Tensor Network Contractions for Simulating Quantum Circuits on FPGAs

Maksim Levental

May 6, 2021

Abstract

Most research in quantum computing today is performed against simulations of quantum computers rather than true quantum computers. Simulating a quantum computer entails implementing all of the unitary operators corresponding to the quantum gates as matrices. For high numbers of qubits, performing matrix multiplications for these simulations becomes quite expensive, since n-qubit operators correspond to 2^n-dimensional matrix representations. One way to accelerate such a simulation is to use field programmable gate array (FPGA) hardware to efficiently compute the matrix multiplications. Though FPGAs can efficiently perform matrix multiplications they are memory bound, having relatively small block random access memory. One way to potentially reduce the memory footprint of a quantum computing system is to represent it as a tensor network; tensor networks are a formalism for representing compositions of tensors wherein economical tensor contractions are readily identified. Thus we explore tensor networks as a means to reducing the memory footprint of quantum computing systems and broadly accelerating simulations of such systems.

Contents

1 Introduction

Quantum computing (QC) refers to the manipulation and exploitation of properties of quantum mechanical (QM) systems to perform computation. QM systems exhibit properties such as superposition and entanglement and clever *quantum algorithms* transform these systems to perform general computation. Unsurprisingly, the technique was intially conceived of as a way to simulate physical systems themselves:

> "... [N]ature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy."

This closing [2] remark from the keynote at the 1st Physics of Computation Conference in 1981, delivered by the late Richard Feynman, cavalierly, but accurately, expresses that initial goal of quantum computing. Although modeling and simulating physical systems on quantum computers remains a thriving area of research we narrow our focus here to QC as it pertains to solving general computational problems. Such problems include unstructured search [4], integer factorization [12], combinatorial optimization [1], and many others. It is conjectured that some quantum algorithms enable quantum computers to exceed the computational power of classical computers [13]. QC systems are composed of so-called quantum bits, or *qubits*, that encode initial and intermediate states of computations (final states are the results of measurement and therefore are encoded on classical bits). Transformations between states are effected by time-reversible *unitary operators.* A formalism for representing quantum computation is the *quantum circuit* formalism, where semenatically related collections of n qubits are represented as *registers* and transformations are represented as gates, connected to those registers by wires, and applied in sequence.

As already mentioned, in hardware, quantum circuits correspond to physical systems that readily exhibit quantum mechnical properties; examples of physical qubits include transmons, ion traps and topological quantum computers [9]. Current state of the art QC systems are termed Noisy Intermediate-Scale Quantum (NISQ) systems. Such systems are characterized by moderate quantities of physical qubits (50-100) but relatively few logical qubits (i.e. qubits robust to inteference and noise). Due to these limitations (and, more broadly, the relative scarcity of functioning QC systems), most research on quantum algorithms is performed with the assistance of simulators of QC systems. Such simulators perform simulations by representing n-qubit circuit as 2^n-dimensional complex vectors and transformations on those vectors as 2^n-dimensional complex matrix-vector multiplication. Naturally, due to this exponential growth, naively executing such simulations quickly become infeasible for $n > 50$ qubits [10], both due to memory constraints and compute time.

A field-programmable gate array (FPGA) is a device designed to be configured by a user into various arrangements of (classical) gates and memory. FPGAs consist of arrays (hence the name) of configurable logic blocks (CLBs), block ram (BRAM), and programmable busses that connect CLBs and RAM into various topologies (see figure 2.3). The CLBs typically contain arithmetic logic units and lookup tables (LUTs), that can be programmed to represent truth tables for many boolean functions. Using high-level description languages (such as VHDL or Verilog) designers specify modules and compose them into circuits (also known as a *dataflows*) that perform arbitrary computation. Typically FPGAs run at lower clock speeds (100-300MHz) than either conventional processors or even graphics processors but nonetheless they excel at latency constrained computations owing to their fully "synchronous" nature (all modules in the same *clock domain* execute simultaneously).At first glance FPGAs seem like a suitable platform for performant simulation of quantum systems when runtime is of the essence. Unfortunately BRAM is one of the more limited resources on an FPGA and therefore it becomes necessary to explore memory reduction strategies for simulations (as well as runtime reduction strategies).

It's the case that matrices are a subclass of a more general mathematical object called a *tensor* and composition of matrices as *tensor contraction*. *Tensor networks* (TNs) are decompositions (i.e. factorizations) of very high-dimensional tensors into networks (i.e. graphs) of low-dimensional tensors. TNs have been successfullly employed in reducing the memory requirements of simulations of QC systems [10]. The critical feature of tensor networks that make them useful for QC is the potential to perform tensor contractions on the smaller tensors in an order such that, ultimately, the memory and compute time requirements are lower than for the matrix-vector multiplication representation. Existing applications of TNs to quantum circuits focus primarily on memory constraints on general purpose computers [3] and distributed environments [7]. Hence, we explore tensor networks as a means of reducing the memory footprint of quantum circuits with particular attention to dimensions of the designs space as they pertain to deployment to FPGAs.

The remainder of this report is organized as follows: section 2 covers the necessary background wherein subsection 2.1 very briefly reviews quantum computation and quantum circuits (with particular focus on aspects that will be relevant for tensor networks and FPGAs), section 2.2 defines tensor networks fairly rigorously and discusses algorithms for identifying optimal contraction orders, section 2.3 discusses the constraints imposed by virtue of deploying to FPGA, section 3 describes our implementation of <TODO>, section 4 reports our results on various empirical experiments, and section 5 concludes with future research directions.

2 Background

2.1 Quantum Computing

We very (very) quickly review quantum computing and quantum circuits as they pertain to our project. For a much more pedagogically sound introduction consult [5]. As already alluded to, quantum computing exploits properties of quantum mechanical systems in order to perform arbitrary computation. The fundamental unit of quantum computation is a qubit, defined as two-dimensional quantum system with state vector ψ an element of a Hilbert space[1] H:

$$\psi := \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. This exhibits the superposition property of the qubit[2]. Collections of qubits have state vectors that represented by the *tensor product* of the individual states of each qubit; for example, two qubits ψ, ϕ have state vector

$$\psi \otimes \phi := \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} \equiv \begin{pmatrix} \alpha\alpha' \\ \alpha\beta' \\ \beta\alpha' \\ \beta\beta' \end{pmatrix}$$

where the second $\otimes$ is the Kronecker product and $\alpha\alpha'$ indicates conventional complex multiplication. Note that the basis relative to which $\psi \otimes \phi$ is represented is the standard basis for $\mathbb{C}^4$ and thus we observe exponential growth in the size of the representation of an n-qubit system. An alternative notation for state vectors is Dirac notation:

$$\psi \equiv \alpha |0\rangle + \beta |1\rangle$$
$$\psi \otimes \phi \equiv \alpha\alpha' |00\rangle + \alpha\beta' |01\rangle + \beta\alpha' |10\rangle + \beta\beta' |11\rangle$$

Of particular import for QC are the *entangled* or *bell states*; they correspond to multi-qubit states, such as

$$\xi = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$$

that cannot be "factored" into component states[3]. Then, naturally, changes in qubit states are represented as unitary[4] matrices U; for example

$$\psi' = U\psi = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} U_{00}\alpha + U_{01}\beta \\ U_{10}\alpha + U_{11}\beta \end{pmatrix}$$

Matrix representations of transformations on multi-qubit states are constructed using the Kronecker product on the individual matrix representations; for example

$$U \otimes V := \begin{pmatrix} U_{00}V & U_{01}V \\ U_{10}V & U_{11}V \end{pmatrix} := \begin{pmatrix} U_{00}V_{00} & U_{00}V_{01} & U_{01}V_{00} & U_{01}V_{01} \\ U_{00}V_{10} & U_{00}V_{11} & U_{01}V_{10} & U_{01}V_{11} \\ U_{10}V_{00} & U_{10}V_{01} & U_{11}V_{00} & U_{11}V_{01} \\ U_{10}V_{10} & U_{10}V_{11} & U_{11}V_{10} & U_{11}V_{11} \end{pmatrix}$$

Here we see again an exponential growth in representation size as a function of number of qubits.

As already alluded to, quantum circuits are a formalism for representing quantum computation in general and algorithms designed for quantum computers in particular. In the quantum circuit formalism qubit

[1] A Hilbert space H is a vector space augmented with an inner product such that, with respect to the metric induced by that inner product, all Cauchy sequences converge.

[2] We say that the qubit is in a superposition of the basis vectors/states.

[3] ξ cannot be factored because there is no solution to the set of equations (for $\alpha, \alpha', \beta, \beta'$)

$$\alpha\alpha' = \frac{1}{\sqrt{2}}, \quad \alpha\beta' = 0, \quad \beta\alpha' = 0, \quad \beta\beta' = \frac{1}{\sqrt{2}}$$

[4] A matrix U is unitary iff $UU^\dagger = U^\dagger U = I$, i.e. it is its own Hermitian conjugate or more simply if it is "self-inverse".

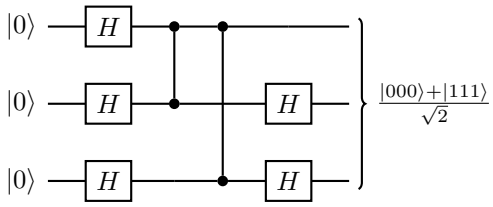

Figure 2.1: Quantum Circuit representing 3-qubit entanglement effected by application of success Hadamard gates.

states are represented by wires and unitary transformations are represented by gates (see figure 2.1), much like classical combinational logic circuits might be, though, whereas combinational logic is "memoryless"[5], sequences of quantum gates specified by a quantum circuit do in fact connote the evolution (in time) of the qubits. In addition quantum gates, as opposed to classical gates, are necessarily reversible and hence there are no quantum analogs to some classical gates such as NOT and OR.

2.2 Tensor Networks

We quickly define tensors and and then move on to tensor network methods.

2.2.1 Definition

One definition of a tensor[6] T is as an element of a tensor product space[7]:

$$T \in \underbrace{V \otimes \cdots \otimes V}_{p \text{ copies}} \otimes \underbrace{V^* \otimes \cdots \otimes V^*}_{q \text{ copies}}$$

where V^* is dual[8] to V. Then T, in effect, acts a multilinear map

$$T : \underbrace{V^* \times \cdots \times V^*}_{p \text{ copies}} \times \underbrace{V \times \cdots \times V}_{q \text{ copies}} \to \mathbb{R}$$

by "applying" p elements from V to p elements of V^* and q elements from V^* to q elements of V. Note the swapping of the orders of V, V^* in both the definitions and the description. T's coordinate basis representation

$$T \equiv T^{i_1 \ldots i_p}_{j_1 \ldots j_q} \, \mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}^{j_1} \otimes \cdots \otimes \mathbf{e}^{j_q} \tag{2.1}$$

is determined by its evaluation on each set of bases

$$T^{i_1 \ldots i_p}_{j_1 \ldots j_q} := T\left(\mathbf{e}^{i_1}, \ldots, \mathbf{e}^{i_p}, \mathbf{e}_{j_1}, \ldots, \mathbf{e}_{j_q}\right)$$

The pair (p, q) is called the *type* or *valence* of T while $(p + q)$ is the *order* of the tensor. **Note that we do not use rank to mean either of these things**. Note further that eqn. 2.1 in fact represents a linear sum of basis elements, as it employs Einstein summation convention[9].

[5]The output of a combinational logic circuit at any time is only a function of the elements of the circuit and its inputs.

[6]There are several more at varying levels of mathematical sophistication. Chapter 14 of [11] is the standard reference. Ironically, it is this author's opinion that one should shy away from physics oriented expositions on tensors.

[7]The collection of tensor products of elements of the component spaces quotiented by an equivalence relation.

[8]The dual space to a vector space V is the vector V^* consisting of linear maps $f : V \to \mathbb{R}$. The dual basis of the dual space consists of f_i such that $f_i(\mathbf{e}_i) = \delta_{ij}$. It is convention to write f_i as $\mathbf{e}^i$ (note the superscript index).

[9]Repeated indices in juxtapose position indicate summation $a_i b^i := \sum_i a[i]b[i]$.

There are two important operations on tensors we need to define. Firstly, we can form the tensor product Z of two tensors T, W, of types $(p,q), (r,s)$ respectively, to obtain a tensor of type $(p+r, q+s)$:

$$\begin{aligned}
Z &:= T \otimes W \\
&= \left(T^{i_1 \dots i_p}_{j_1 \dots j_q}\ \mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}^{j_1} \otimes \cdots \otimes \mathbf{e}^{j_q}\right) \otimes \left(W^{k_1 \dots k_r}_{l_1 \dots l_s}\ \mathbf{e}_{k_1} \otimes \cdots \otimes \mathbf{e}_{k_r} \otimes \mathbf{e}^{l_1} \otimes \cdots \otimes \mathbf{e}^{l_s}\right) \\
&= \left(T^{i_1 \dots i_p}_{j_1 \dots j_q} W^{k_1 \dots k_r}_{l_1 \dots l_s}\ \mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}_{k_1} \otimes \cdots \otimes \mathbf{e}_{k_r} \otimes \mathbf{e}^{j_1} \otimes \cdots \otimes \mathbf{e}^{j_q} \otimes \mathbf{e}^{l_1} \otimes \cdots \otimes \mathbf{e}^{l_s}\right) \\
&:= Z^{i_1 \dots i_{p+r}}_{j_1 \dots j_{q+s}}\ \mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_{p+r}} \otimes \mathbf{e}^{j_1} \otimes \cdots \otimes \mathbf{e}^{j_{q+s}}
\end{aligned}$$

Despite it being obvious, its important to note that the tensor product Z produces a tensor of order $(p+r+q+s)$, i.e. higher than either of the operands. On the contrary, *tensor contraction* reduces the order of a tensor. We define the contraction Y of type (a,b) of a tensor T to be the "pairing" of the ath and bth bases:

$$\begin{aligned}
Y &:= T^{i_1 \dots i_p}_{j_1 \dots j_q}\ \mathbf{e}_{i_1} \otimes \cdots [\mathbf{e}_{i_a}] \cdots \otimes \mathbf{e}_{i_p} \otimes \left(\mathbf{e}_{i_a} \cdot \mathbf{e}^{j_b}\right) \otimes \mathbf{e}^{j_1} \otimes \cdots \left[\mathbf{e}^{j_b}\right] \cdots \otimes \mathbf{e}^{j_q} \\
&= T^{i_1 \dots i_p}_{j_1 \dots j_q} \delta^{j_b}_{i_a}\ \mathbf{e}_{i_1} \otimes \cdots [\mathbf{e}_{i_a}] \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}^{j_1} \otimes \cdots \left[\mathbf{e}^{j_b}\right] \cdots \otimes \mathbf{e}^{j_q} \qquad \left(\text{since } \mathbf{e}_i \cdot \mathbf{e}^j = \delta^j_i\right) \\
&= \left(\sum_{j_b} T^{i_1 \dots j_b \dots i_p}_{j_1 \dots j_b \dots j_q}\right)\ \mathbf{e}_{i_1} \otimes \cdots [\mathbf{e}_{i_a}] \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}^{j_1} \otimes \cdots \left[\mathbf{e}^{j_b}\right] \cdots \otimes \mathbf{e}^{j_q} \\
&:= Y^{i_1 \dots [i_a] \dots i_p}_{j_1 \dots [j_b] \dots j_q}\ \mathbf{e}_{i_1} \otimes \cdots \left[\mathbf{e}_{i_a}\right] \cdots \otimes \mathbf{e}_{i_p} \otimes \mathbf{e}^{j_1} \otimes \cdots \left[\mathbf{e}^{j_b}\right] \cdots \otimes \mathbf{e}^{j_q}
\end{aligned}$$

where $(\cdot)$ means inner product and $[\cdot]$ means omission from sequence. Notice that the order of Y is $(p-1, q-1)$. Finally notice that we can omit writing out bases and just manipulate coordinates. We shall do as such when it simplifies presentation.

As mentioned in the introduction, matrices can be represented as tensors; for example, the two dimensional $n \times n$ matrix M is taken to be a tensor of type $(1,1)$ with basis representation

$$M \equiv M^i_j\ \mathbf{e}_i \otimes \mathbf{e}^j$$

where upper indices correspond to the row index and lower indices correspond to the column index of the conventional matrix representation and both range from 1 to n. The attentive reader will notice that the coordinate representation of the tensor product for type $(1,1)$ tensors is exactly the Kronecker product for matrices. Similarly, tensor contraction for type $(1,1)$ tensors is the familiar matrix trace:

$$M^i_j \left(\mathbf{e}_i \cdot \mathbf{e}^j\right) = M^i_j \delta^j_i = \sum_i M^i_i$$

More usefully, we can express matrix-vector multiplication in terms of tensor contraction; let

$$\mathbf{x} := \begin{pmatrix} x^1 \\ x^2 \end{pmatrix} \equiv x^1 \mathbf{e}_1 + x^2 \mathbf{e}_2 \equiv x^i \mathbf{e}_i$$

where we switch to valence index notation in the column vector for closer affinity with tensor notation. Then it must be the case that

$$\mathbf{y} = M\mathbf{x} = \left(M^i_j \mathbf{e}_i \otimes \mathbf{e}^j\right)\left(x^k \mathbf{e}_k\right) = \left(M^i_j x^k \mathbf{e}_i\right)\left(\mathbf{e}^j \cdot \mathbf{e}_k\right) = M^i_j x^k \delta^j_k \mathbf{e}_i = M^i_j x^j \mathbf{e}_i$$

Letting $y^i := M^i_j x^j$ we recognize conventional matrix-vector multiplication. Employing tensor contraction in this way extends to matrix-matrix multiplication (and tensor composition more broadly); for two type $(1,1)$ tensors M, N we can form the type $(1,1)$ tensor Z corresponding to matrix product $M \cdot N$ of $n \times n$ by first taking the tensor product

$$Z^{ik}_{lj} := M^i_l N^k_j$$

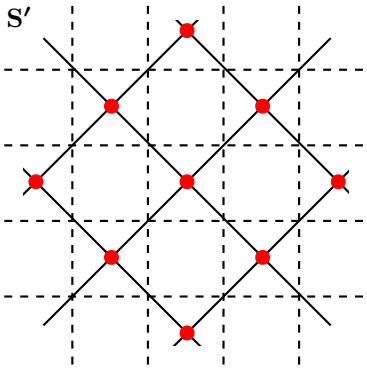

Figure 2.2: Projected Entangled Pair State (PEPS) for a 3 × 3 lattice with open boundary conditions.

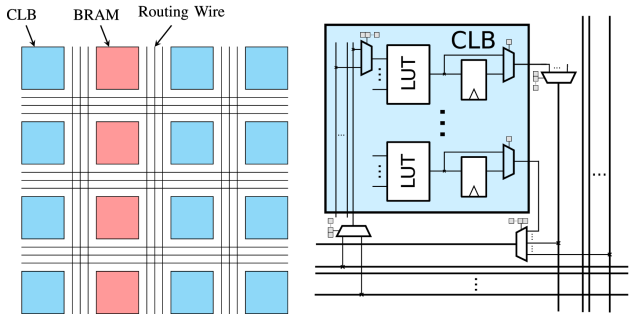

Figure 2.3: FPGA floorplan diagram [8].

and then contracting along the off diagonal

$$Z_j^i := Z_{kj}^{ik} = M_k^i N_j^k \equiv \sum_{k=1}^{n} M_k^i N_j^k$$

One can confirm that this is indeed conventional matrix multipliation of two $n \times n$ matrices.

2.2.2 Tensor Networks

2.3 FPGAs

FPGAs consist of arrays (hence the name) of configurable logic blocks (CLBs), block ram (BRAM), and programmable busses that connect CLBs and RAM into various topologies (see figure 2.3).

3 Implementation

asdsd

4 Evaluation

assd

5 Conclusion

References

[1] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[2] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6/7):467–488, 1982.

[3] E. Schuyler Fried, Nicolas P. D. Sawaya, Yudong Cao, Ian D. Kivlichan, Jhonathan Romero, and Alán Aspuru-Guzik. qtorch: The quantum tensor contraction handler. *PLOS ONE*, 13(12):e0208510, Dec 2018.

[4] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[5] Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, Andreas Bärtschi, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'Malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Patrick J. Coles, Marc Vuffray, and Andrey Y. Lokhov. Quantum algorithm implementations for beginners, 2020.

[6] Igor L. Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM J. Comput.*, 38(3):963–981, June 2008.

[7] Alexander McCaskey, Eugene Dumitrescu, Mengsu Chen, Dmitry Lyakh, and Travis Humble. Validating quantum-classical programming models with tensor network simulations. *PLOS ONE*, 13(12):e0206704, Dec 2018.

[8] K. E. Murray, M. A. Elgammal, V. Betz, T. Ansell, K. Rothman, and A. Comodi. Symbiflow and vpr: An open-source design flow for commercial and novel fpgas. *IEEE Micro*, 40(04):49–57, jul 2020.

[9] National Academies of Sciences Engineering and Medicine. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC, 2019.

[10] Edwin Pednault, John A. Gunnels, Giacomo Nannicini, Lior Horesh, Thomas Magerlein, Edgar Solomonik, Erik W. Draeger, Eric T. Holland, and Robert Wisnieff. Pareto-efficient quantum circuit simulation using tensor contraction deferral, 2020.

[11] S. Roman. *Advanced Linear Algebra*. Graduate Texts in Mathematics. Springer New York, 2007.

[12] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[13] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.