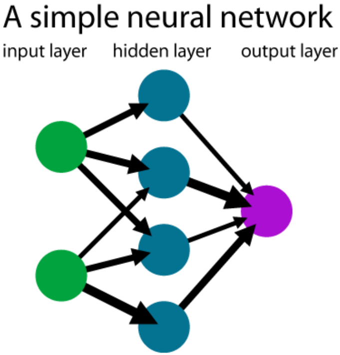
Generative Modeling using Matchgate Circuits

Peter Luo, Hong Ye Hu, Susanne F. Yelin

September 25, 2023

Neural Networks

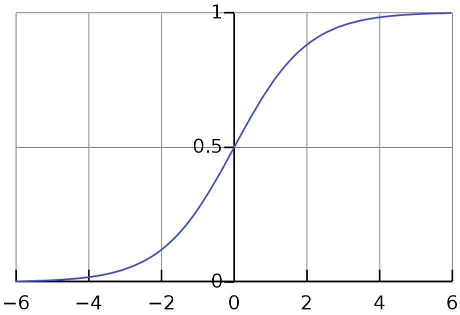
- Neural Networks are a means by which machine learning is carried out, and its architecture is inspired by neurons



- Each neuron will output

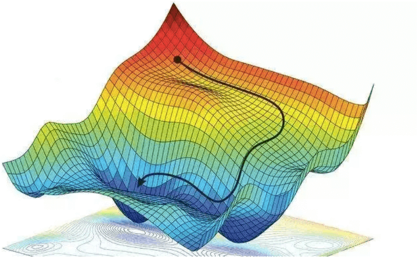
$$f\left(b + \sum w_i x_i\right),$$

where f is an "activation function", the w_i are the "weights", the b is the "bias", and the x_i being the inputs



How Neural Networks Learn

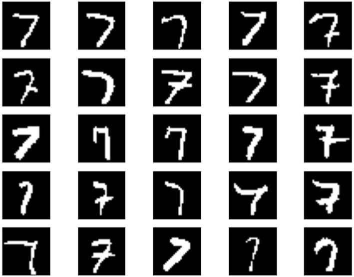
- Define a "loss function" $L(\boldsymbol{w}, \boldsymbol{b})$ which captures how close the network's output is to the desired output



- We can use optimization algorithms to adjust $\boldsymbol{w}$ and $\boldsymbol{b}$ to get closer and closer to the minimum of L

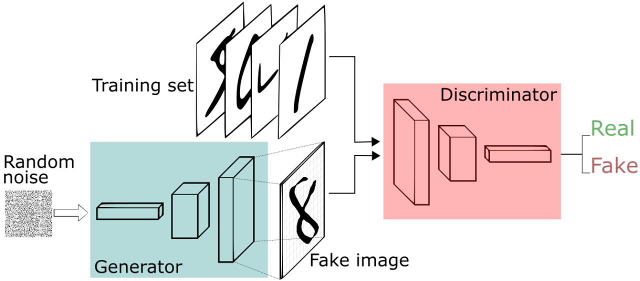
Generative Modeling

- Rather than doing a classification or prediction task, generative models strive to generate more instances of their training data
- Examples: DALL-E, Voice Generators
- Goal of this project: To use a Matchgate circuit and the MNIST dataset to generate new images of handwritten digits



Generative Adversarial Networks (GAN)

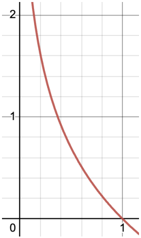
- Consists of two networks G and D with parameters $\boldsymbol{\theta}$ and ϕ and their own loss functions $L_G(\boldsymbol{\theta}, \phi)$ and $L_D(\boldsymbol{\theta}, \phi)$
- $G : \{0,1\}^{28\times 28} \to \{0,1\}^{28\times 28}$ and $D : \{0,1\}^{28\times 28} \to [0,1]$



- Assume that the real data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ comes from an underlying probability distribution p_{real}, and recall that the input to G are samples $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ from p_{prior}
- The loss functions are

$$L_G(\boldsymbol{\theta}, \boldsymbol{\phi}) = - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim p_{\text{prior}}} [\log(D(G(\boldsymbol{z})))]$$

$$L_D(\boldsymbol{\theta}, \boldsymbol{\phi}) = - \left(\mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{\text{real}}} [\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\boldsymbol{z} \sim p_{\text{prior}}} [\log(1 - D(G(\boldsymbol{z})))] \right).$$



Quantum Computing

- Leverages principles from quantum mechanics to do computations
- Consequently, it can solve certain problems exponentially faster than classical computers, reducing runtime from millions of years to just a few minutes
- It can also break some commonly used encryption methods, such as RSA or ECC
- It can simulate molecular interactions efficiently, which has implications in drug discovery and the material sciences

Limitations

- Quantum computers need to be kept at very cold temperatures, thus requiring a large amount of energy
- Real quantum devices are noisy
- Number of qubits that we're able to run on the best quantum computers is relatively small (on the order of 10^2)
- A lot is known about what is theoretically possible, but not a lot is known about their actual performance
- Temporary solution: classical simulation

Quantum Computing

Classical computing:

- Consists of logic gates (e.g. AND, NOT, OR)
- One classical bit: a 0 or a 1

Quantum computing:

- Consists of quantum gates, represented by unitary matrices with complex entries
- One quantum bit: a vector of two complex numbers, which describes the probability of being measured in the 0 or 1 state

Quantum Bits

- One qubit looks like

$$a\,|0\rangle + b\,|1\rangle \to \begin{pmatrix} a \\ b \end{pmatrix}$$

- $|a|^2$ and $|b|^2$ are the probabilities so we require $|a|^2 + |b|^2 = 1$
- For two qubits,

$$a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle \to \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

- The number of complex numbers we need to keep track of scales exponentially with the number of qubits
- Quantum computers are hard to simulate classically

Quantum Gates

- An n-qubit gate is a $2^n \times 2^n$ unitary matrix with complex entries
- A matrix U is unitary if $UU^\dagger = I$, where $\dagger$ means the conjugate transpose
- 1-qubit examples:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \qquad \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \qquad \begin{pmatrix}\cos(\theta/2) & -\sin(\theta/2)\\ \sin(\theta/2) & \cos(\theta/2)\end{pmatrix}$$

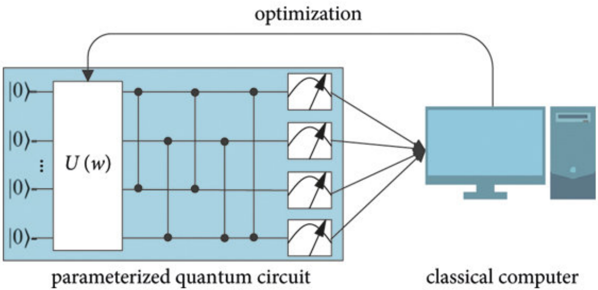
- Example:

$$H\left|0\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{pmatrix} \rightarrow \frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle$$

- To simulate quantum computers is to do matrix multiplications

Parametrized Quantum Circuit

- Quantum gates can be parametrized
- By replacing the neural network with a parametrized quantum circuit and then measuring its output, we obtain a hybrid quantum-classical system compatible with machine learning procedures



Computing Gradients

- Parameter-Shift Rule
- If $L(\boldsymbol{\theta})$ is the expectation value of some quantity with respect to the output of a quantum circuit with gates of the form $U_G(\boldsymbol{\theta}) = e^{-ia\theta G}$ where $G^2 = I$, then

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = r\left(L\left(\boldsymbol{\theta} + \frac{\pi}{4r}\vec{\mathbf{e}}_i\right) + L\left(\boldsymbol{\theta} - \frac{\pi}{4r}\vec{\mathbf{e}}_i\right)\right)$$

- Proof uses

$$U_G(\theta) = e^{-i\theta G} = I\cos\theta - iG\sin\theta$$

and

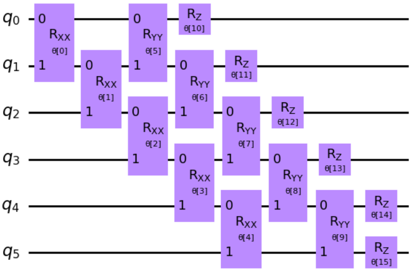
$$\frac{d}{d\theta}U_G(\theta) = -iGe^{-i\theta G} = -iGU_G(\theta)$$

Matchgates

- Matchgates are a restricted class of parametrized quantum gates, generated by

$$e^{-iX\otimes X\theta/2}, \quad e^{-iY\otimes Y\theta/2}, \quad e^{-iX\otimes Y\theta/2}, \quad e^{-iY\otimes X\theta/2}, \quad e^{-iZ\theta/2}$$

- Matchgates are differentiable with respect to θ and can be simulated in polynomial time



Valiant's Definition

- Matchgates were first introduced by Valiant (2001)
- For a given undirected weighted graph G wher $V(G) = [n]$, define the Pfaffian to be

$$\mathrm{Pf}(B) = \sum_{\substack{\text{perfect matchings} \\ M}} \mathrm{sgn}(M) \prod_{(ij)\in M} w_{ij},$$

  where B is the adjacency matrix of G
- Theorem (Cayley): $\mathrm{Pf}(B)^2 = \mathrm{Det}(B)$
- Pfaffians are computable in polynomial time

- Define the Pfaffian sum to be

$$\mathrm{PfS}(B) = \sum_{A \subseteq [n]} \left(\prod_{i \in A} \lambda_i \right) \mathrm{Pf}(B[A])$$

- Pfaffian sums are just Pfaffians:

$$\mathrm{PfS}(B) = \begin{cases} \mathrm{Pf}(B + \Lambda_n) & n \text{ even} \\ \mathrm{Pf}(B^+ + \Lambda_{n+1}) & n \text{ odd} \end{cases}$$

where

$$\Lambda_n = \begin{pmatrix} 0 & \lambda_1\lambda_2 & -\lambda_1\lambda_3 & \cdots \\ -\lambda_1\lambda_2 & 0 & \lambda_2\lambda_3 & \cdots \\ \lambda_1\lambda_3 & -\lambda_2\lambda_3 & \ddots & \\ \vdots & \vdots & & \end{pmatrix}$$

- A matchgate Γ is (G, X, Y, T) where G is a graph and X, Y, T partition the vertices. For $Z \subseteq X \cup Y$, define

$$\chi(\Gamma, Z) = \mu(\Gamma, Z)\mathrm{PfS}(G - Z)$$

- The character matrix $\chi(\Gamma)$ is the $2^{|X|} \times 2^{|Y|}$ matrix with

$$\chi(\Gamma)_{i,j} = \chi(\Gamma, X_i \cup Y_j).$$

- Defined in this way, not all matchgates are unitary
- Take the ones that are unitary and interpret them as quantum gates
- For a Matchgate circuit C,

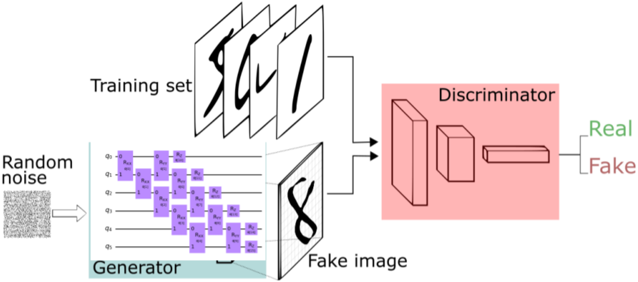
$$|\langle \underline{y}| C |\underline{x}\rangle|^2 = |\mathrm{PfS}(M_{\underline{x},\underline{y}})|^2.$$

Matchgate Speedup

- Valiant: Computing Pfaffians is fast
- Terhal and DiVincenzo (2001): There is a correspondence to noninteracting fermions in 1D, and computing determinants is fast

Setup

1. Use a parametrized quantum circuit made of matchgates as our G
2. Use a classical neural network as D
3. Train it just like you would train a GAN, i.e. use the same loss functions



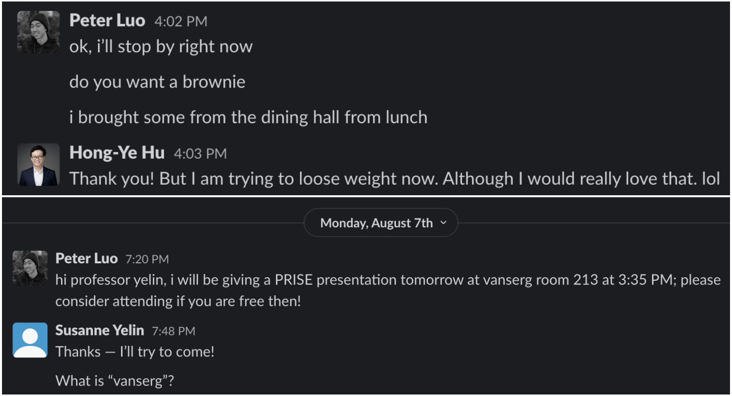
- Goal: Understand the performance of QML on a real dataset with a large number of qubits

Code

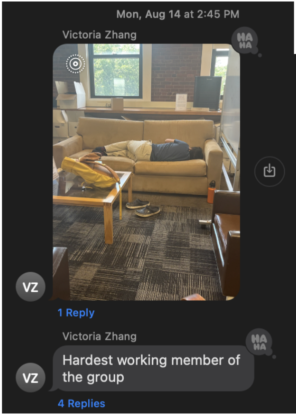
- Written in Julia, using Yao package
- Utilized FLOYao package to harness Matchgate optimizations
- Current progress

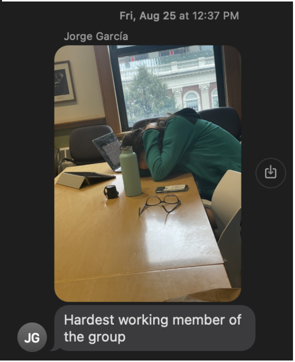
Acknowledgements

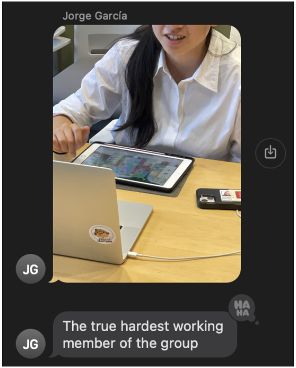
- Hong-Ye Hu and Susanne Yelin



- Yelin group undergrads: Jorge, Fiona, Victoria, and Kevin







Questions?