search for new quantum experiments. *Phys. Rev. Lett.*, 116:090405, Mar 2016. doi: 10.1103/PhysRevLett.116.090405. URL https://link.aps.org/doi/10.1103/PhysRevLett.116.090405.

Krenn, M., Kottmann, J. S., Tischler, N., and Aspuru-Guzik, A. Conceptual understanding through efficient automated design of quantum optical experiments. *Physical Review X*, 11(3):031044, 2021. doi: 10.1103/PhysRevX.11.031044.

Krenn, M., Landgraf, J., Foesel, T., and Marquardt, F. Artificial intelligence and machine learning for quantum technologies. *Physical Review A*, 107(1):010101, 2023. doi: 10.1103/PhysRevA.107.010101.

Li, Y., Keuper, M., Zhang, D., and Khoreva, A. Divide & bind your attention for improved generative semantic nursing, 2023.

Lin, S., Liu, B., Li, J., and Yang, X. Common diffusion noise schedules and sample steps are flawed, 2023.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022. doi: 10.48550/arXiv.2201.09865.

McArdle, S., Endo, S., Aspuru-Guzik, A., Benjamin, S. C., and Yuan, X. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020. doi: 10.1103/RevModPhys.92.015003.

Melnikov, A. A., Poulsen Nautrup, H., Krenn, M., Dunjko, V., Tiersch, M., Zeilinger, A., and Briegel, H. J. Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences*, 115(6):1221–1226, 2018. doi: 10.1073/pnas.1714936115.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022.

Moro, L., Paris, M. G., Restelli, M., and Prati, E. Quantum compiling by deep reinforcement learning. *Communications Physics*, 4(1):178, 2021. doi: 10.1038/s42005-021-00684-3.

Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models, 2021.

Ning, M., Sangineto, E., Porrello, A., Calderara, S., and Cucchiara, R. Input perturbation reduces exposure bias in diffusion models, 2023.

Niu, Z., Zhong, G., and Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021. doi: 10.1016/j.neucom.2021.03.091.

Ostaszewski, M., Trenkwalder, L. M., Masarczyk, W., Scerri, E., and Dunjko, V. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34: 18182–18194, 2021. doi: 10.48550/arXiv.2103.16089.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

Preskill, J. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018. doi: 10.22331/q-2018-08-06-79.

Preti, F., Schilling, M., Jerbi, S., Trenkwalder, L. M., Nautrup, H. P., Motzoi, F., and Briegel, H. J. Hybrid discrete-continuous compilation of trapped-ion quantum circuits with deep reinforcement learning. *arXiv preprint arXiv:2307.05744*, 2023. doi: 10.48550/arXiv.2307.05744.

Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021. doi: https://doi.org/10.48550/arXiv.2103.00020.

Raussendorf, R. and Briegel, H. J. A one-way quantum computer. *Physical review letters*, 86(22):5188, 2001. doi: 10.1103/PhysRevLett.86.5188.

Raussendorf, R., Browne, D. E., and Briegel, H. J. Measurement-based quantum computation on cluster states. *Physical review A*, 68(2):022312, 2003. doi: 10.1103/PhysRevA.68.022312.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022. doi: 10.48550/arXiv.2112.10752.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023.

Sarra, L., Ellis, K., and Marquardt, F. Discovering quantum circuit components with program synthesis. *arXiv preprint arXiv:2305.01707*, 2023. doi: 10.48550/arXiv.2305.01707.

Shen, Y. Prepare ansatz for vqe with diffusion model. *arXiv preprint arXiv:2310.02511*, 2023. doi: 10.48550/arXiv.2310.02511.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022. doi: 10.48550/arXiv.2209.14792.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019. doi: 10.1117/12.2520589.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015. doi: 10.48550/arXiv.1503.03585.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. doi: 10.48550/arXiv.1706.03762.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976): 1089–1100, 2023. doi: 10.1038/s41586-023-06415-8.

Weiden, M., Younis, E., Kalloor, J., Kubiatowicz, J., and Iancu, C. Improving quantum circuit synthesis with machine learning. *arXiv preprint arXiv:2306.05622*, 2023. doi: 10.48550/arXiv.2306.05622.

Wiegreffe, S. and Pinter, Y. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019. doi: doi.org/10.48550/arXiv.1908.04626.

Zhang, S.-X., Hsieh, C.-Y., Zhang, S., and Yao, H. Neural predictor based quantum architecture search. *Machine Learning: Science and Technology*, 2(4):045027, oct 2021. doi: 10.1088/2632-2153/ac28dd. URL https://dx.doi.org/10.1088/2632-2153/ac28dd.

Zhang, Y.-H., Zheng, P.-L., Zhang, Y., and Deng, D.-L. Topological quantum compiling with reinforcement learning. *Physical Review Letters*, 125(17):170501, 2020. doi: 10.1103/PhysRevLett.125.170501.

A. Quantum circuit encoding

In this section, we detail the procedure to encode quantum circuits into continuous tensors, which can then be fed into deep neural networks. Inspired by methods from the natural-language-processing (NLP) community, we first transform quantum gates into discrete tokens and then into continuous embeddings (see Extended Data Fig. 1). We will also refer to these embeddings as *colors*, as done in Figure 1a.

The proposed encoding is constructed to account for:

1. any sequence length (e.g., number of gates);
2. any spatial size (e.g., number of qubits);
3. any number of local or non-local spatial connections, as for instance two-qubit gates with target and control nodes;
4. flexibility in the number of tokens, to possibly extend the number of gates at later stages of training or fine-tunings.

Points 1 and 2 are achieved by mapping the time (length) and space (qubits) dimensions each to one of the tensor's dimensions. Moreover, we require that only one gate can be present at each time step. Additional to these two axes, the tensor has a dedicated "gate dimension", of arbitrary size, used to describe each of the various gates. To address point 3, we describe the control and target nodes of a multi-qubit gate with the same numerical embedding but with opposite signs. As continuous vectors are used to describe gates, point 4 is seemingly addressed by assigning a new vector to each new gate. We note here that the proposed encoding is general and can be applied to any types of categorical time sequence. For instance, such continuous tensors can be used to represent parametrized gates or, extending beyond quantum circuits, optical devices in quantum optical experiments.

Encoding – The circuit encoding consists of two steps. First, we tokenize the circuit gates into an integer matrix representation. With it, each gate corresponds to an integer number, and we use a sign to specify the target and control nodes. While one could work with such circuit representation, machine learning models' stability benefits from normalized continuous inputs, preferably centered around 0 and in the range [-1,1]. Hence, we transform each of the N gates/tokens into a vector $\mathbf{v} \in \mathbb{R}^d$, where the dimension d is arbitrary. Again, the two nodes of a multi-qubit gate share the same $\mathbf{v}$ but with opposite signs. We usually consider $d \geq N$, and choose the vectors $\mathbf{v}$ to be mutually orthogonal, defining a linearly independent set. The latter is not strictly necessary, but makes the decoding of model generated embeddings more robust to numeric variances. In this work, we consider $d = N + 2$, where two tokens are used for padding and background. More on the former is presented in the Training section below.

As mentioned in the main text, in principle all the previous embeddings can be generated by trained machines (as done in NLP embeddings). Such training is usually performed prior to the diffusion model training. Importantly, if one trains it together with the denoising model, the trained embedder would learn to set all tokens to zero such that the denoising model can then extract the noise trivially.

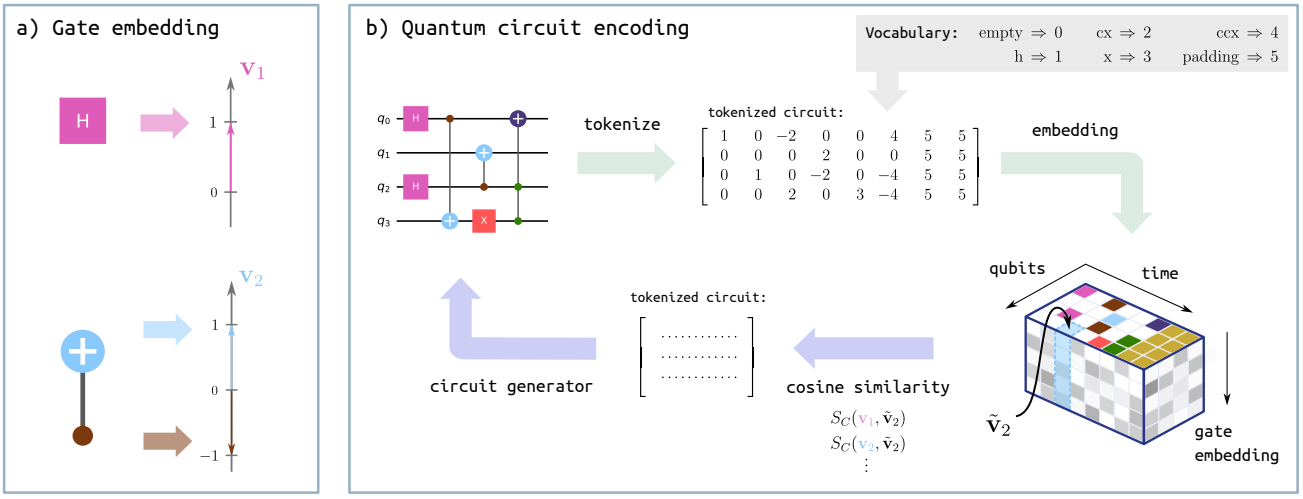
Decoding – The first step of the decoding entails transforming the continuous tensors $\mathbf{v}_{\text{gen}}$ generated by the model back to tokens. For that, we choose the token k whose embedding $\mathbf{v}_k$ has the highest absolute cosine similarity S_C to $\mathbf{v}_{\text{gen}}$:

$$\tilde{k} = \arg\max_k |S_C(\mathbf{v}_k, \mathbf{v}_{\text{gen}})|. \tag{1}$$

To resolve the node type, we perform a second step:

$$k = \tilde{k} \cdot \text{sign}\, S_C(\mathbf{v}_{\tilde{k}}, \mathbf{v}_{\text{gen}}) \tag{2}$$

Such calculation is performed for every space-time position of the tensor encoding, returning a tokenized integer matrix, which exactly describes a valid circuit. We refer to error circuits as those in which the model places two or more gates per time step or does not properly place either the target and control nodes of multi-qubit gates. Interestingly, we notice that the trained models matches the original embeddings with high precision (i.e.,$\mathbf{v}_{gen} \approx \mathbf{v}_k$) and assume one could "overload" the vector space $\mathbb{R}^d$ with $N > d$ gates. This is further motivated by the results in visual image tasks (Rombach et al., 2022), where diffusion generated images are highly color accurate.



Extended Data Figure 1. **Quantum circuit tensor encoding. (a)** Schematic representation of the gate embeddings for a single and multi qubit gate. **(b)** Quantum circuit encoding and decoding pipeline. For encoding (green arrows), an input quantum circuit (top left) is first tokenized based on the proposed vocabulary. Then, the token matrix is transformed into a continuous tensor based on the chosen embeddings $\mathbf{v}_i$ (bottom right). In order to decode a continuous tensor into a circuit (blue arrows), we first use the cosine similarity between input embeddings and the ones assigned to existing tokens to generate a tokenized matrix, which is then transformed back into a circuit by means of the vocabulary. The transformation between circuits and tokens depends on such vocabulary, and can be changed at will to cope with the desired computing framework or platform. Further details are given in text.

B. Dataset

In this section we detail the procedure used to generate the datasets, with which both models presented in this work were trained (one for entanglement generation and one for unitary compilation). In both case we follow the same steps:

1. **Generate random circuits.** For each circuit, given the full set of gates considered for the current task, we first sample a subset of it. Next, we sample the number of gates to be placed from a uniform distribution with given lower and upper bounds. Then, we sample from the gate subset such a number of gates and place them in the circuit. Finally, we compute the condition associated with the generated circuits. For the entanglement generation task, we calculate the SRVs and convert them to prompts, e.g. "Generate SRV: [1, 2, 2]". For the unitary compilation task, we calculate the unitary and store it together with a prompt specifying the gate set that was used to sample the circuit, e.g. "Compile using: [cx, h, x]".

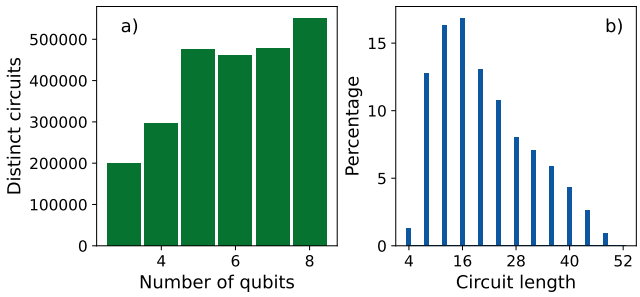
2. **Optimize circuits.** We use the optimization-stage of the Qiskit `qiskit.compiler.transpile` function to optimize all the random circuits (Qiskit contributors, 2023). This step mainly removes duplicated consecutive gates and generally improves the quality of dataset. Training with such an optimized dataset indeed yields a significant increase of the model's accuracy. After optimizing, we delete all circuit duplicates.

3. **Balance dataset.** We balance the dataset in a two-step procedure. First, we balance the text prompts, i.e., the SRVs or compilation gate sets, such that every prompt has the same number of circuits. In the second step, we balance the circuit lengths within the previously balanced text prompt bins. For that, we limit the number of circuits to a maximum threshold for each gate count. As the threshold we choose q-quantiles of the lengths within the text prompt bins. In particular, we chose $q = 0.25$ for both the entanglement generation and unitary compilation task, and $q = 0.05$ for the fine-tuning to 9 and 10 qubits in the entanglement generation task. Our experiments show that such balancing increases the dataset quality and improves the model performance.

Using these steps, we produce datasets with the parameters specified in Extended Data Table 1. For the entanglement generation task, we use a multi-qubit dataset of 3 to 8 qubit circuits, with numbers of qubit and circuit length distribution presented in Extended Data Fig. 2. We trained the compilation task on a dataset with 2.5 million distinct 3-qubit circuits, created from 1.17 millions distinct unitaries.

	qubits —	min gates —	max gates	gate pool
SRV	3	2	16	H, CX
	4	3	20	
	5	4	28	
	6	5	40	
	7	6	52	
	8	7	52	
SRV, fine-tune	9	8	36	H, CX
	10	9	36	
Compilation	3	2	12	H, CX, Z, X, CCX, SWAP

Extended Data Table 1. **Dataset sampling parameters.** Parameters used for the generation of the various training datasets used throughout the manuscript. SRV refers here to the entanglement generation task. We note that these parameters are chosen prior to the optimization step (step 2 in the Methods Dataset section), which effectively reduces the final gate number for some circuits.



Extended Data Figure 2. **Entanglement generation dataset distribution.** Characteristics of the training dataset used for the entanglement generation task, sampled according to Extended Data Table 1 and balanced as described in Methods, Training section. Depending on the training step (max or bucket padding, see aforementioned section), we sample batches either from the whole dataset or buckets containing circuits of fixed number of qubits. **(a)** Number of distinct circuits as a function of the number of qubits. For lower qubit counts, less distinct circuits exist, resulting in an inevitable lower number in the training dataset. **(b)** Distribution of circuit lengths, which are in this case multiples of the U-Net scaling factor 4, due to the length padding explained in Methods, Pipeline and Architectures section.

C. Pipeline and architectures

In this section we present the details of the machine learning models, namely the denoising U-Net, the text encoder CLIP and the unitary encoder. Moreover, we explain how to combine conditions for the unitary compilation task.

U-Net – For the U-Net denoising model, we adjust the architecture of Stable Diffusion (Rombach et al., 2022). An overview of the model is presented in Extended Data Fig. 3a. The structure of the self and cross-attention layers is inspired by the Stable Diffusion model, which proposed a reordering of the normalization layers, compared to the original transformer architecture (Vaswani et al., 2017), and introduced the use of a residual connection.

We design the architecture such that it accepts the circuit tensor encodings proposed in the Quantum circuit encoding section above. As discussed in the main text, to be useful for the proposed tasks, the model needs to account for: i) non-locality of qubit-connections. ii) circuits of any size both in terms of the number of qubits and its length.

To achieve this, all convolutions in the U-Net are performed *only* on the time dimension. From a technical perspective, this entails having convolution kernels of size $1 \times f$, where f is the filter size in the time dimension, which varies throughout the architecture (see Extended Data Fig. 3). Such feature ensures that there is no locality bias induced in the qubit space dimension and, most importantly, that the perceptive field in this dimension is not restricted. This is crucial for instance in bigger circuits, where connections between outermost qubits exist. Computations over the spatial dimension are only performed in the self attention-layers which, most importantly, *attend* simultaneously to the whole dimension, preventing any locality assumptions or size restrictions. To simplify the up and down scaling procedures contained in the U-Net, we restrict here to tensor encodings of lengths (time dimension) product of 4 (see Extended Data Fig. 2b). In practice we allow

all circuit lengths and add padding tokens to the next higher multiple of 4.

Some conditions require an absolute position matching to the corresponding qubits (e.g. the elements of the SRVs strictly define which qubit has which Schmidt rank). We account for this by adding an absolute 2D sinusoidal position encoding (Vaswani et al., 2017) to the model input. To get a 2D position, we split the gate dimension in two chunks and apply on each a distinct encoding for the corresponding x and y coordinate. Inspired by (Kazemnejad et al., 2023), we tried different approaches: no positional encoding in either the spatial or time dimension, or introducing in both dimensions a causal attention mask. We noticed a significant accuracy drop when not using a 2D position encoding. We suspect that the application of the causal attention mask prevents the model seeing which gates it places in the subsequent time steps of the circuit. As any gate withing the circuit is able to alter the output state significantly, not having access to such information presents an insuperable challenge.

Text prompt encoding – As described in the main text, we use a CLIP model (Radford et al., 2021) with pre-trained weights from (Ilharco et al., 2023) (version *ViT-B-32* trained on *laion2b_s34b_b79k*) to encode the text prompts. We discard the last layer and take the penultimate transformer output as encoding. The CLIP pipeline pads tokens to a sequence length of 77 elements. Since the last layer is discarded, the output-tensor has 512 channels i.e., a final shape of 77×512.

Unitary encoder – Given a unitary matrix, we create an input tensor by splitting the unitary's real and imaginary parts into two channels, which is fed into the unitary encoder. Extended Data Fig. 3b schematically represents the used architecture. After an initial convolution layer, we introduce a 2D positional encoding layer, such as to encode the absolute position of the unitary entries. Then, we use a transformer encoder with self-attention layers, taking advantage of their global attention mechanism. Such global attention is key when inspecting a unitary matrix, as the information contained on it is highly non-local. We introduce in between the two attention blocks a down-scaling layer with kernel-size 2x2. The output tensor has size $l \times l \times 512$, where l depends on the size of the input unitary.

Conditioning – A key feature of the proposed model is its ability to generate quantum circuits based on a given conditioning. As usually done in this type of models, the input condition $\mathbf{c}$ is passed into the key and value inputs of the cross-attention layers of the U-Net. In our case, $\mathbf{c}$ can contain multiple information. In simple scenarios, as in entanglement generation, the CLIP encoded text prompt is the only condition. When performing unitary compilation, we concatenate the output of the CLIP encoder (size 77×512) with the flattened output of the unitary encoder (size $l^2 \times 512$), meaning a final conditioning of size $77 + l^2 \times 512$. Last, the denoising time step t (see below for details), encoded via a sinusoidal position encoding, is added to the residual part of the residual convolutions blocks inside the U-Net.

D. Training

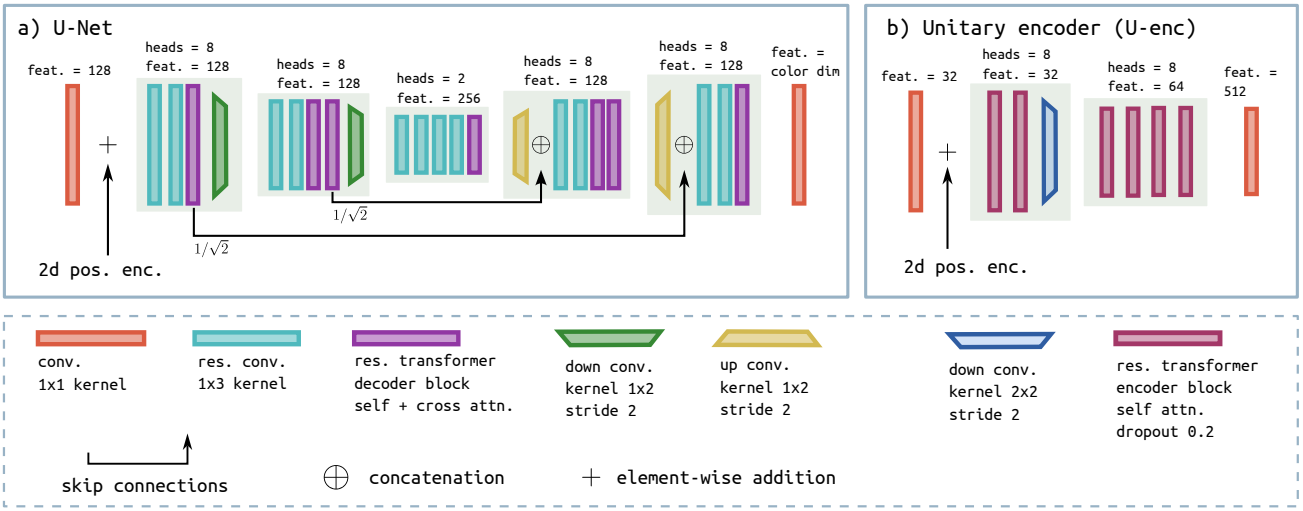
In this section we present the details needed to reproduce the training of the presented models. Our models were trained using a single Nvidia RTX A6000 Ada graphical processing unit and an AMD Ryzen Threadripper Pro 5965wx processor. The training times vary between 12 to 24 hours depending on the particular application.

We train the DM according to the denoising diffusion probabilistic models (DDPM) procedure (Ho et al., 2020). We choose to parameterize the model as an ϵ-predictor, i.e., at every time step t the model learns to directly predict the noise $\boldsymbol{\epsilon}_t$ of a noisy tensor $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$, $\mathbf{x}_0$ is a sample of a training dataset with an arbitrary distribution $q(\mathbf{x}_0)$ and $\bar{\alpha}_t$ is a variance schedule. Here we set $\bar{\alpha}_t = \prod_{i=0}^{t} \alpha_i = \prod_{i=0}^{t}(1-\beta_i)$. Considering a denoising model with parameters θ, whose noise prediction $\boldsymbol{\epsilon}_\theta$ is conditioned on $\mathbf{c}$, its parameters are optimized to minimize the loss function

$$\mathcal{L} = \mathbb{E}_{t\sim\mathcal{U}[0,T],\, \mathbf{x}_0\sim q(\mathbf{x}_0),\, \boldsymbol{\epsilon}_t\sim\mathcal{N}(0,\mathbf{I})} \left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c})\|_2^2 \right].$$

In order to sample according to the given condition, we use classifier-free-guidance (CFG) (Ho & Salimans, 2022). During training, we set the condition $\mathbf{c}$ to an empty token $\varnothing$ with a probability of 10%, to train the model both for conditional and unconditional predictions. To reduce the exposure bias of the train-inference mismatch, we apply input perturbation (Ning et al., 2023) with $\gamma = 0.1$. As variance schedule, we use the cosine beta schedule from (Nichol & Dhariwal, 2021). We train for $T = 1000$ diffusion steps and utilize the Adam optimizer (Kingma & Ba, 2014) with one-cycle learning rate policy (Smith & Topin, 2019). All further training parameters are listed in Extended Data Table 2.

For the compilation task, we simultaneously train the unitary encoder with the U-Net, i.e., treating the corresponding part of the condition as $\mathbf{c}_\Phi(U)$, where U is a given unitary and Φ are the parameters of the unitary encoder. The CLIP model for the text prompt encoding is kept frozen. In our dataset, the majority of the unitaries have a single unique circuit implementation



Extended Data Figure 3. **Machine learning architectures.** **(a)** Scheme of the denoising U-Net architecture predicting the noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c})$. First, we project the input tensor features into a higher space through a convolutional layer (red) and then apply a 2D positional sinusoidal encoding. Then, we apply a typical encoder-decoder structure, with skip connections scaled with $1/\sqrt{2}$. The time step encoding t is injected into residual convolution layers (turquoise). The condition embeddings $\mathbf{c}$ are input to the residual transformer blocks (purple) as detailed in Appendix C. All the transformer blocks have a residual connection. **(b)** Scheme of the unitary encoder used to transform input unitaries into conditionings (see Appendix C)

due to extensive random sampling in the unitary space. This results in over-fitting, as the model can just memorize a one-to-one mapping of the unitary to a tensor encoding. We mitigate this by implementing dropout in the U-enc. Despite some overfitting during training, the model performs well when compiling new unitaries, as we show in Figure 4. For the entanglement generation task, training the model shows a steady decrease in the validation loss, resulting from many circuits that correspond to the same "overloaded" SRV (see Method's Dataset section).

A general quantum circuit synthesizer must be able to generate circuits with different sizes, i.e., different number of qubits and max number of gates. Translating this to image generation essentially means producing images with different sizes and aspect ratios. Tackling such problem is an ongoing research topic, which has recently been addressed by Stable Diffusion XL (Podell et al., 2023). Image generation datasets consist of many images with different sizes. Conventionally, all images are cropped, padded, resized or even discarded to one global size. Stable Diffusion XL introduces two procedures to mitigate this issue: micro-conditioning, i.e., conditioning on the original size of the images, and multi-aspect (ratio) fine-tuning. In the presented model, we implement the latter in a two step process, as proposed in Ref. (Podell et al., 2023):

1. **Max padding** – First, we train on a single tensor size. For this, we calculate the maximum tensor size of the whole dataset (i.e., the most qubits and, separately, the highest number of gates). We then apply padding to all circuits smaller than this maximum size. In order to pad, we use an additional "padding" token and apply the same embedding method as for gates (as mentioned in the Quantum circuit encoding section and shown in Extended Data Fig. 1b).

2. **Bucket padding** – After that, we split the dataset into *buckets*, each containing circuits of a given qubit number. Then, the model is fine-tuned with batches from such buckets, ensuring that all circuits within a batch have the same number of qubits, but still allowing these to have varying lengths. Our experiments show a great improvement in the accuracy of the model when this last step was implemented.

Importantly, for both training steps and for each training batch, we cut the circuits to the longest circuit in the time dimension within the batch, removing unnecessary length padding and speeding up calculations. We believe that the model seeing the padding token for circuits smaller than the longest one within the batch is beneficial, as it can learn that the number of actual gates can be smaller than the tensor size.

model	step	method	lr	epochs
SRV 3 to 8 qubits	1	max pad	$3 \cdot 10^{-4}$	75
	2	bucket pad	$5 \cdot 10^{-5}$	50
Fine-tune to 9 or 10 qubit		bucket pad	$5 \cdot 10^{-5}$	25
Compilation 3 qubits		bucket pad	$3 \cdot 10^{-4}$	150

Extended Data Table 2. **Training parameters.** For the multi-qubit entanglement generation task (referred here as SRV), we use two training steps as explained in Methods, Training section. The fine-tuning and compilation trainings are done with circuits with fixed qubit numbers (either 9 and 10 for entanglement generation or 3 for unitary compilation). Hence, no special padding is needed in the space (qubit) dimension. Length padding is done per training batch. For all trainings we use a batch size of 256.

Task	qubits	max gates	gate-pool	samples	denoise steps	t_{start}	g	Notes
Figure 2a, d, e	3 to 8	16	H, CX	8192	20		7.5	samples per # entangled qubits
Figure 2b	5	16	H, CX	8192	20		7.5	samples per SRV
Figure 2c	9 to 10	16	H, CX	512	20		7.5	samples per # entangled qubits, 10 fine-tune runs
Figure 2c inset	9 to 10	16	H, CX	16384	20		7.5	samples per # entangled qubits, 10 fine-tune runs
Figure 2f	5	4 to 52	H, CX	8192	20		7.5	samples per # entangled qubits per max gates
Figure 3a	7	16	H, CX		40	39	7.5	no initial gates
Figure 3b	5	16	H, CX		40	39	7.5	one initial circuit with 6 gates
Figure 3c	5	20	H, CX	1024	40	38	7.5	256 initial circuits with 5 to 6 gates
Figure 4a, b	3	12	H, CX, Z, X, CCX, SWAP	1024	20		7.5	samples per unitary; 3100 unitaries with 2 to 12 gates

Extended Data Table 3. **Sampling parameters.** Parameters for all the results reported throughout this paper. Here g is the CFG guidance-scale. For editing and masking t_{start} is the time step from which we start denoising (analogous to img2img (Meng et al., 2022)).

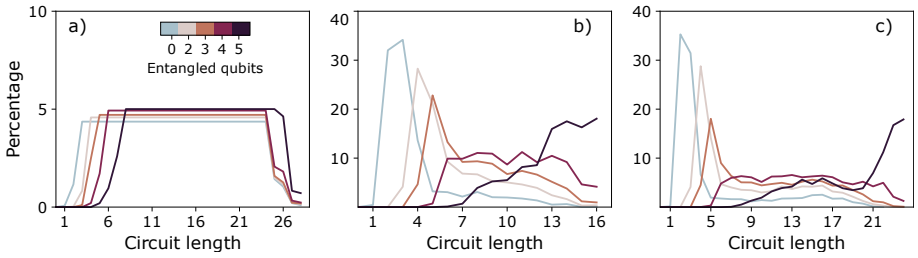
D.1. Fine tuning

In Figure 2c we showed that a trained model can be fine-tuned after training and learn to solve new tasks. In that case, we showcased how it is possible to improve the accuracy of the model in bigger qubit counts: from the original 3 to 8 qubit model to 9 and 10 qubits. To do so, we perform an extra step of the bucket padding step, with circuits of 9 and 10 qubits, with parameters given in Extended Data Table 2. In this case, we consider each qubit count as a completely different training. It is expected that, due to the fine-tuning, the accuracy of the model on the original task (e.g., 3 to 8 qubit circuits) may decrease. Here, we focused only on improving the accuracy of the new tasks. If one wants to keep the performance of old tasks, a mixed-qubit training can be used, where the new expensive circuits are combined with a dataset of already know tasks, e.g., the latter can be automatically generated by the model beforehand, as proposed by DreamBooth (Ruiz et al., 2023).

E. Inference and benchmark

Once trained, we sample (infer) new circuits from the model in a variety of scenarios. To sample a new circuit we provide the model with condition $\mathbf{c}$ and a noise input tensor $\tilde{\mathbf{x}}_T \sim \mathcal{N}(0, \mathbf{I})$. The noise tensor imposes the size of the output circuits, i.e., the number of qubits and the maximal number of gates the model is able to place. As the model has seen a variety of padded circuits during training, it is able to place the padding token if needed. Hence, a good strategy is to input sufficiently long tensors to the machine and allow the model to restrict the total number of gates by placing the padding token. We believe such feature underlies the stability of the accuracy over increasing tensor sizes shown in Figure 2f.

As sampling method, we use the DDPM sampler (Ho et al., 2020), with lower denoising steps as proposed from the denoising diffusion implicit model (DDIM) (Song et al., 2022). We use the re-scaled classifier-free-guidance formula (Sec. 3.4 of Ref. (Lin et al., 2023) with $\phi = 0.7$) to guide the model according to the condition. The number of denoising steps and the guidance scales used to generate the datasets presented in the figures of this manuscript are listed in Extended Data Table 3.



Extended Data Figure 4. **Generated circuit lengths distributions.** Distribution of circuit lengths w.r.t. to the number of entangled qubits for: **(a)** the training (balanced) dataset of Figure 2 filtered for 5 qubit circuits; **(b-c)** generated circuits with an input tensor constraining a maximum of 16 and 24 gates, respectively.

We note that the masking task presented in Figure 3a represents a bigger challenge than the rest of the tasks, due to hard restrictions imposed on the generated circuits. Indeed, we see an increase of the number of error circuits, from below 1% in the non-masked entanglement generation task to $\sim 90\%$ here. Moreover, the number of samples needed to find a proper solution will vary depending on how restrictive the masking is. In contrast, the editing task presented in Figure 3b is not as restrictive. Hence, the number of error circuits does not increase in the case and the model is able to find multiple distinct solutions with less needed samples.

F. Circuit length vs. SRV

Machine learning models often tend to exploit features of the training data to improve their accuracy. In this case, our experiments show that the model correlates the length of the generated circuits with the number of entangled qubits (Extended Data Fig. 4). While we balance the circuit length in the training set (see Extended Data Fig. 4a and Dataset section above), the generated circuits (Extended Data Fig. 4b and c) show peaked distributions, whose peak increases as the number of entangled qubits increase. As shown in the figure, this feature appears for any tensor size. However, we see that reducing the tensor size slightly improves this feature. Indeed, further investigations revealed that the peaks in the length distribution correspond to lengths in which randomly generated circuits would have the best accuracy. We can then consider that while training, the model found that producing such distribution is a shortcut to decreasing the loss function and hence biased its prediction for this. Different dataset balancing approaches were tested, all yielding similar results. While such preference biases the generation, the final length of the circuit can still be tailored by means of the input tensor size and the model can still produce circuits of all lengths for each SRV (if possible).