Comparison of Hybrid and Quantum Neural Networks

By Neema Badihian and Saaketh Rayaprolu

History of Neural Networks

- 1943 - Neurologist Warren McCulloch & Mathematician Walter Pitts create electric circuit to model neurons
- 1949 - Donald Hebb suggests neural pathways get stronger the more they are used
- 1958 - Frank Rosenblatt creates single perceptron

History of Neural Networks

- 1959 - Bernard Widrow & Marcian Hoff create Multiple Adaptive Linear Elements (MADALINE) to eliminate noise in phone lines
- 1969 - Marvin Minsky & Seymour Papert criticize Rosenblatt's perceptron and identify issues in neural networks
- 1982 - Jon Hopfield creates Hopfield Net which uses recurrent neural network

History of Neural Networks

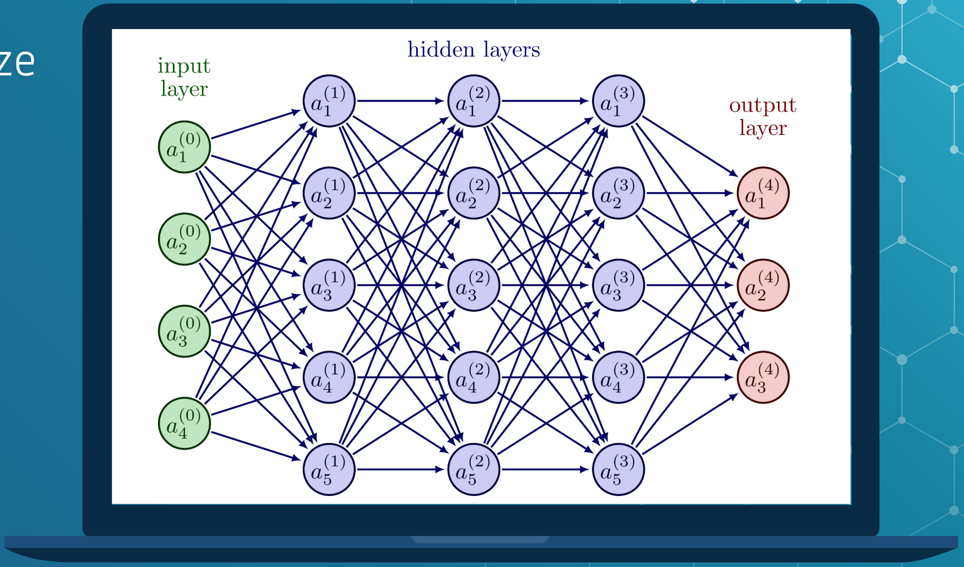
- 1985 - D.B. Parker publishes report on Paul Werbos' 1974 dissertation of backpropagation
- Now - Neural Networks are stronger than ever!

Neural Network Uses

- Handwriting Recognition
- Facial Recognition
- Financial Analysis
- Weather Prediction
- Medical Diagnostics

Classical Neural Network Layers

- Input Layer
  - Takes in data to analyze
- Hidden Layer
  - Manipulates data
- Output Layer
  - Returns results



Classical Neural Network Logic

Forward Propagation

- Input external data
- Neurons are multiplied by weight, adjusted by the bias, and summed
  - $z_m^{(1)} = \Sigma_1^n a_n^{(0)} w_{n,m}^{(0)} + b_n^{(0)}$
- Result is passed to an activation function
  - $a_3^{(1)} = g(z_3^{(1)})$
  - Makes neural network non-linear
- Repeat process until the output layer

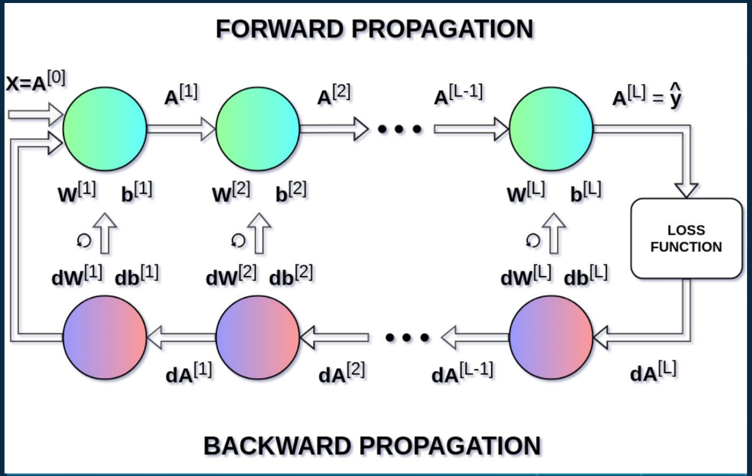
Classical Neural Network Logic

Backward Propagation

- Loss function determines output similarity to expected results
- Gradient descent method minimizes loss function
- New weights and biases are calculated and passed to appropriate neurons

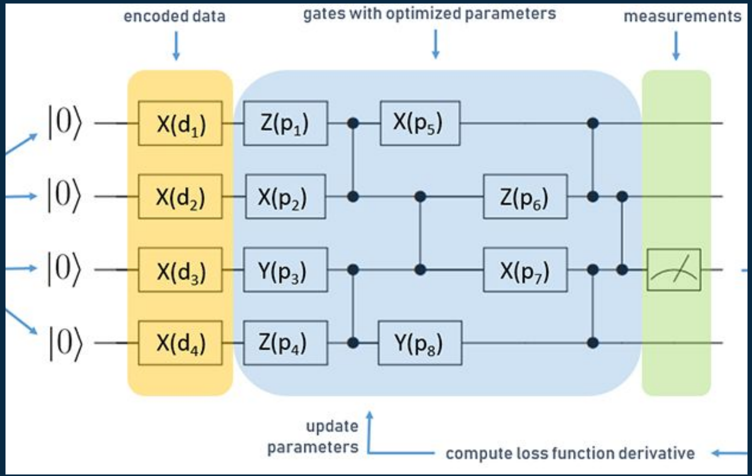
Classical Neural Network Logic

- Forward and backward propagation are repeated as necessary



Quantum Neural Networks

- Expands uses of neural networks to quantum chemistry, higher complexity optimization problems, etc
- Works best on quantum machines, very complex operations for classical machines to handle



Quantum Neural Network - Training

- Given input, L hidden layers, and output layer
- Initialize input qubits with unitaries encoding values of the data (can be arbitrary)
- $\rho^{out} = \mathrm{tr}_{in,\ hid}(U^{tot}(\rho^{in} \otimes |0\ldots0\rangle_{in,hid}\langle 0\ldots0|)U^{tot\dagger})$
  - U^{tot} is the product of all layer unitaries
  - $U^{l} = U^{l}_{N}U^{l}_{N-1}\ldots U^{l}_{2}U^{l}_{1}$: unitaries between layers *l-1* and *l* (akin to weights): updates iteratively
- Can be redefined as: $\rho^{out} = \varepsilon^{out}(\varepsilon^{L}(\ldots\varepsilon^{1}(\rho^{in})\ldots))$

Quantum Neural Network - Feed Forward

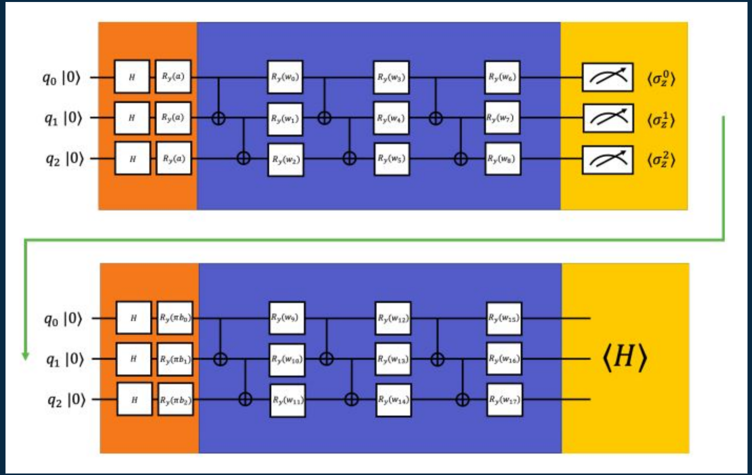
- All $U_x^{\,l}$ are initialized as ansatze
- Channel $\varepsilon^l(\rho^{l-1}_x) = U^l(\rho^{l-1}_x \otimes |0...0\rangle_l\langle 0...0|)U^{l\dagger}$
- For each step l, apply channel ε^l and then trace out the l-1 system
- Parameter matrix entry $K^l_j=(\eta/X)2^{m_{l-1}}\sum_x tr_{rest}(M^l_j)$
  - $M^l_j = [\varepsilon^l(\rho^{l-1}_x),\varepsilon^l(id_{l-1}\otimes\sigma^l_x)]$
- In essence, the feed-forward process acts on multiple levels of multilayer operations

Quantum Neural Network - Cost

- Maximize $C=(1/N)\sum_N \langle \varphi^{out}_n | \rho^{out}_n | \varphi^{out}_n \rangle$
- $\Delta C=(\epsilon/N)\sum_x \sum_{L+1} tr(\sigma^l_x \Delta\varepsilon^l(\rho^{l-1}_x))$
  - Range: [0,1]
- Each step updates $U \rightarrow e^{i\epsilon K}U$
  - K is a matrix made of all K^l_j defined earlier
- At any point, only two layers of memory are needed

Hybrid Neural Networks

- Accuracy of quantum algorithms + classical cost functions for speed
- Hadamard and Ry gates to encode data
- PQC is used to manipulate data stored in perceptrons
- Classical activation relies on expected values



Hybrid Neural Network - Encoding

- $|\psi_{encoded}\rangle = \otimes R_Y(a_i) \cdot H|0\rangle^{\otimes n}$
- Hadamard all qubits to place system in uniform superposition
- R_Y gates with angle θ representing the data being encoded into each neuron
    - R_Y is used because rotation occurs on X-Z plane (all real values)
    - Operational overhead decreases when not using complex numbers

Hybrid Neural Network - Weights

- Repeating m times for m qubits in the next layer:
  - Entangle all qubits with CNOT
    - Common methods: Full, Linear, Circular
  - Apply R_Y gates with angle ω representing weight of neuron
  - ω is an ansatz that iteratively gets refined
- Parametrized Quantum Circuits (PQC) are utilized due to high operational efficiency

Hybrid Neural Network - Activation

- Common activation functions:
  - Measurement of qubits under Z basis: $\langle \boldsymbol{\sigma}_z^{\,i} \rangle$
    - Expectation is used because quantum tomography is inefficient to implement
  - Fidelity with a fixed state: $|\langle \psi_{fixed} | \psi_i \rangle|^2$
- Use classical channels to initialize measurement data as θ for next layer of qubits (feedback loop)
  - First multiply output value by π to normalize new θ from [-1 , 1] to [-π , π]

Next Steps

- Complexity comparisons of Classical, Quantum, and Hybrid NN's
  - Why QNNs are the most complex to run
  - Whether benefits of HNN actually outweigh CNN and QNN (likely circumstantial)
- Runtime analyses of NNs and implementation of HNN

THANKS!

ANY QUESTIONS?

QREDITS

- https://www.nature.com/articles/s41467-020-14454-2.pdf
- https://arxiv.org/pdf/1912.06184.pdf
- https://arxiv.org/pdf/2108.01468.pdf
- https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec
- https://kasperfred.com/series/introduction-to-neural-networks/computational-complexity-of-neural-networks