Quantum Dynamics by Solving Probabilistic Differential Equations via Autoregressive Networks

Di Luo[*]
Department of Physics,
University of Illinois, Urbana-Champaign,
Urbana, IL, 61801, USA.
diluo2@illinois.edu

Zhuo Chen[†]
Department of Physics,
University of Illinois at Urbana-Champaign,
Urbana, IL, 61801, USA.
zhuoc3@illinois.edu

Juan Carrasquilla
Vector Institute for Artificial Intelligence,
MaRS Centre,
Toronto, Ontario, M5G 1L7, Canada.
carrasqu@vectorinstitute.ai

Bryan K. Clark
Department of Physics,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA.
bkclark@illinois.edu

Abstract

Quantum mechanics is fundamental to modern science and technology. Given the high dimensional nature of quantum mechanics, simulating quantum systems requires a large amount of computational power, which demands algorithms to efficiently approximate such systems. In this paper, we apply a transformation that maps quantum dynamics to classical probabilistic differential equations. We then parameterize the probability distribution with autoregressive neural networks, which allows us to design efficient stochastic algorithms to simulate quantum evolution and solve for steady-state solutions.

1 Introduction.

The laws that specify the properties of materials and devices around us are quantum mechanical in nature. Our ability to simulate quantum mechanics is important both for making predictions about physical phenomena as well as for engineering quantum devices and computers [1, 26]. Although simulating quantum systems simply requires solving a first order differential equation, the computational complexity scales exponentially with system size, making it intractable for all but the smallest problems.

A canonical approach to approximately overcome this problem has been to develop a compact representation of a quantum state and then to update this representation under the rules of dynamic evolution. Historically, these compact representations have been motivated by dressed versions of exactly solvable physical systems. More recently, neural networks have been used to represent wave functions [2, 7, 8, 10, 11, 18, 19, 21, 23, 24]. Naively, a natural choice for such a compact representation might be a generative model; unfortunately, the fact that the standard representation of a quantum state can't be viewed as a probability distribution precludes this as a direct approach.

In this work, we simulate quantum dynamics using an approach which maps the quantum state to a probability distribution via positive operator-valued measure (POVM) [3, 4, 20]. This mapping turns the differential equations governing the evolution of quantum states to a classical first order differential

[*]Co-first authors
[†]Co-first authors

equations on probabilities. We parameterize the probability distribution with an autoregressive model and describe an algorithm which updates the parameters of this model under the application of quantum dynamics. This extended abstract investigates how the accuracy of this approach is affected by both algorithmic parameters (such as the size of the transformer) as well as physical parameters (such as the amount of dissipation in the system).

2 Mapping Quantum Mechanics to a Probabilistic Theory

The most general quantum mechanical dynamic evolution corresponds to time evolving a quantum system coupled to an environment, i.e. an open system. The time evolution of a generic open quantum system is described by the first order differential equation,

$$\frac{\partial \rho}{\partial t}(t) = \mathcal{L}\left[\rho(t)\right], \tag{1}$$

where $\mathcal{L}$ is a linear operator acting on the positive semi-definite (trace-one) density matrix ρ which is exponentially large in the system size n. The density matrices ρ can be mapped to a probability distributions $p(\boldsymbol{a}) \equiv p(a_1, a_2, \cdots, a_n)$ using informationally complete POVM, where $a_i \in \{0,1,2,3\}$ can be viewed as the measurement outcome for each qubit [3]. This mapping transforms Eq. 1 into the probabilistic equation

$$\frac{\partial p_t}{\partial t}(a_1, a_2, a_3, \cdots, a_N) = \sum_{b_1, b_2, b_3, \cdots, b_N} p_t(b_1, b_2, b_3, \cdots, b_N) L_{a_1, a_2, a_3, \cdots, a_N}^{b_1, b_2, b_3, \cdots, b_N}, \tag{2}$$

While this equation is general, L is system-specific. In this work, we consider the transverse-field Ising model coupled to a Markovian bath; for that system,

$$L_{\boldsymbol{a}}^{\boldsymbol{b}} = -i \operatorname{Tr}\left(\mathcal{H}[N^{(\boldsymbol{b})}, M_{(\boldsymbol{a})}]\right) + \sum_k \frac{\gamma_k}{2} \operatorname{Tr}\left(2\Gamma_k N^{(\boldsymbol{b})} \Gamma_k^\dagger M_{(\boldsymbol{a})} - \Gamma_k^\dagger \Gamma_k \{N^{(\boldsymbol{b})}, M_{(\boldsymbol{a})}\}\right), \tag{3}$$

where $[\cdot,\cdot]$ refers to commutator and $\{\cdot,\cdot\}$ refers to anticommutator. Here the Hamiltonian $\mathcal{H}$ is defined as $\mathcal{H} = J\sum_{\langle i,j\rangle} \sigma_i^{(z)} \sigma_j^{(z)} + h\sum_k \sigma_k^{(x)}$, where $\sigma_i^{(\alpha)}$ with $\alpha = x, y, z$ are Pauli matrices and the pair of angled brackets refers to neighboring sites. The jump operators are chosen to be $\Gamma_k = \sigma_k^{(-)} = \frac{1}{2}(\sigma_k^{(x)} - i\sigma_k^{(y)})$. We choose a set of factorized operators $\{M_{(\boldsymbol{a})}\} = \{M_{(a_1)} \otimes M_{(a_2)} \otimes M_{(a_3)} \otimes \cdots\}$ which leads to $\{N^{(\boldsymbol{b})}\} = \{N^{(b_1)} \otimes N^{(b_2)} \otimes N^{(b_3)} \otimes \cdots\}$, where $M_{(a_i)}$ are four 2×2 positive semi-definite matrices and $N^{(b_i)}$ are four 2×2 Hermitian matrices [3].

A crucial property of L for the success of the techniques in this paper, is that, for each $\boldsymbol{a}$ the scalar $L_{\boldsymbol{a}}^{\boldsymbol{b}}$ is non-zero only on a small (i.e. polynomial in N) set of $\boldsymbol{b}$ and those non-zero terms can be quickly computed given $\boldsymbol{a}$.

3 Solving Quantum PDE as a High Dimensional Probabilistic PDE

Notice that Eq. 2 is a high dimensional probabilistic PDE with solution $p_t(\boldsymbol{a}) = \sum_{\boldsymbol{b}} p_0(\boldsymbol{b}) \exp\left(tL_{\boldsymbol{a}}^{\boldsymbol{b}}\right) \equiv \sum_{\boldsymbol{b}} p_0(\boldsymbol{b}) S_t(\boldsymbol{a}, \boldsymbol{b})$. For real-time evolution, the matrix $S_t(\boldsymbol{a}, \boldsymbol{b})$ is a quasi-stochastic matrix with both positive and negative values but whose sum $\sum_{\boldsymbol{b}} S_t(\boldsymbol{a}, \boldsymbol{b}) = 1$. We propose a general algorithm for solving Eq. 2 using the forward-backward trapezoid method [13] with exact sampling, which is analogous to the cases of the IT-SWO [17] and the real-time wavefunction dynamics [9] with Monte Carlo sampling. The solution is obtained iteratively by setting

$$\sum_{\boldsymbol{b}} \left(\delta_{\boldsymbol{a}}^{\boldsymbol{b}} - \tau L_{\boldsymbol{a}}^{\boldsymbol{b}}\right) p_{t+2\tau}(\boldsymbol{b}) = \sum_{\boldsymbol{b}} \left(\delta_{\boldsymbol{a}}^{\boldsymbol{b}} + \tau L_{\boldsymbol{a}}^{\boldsymbol{b}}\right) p_t(\boldsymbol{b}), \tag{4}$$

where $\delta_{\boldsymbol{a}}^{\boldsymbol{b}}$ is the Kronecker delta function and τ is the step size. It is challenging to solve Eq. 4 directly from matrix operations as the operators on both sides grow exponentially with respect to the number of particles in the system. Instead, we parameterize the probability distribution with an autoregressive neural network as $p_{\theta(t)}$ and design a cost function which solves for $p_{\theta(t+2\tau)}$ given $p_{\theta(t)}$. This parameterization allows the configuration $\boldsymbol{a}$ to be sampled exactly and efficiently. Therefore, at each step in time, one needs two neural networks, $p_{\theta(t)}$ and $p_{\theta(t+2\tau)}$, where $p_{\theta(t)}$ is the pretrained neural network at the last step. Then the cost function can be defined as

$$\mathcal{C} = \frac{1}{N_s} \sum_{\boldsymbol{a} \sim p_{\theta(t+2\tau)}}^{N_s} \frac{1}{p_{\theta(t+2\tau)}(\boldsymbol{a})} \left| \sum_{\boldsymbol{b}} \left[p_{\theta(t+2\tau)}(\boldsymbol{b}) \left(\delta_{\boldsymbol{a}}^{\boldsymbol{b}} - \tau L_{\boldsymbol{a}}^{\boldsymbol{b}} \right) - p_{\theta(t)}(\boldsymbol{b}) \left(\delta_{\boldsymbol{a}}^{\boldsymbol{b}} + \tau L_{\boldsymbol{a}}^{\boldsymbol{b}} \right) \right] \right|, \quad (5)$$

where the sum is over $\boldsymbol{a}$ sampled stochastically from $p_{\theta(t+2\tau)}$. We determine the autoregressive model at the each time step using the following algorithm. At $t = 0$, we initialize an autoregressive model with some initial probability distribution (corresponding to the initial distribution of our quantum system). In some cases this can be done analytically; otherwise, one should do it with quantum tomography [4, 5]. To generate the next step, one samples $\boldsymbol{a}$ from the autoregressive neural network at time $t + 2\tau$. Given $\boldsymbol{a}$, we generate all $\boldsymbol{b}$ that are non-zero in $L_{\boldsymbol{a}}^{\boldsymbol{b}}$. Because $p_{\theta(t+2\tau)}(\boldsymbol{b})$ and $p_{\theta(t)}(\boldsymbol{b})$ can be evaluated explicitly in an autoregressive model, the summand of the objective function $\mathcal{C}$ can be evaluated for this sampled $\boldsymbol{a}$. We use automatic differentiation in Pytorch [22] with respect to $p_{\theta(t+2\tau)}(\boldsymbol{b})$ to optimize $\mathcal{C}$ and find the updated Transformer parameters for the next step.

One interesting aspect of the dynamics is to understand properties of the system when $T \to \infty$. In this limit, as long as $\gamma_k \neq 0$, the system reaches a fixed point. While one approach to solve for the fixed point is to simply run the dynamics for an extended time, an alternative approach is to solve directly for the fixed point. The fixed point can be obtained by setting $\sum_{\boldsymbol{b}} p(\boldsymbol{b}) L_{\boldsymbol{a}}^{\boldsymbol{b}} = 0$, which solves for the null space of $L_{\boldsymbol{a}}^{\boldsymbol{b}}$ by minimizing

$$\left\| \frac{\partial p}{\partial t} \right\|_1 \approx \frac{1}{N_s} \sum_{\boldsymbol{a} \sim p_\theta}^{N_s} \frac{\left| \sum_{\boldsymbol{b}} p_\theta(\boldsymbol{b}) L_{\boldsymbol{a}}^{\boldsymbol{b}} \right|}{p_\theta(\boldsymbol{a})}, \quad (6)$$

where, again, the sum is over $\boldsymbol{a}$ sampled stochastically from p_θ, and the gradient is taken with respect to $p_\theta(\boldsymbol{b})$.

4 Autoregressive Transformer

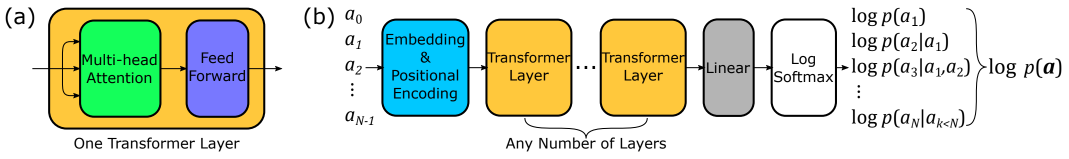

Figure 1: Diagram of the Transformer. (a) Diagram of a single transformer layer, which consists of a self multi-head attention layer and a feed forward layer. (b) Diagram of the full autoregressive model, which consists of one or more of the single transformer layers, together with an embedding layer, a positional encoding layer, a linear layer and a log softmax layer.

In this section, we describe the Transformer architecture used to parameterize the distribution, as well as the evaluation and the sampling processes.

The probability distribution of a given configuration $\boldsymbol{a}$ is modeled as conditional probabilities $p_\theta(\boldsymbol{a}) = \prod_k p_\theta(a_k|a_1, a_2, \cdots, a_{k-1})$ with an autoregressive neural network. The configurations can be sampled exactly from the probability distribution, which allows for fast and accurate sampling compared with Markov chain Monte Carlo techniques. We choose to use an autoregressive Transformer (Fig 1), but other autoregressive models including recurrent neural networks (RNN) [6, 12] and Pixel Convolutional Neural Networks (PixelCNN) [25] are also applicable.

Evaluation: For any sequence $\boldsymbol{a} = \{a_1, a_2, a_3, \cdots, a_N\}$, it is fed into the Transformer together with a default value a_0 (which we choose to be 4). The embedding layer converts each a_i into an embedding vector of length n_d, which, after the positional encoding layer, is fed into the transformer layers. A mask is applied to the multi-head attention to ensure autoregressiveness. The linear layer has a shape of $n_d \times 4$, which converts the output vector from the transformer layers into a vector of length 4 for each site. After the log softmax layer, we get the log of conditional probabilities for each site, which can then be summed over to obtain the overall log probabilities.

Sampling: We first feed a_0 into the Transformer, which gives us $\log p(a_1)$, from which we can sample for the first site and obtain a_1. Then, we can feed both a_0 and a_1 into the Transformer to sample for a_2. This process goes on until we have sampled all sites.

There are two hyper-parameters we adjust in the Transformer: the number of transformer layers stacked on each other n_l and the hidden dimension n_d. It should be noted that both the evaluation process and sampling process can be performed in parallel, i.e. one can evaluate or sample a batch of $\boldsymbol{a}$'s at the same time, which allows for efficient computation.

5 Experimental Results.

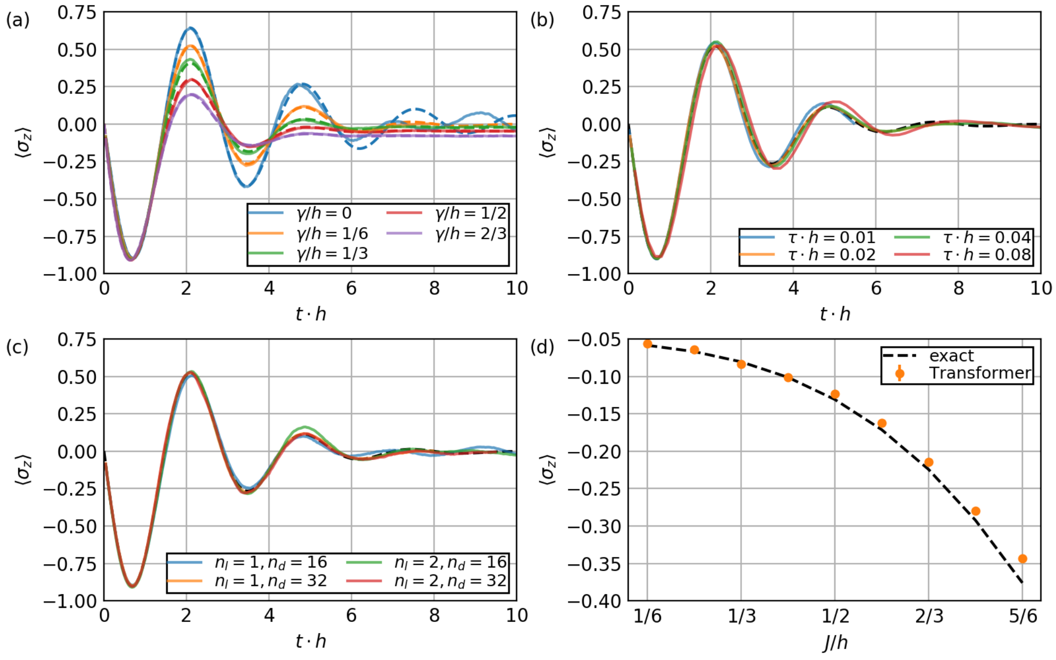

Figure 2: (a,b,c) Values of $\langle\sigma_z\rangle$ as a function of time, (a) for different dissipation γ/h with $J/h = 1/3$, time step $\tau = 0.02$, $n_l = 1$ transformer layers and $n_d = 32$ hidden dimensions for the Transformer, (b) for different time steps τ with $J/h = 1/3$, $\gamma/h = 1/6$, $n_l = 1$, and $n_d = 32$, and (c) for different sizes of Transformers with $J/h = 1/3$, $\gamma/h = 1/6$, and $\tau = 0.02$. (d) Values of $\langle\sigma_z\rangle$ as a function of J/h for the steady-state solution (fixed point) computed using the variational approach with $\gamma/h = 2/3$, $n_l = 1$, and $n_d = 32$. All: During the training process, we choose a sample size of $N_s = 12000$ and use the Adam [16] optimizer to train the neural network until the loss converges (150 iterations in each time step for dynamics and 1000 iterations for variational approach). The exact solutions (dashed lines) are computed using QuTip [14, 15].

In this section, we benchmark the Transformer on a series of examples looking at its efficacy as we tune both algorithmic and physical parameters. We performed both the forward-backward trapezoid integration method using the loss function defined in Eq. 5 and direct minimization of Eq. 6 to find the steady state for $\gamma_k \neq 0$. We then measure the observable $\langle\sigma_z\rangle = 1/N\sum_i\langle\sigma_i^{(z)}\rangle$ as the average of all single site observables.

We start by considering the dependence of $\langle\sigma_z\rangle$ as we change the amount of dissipation γ_k in this system. Larger dissipation corresponds to stronger coupling to the environment and faster decohering of quantum effects. We find (see Fig. 2(a)) that the smaller the dissipation, the quicker there is a deviation between the exact results and the Transformer results. We find that the general behavior is qualitatively correct but there is a clear deviation both in the amplitude as well as frequency of the dynamics. In Fig. 2(b) we modify the time step τ that we use. The exact solution is in the limit $\tau \to 0$. We find the errors here mainly affect the frequency of the oscillation at larger time. In Fig. 2(c), we notice that using different number of layers or hidden dimensions has small effect on the results, presumably because the bottleneck in the quality of our algorithm is not related to representability in this case, i.e. the Transformer with $n_l = 1, n_d = 16$ is already flexible enough to represent the distribution at these various times. Instead, we expect the deviation from the true result may be primarily coming from accumulated errors that come from imperfect optimizations. Finally, in Fig. 2(d) we compute the observable in the limit of large time using the variational approach by minimizing Eq. 6. We find good agreement for these fixed-points from the exact solution.

6 Conclusion

We have utilized a general approach to map quantum mechanical problems into classical probability problems and simulated the dynamics of the probability distribution with an autoregressive Transformer. We then investigate the time evolution for both open and closed quantum systems as well as the fixed point solution for open quantum systems.

Our results closely resemble the exact result for a 10-qubit transverse field Ising model and offer the possibility to scale to larger systems. Our approach could have broad applications for density matrix evolution in different contexts, such as finite temperature quantum evolution.

7 Broad Impacts

The simulation of quantum problems is simultaneously important and difficult, spanning disciplines ranging from chemistry to condensed matter to high energy physics and quantum computing. A large fraction of the supercomputing time throughout the world goes directly toward solving quantum systems. If we could do arbitrary quantum dynamics, this would have direct consequences throughout industry, affecting our ability to develop improved fertilizers, simulate pharmaceuticals, etc. While many approximate methods exist, it is important to develop and benchmark new and systematically improvable approaches to quantum dynamics, especially those that treat the realistic situation of open dynamics. In this work, we demonstrate an improved way of utilizing state-of-the-art machine learning methodologies to significantly improve the simulation for quantum dynamics. Beyond the quantum mechanical applications, our work demonstrates how to approximately solve, using autoregressive neural networks, a high-dimensional probabilistic differential equation. Such equations appear in a wide variety of classical contexts and our work represents an important step forward in the solution of these problems.

References

[1] J. T. Barreiro, M. Müller, P. Schindler, D. Nigg, T. Monz, M. Chwalla, M. Hennrich, C. F. Roos, P. Zoller, and R. Blatt. An open-system quantum simulator with trapped ions. *Nature*, 470(7335):486–491, Feb 2011.

[2] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, Feb 2017.

[3] J. Carrasquilla, D. Luo, F. Pérez, A. Milsted, B. K. Clark, M. Volkovs, and L. Aolita. Probabilistic simulation of quantum circuits with the transformer, 2019.

[4] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

[5] P. Cha, P. Ginsparg, F. Wu, J. Carrasquilla, P. L. McMahon, and E.-A. Kim. Attention-based Quantum Tomography. *arXiv:2006.12469 [quant-ph]*, June 2020.

[6] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[7] X. Gao and L.-M. Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8(1):662, Sep 2017.

[8] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Physical Review X*, 8(1), Jan 2018.

[9] I. L. Gutiérrez and C. B. Mendl. Real time evolution with neural-network quantum states, 2020.

[10] J. Hermann, Z. Schätzle, and F. Noé. Deep neural network solution of the electronic schrödinger equation, 2019.

[11] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla. Recurrent neural network wave functions. *Phys. Rev. Research*, 2:023358, Jun 2020.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[13] A. Iserles. *Euler's method and beyond*, page 8–13. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition, 2008.

[14] J. Johansson, P. Nation, and F. Nori. Qutip: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183(8):1760 – 1772, 2012.

[15] J. Johansson, P. Nation, and F. Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234 – 1240, 2013.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[17] D. Kochkov and B. K. Clark. Variational optimization in the ai era: Computational graph states and supervised wave-function optimization, 2018.

[18] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural network methods in quantum mechanics. *Computer Physics Communications*, 104(1):1–14, Aug. 1997.

[19] S. Lu, X. Gao, and L.-M. Duan. Efficient representation of topologically ordered states with restricted boltzmann machines. *Phys. Rev. B*, 99:155136, Apr 2019.

[20] D. Luo, Z. Chen, J. Carrasquilla, and B. K. Clark. Autoregressive neural network for simulating open quantum systems via a probabilistic formulation, 2020.

[21] D. Luo and B. K. Clark. Backflow transformations via neural networks for quantum many-body wave functions. *Phys. Rev. Lett.*, 122:226401, Jun 2019.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[23] D. Pfau, J. S. Spencer, A. G. de G. Matthews, and W. M. C. Foulkes. Ab-initio solution of the many-electron schrödinger equation with deep neural networks, 2019.

[24] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.*, 124:020503, Jan 2020.

[25] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[26] F. Verstraete, M. M. Wolf, and J. Ignacio Cirac. Quantum computation and quantum-state engineering driven by dissipation. *Nature Physics*, 5(9):633–636, Sep 2009.