Quantum Neural Networks

w/ Simulated Results for a $2 \times 3 \times 2$ Network

Alex Heilman

February 14, 2023

Overview

Classical Neural Networks have shown to be effective in a wide variety of uses.

Overview

Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

Overview

Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

Overview

$$\{|\phi_i^{in}\rangle, |\phi_i^{out}\rangle\}$$

Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

Overview

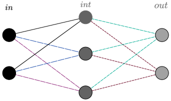
Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

$$\{|\phi_i^{in}\rangle, |\phi_i^{out}\rangle\}$$

$$\downarrow$$



Overview

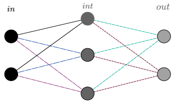
Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

$$\{|\phi_i^{in}\rangle, |\phi_i^{out}\rangle\}$$

$$\downarrow$$



$$\downarrow$$

$$\max_U \left(\sum_{i=1}^{N} \langle \psi_i^{out} | \rho_{out} | \psi_i^{out} \rangle \right)$$

Overview

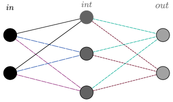
Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

$$\{|\phi_i^{in}\rangle, |\phi_i^{out}\rangle\}$$

$$\downarrow$$



$$\downarrow$$

$$\max_U \left(\sum_{i=1}^{N} \langle \psi_i^{out} | \rho_{out} | \psi_i^{out} \rangle \right)$$

$$\downarrow$$

$$U \rightarrow e^{-\epsilon K} U$$

Overview

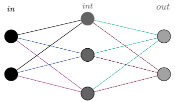
Classical Neural Networks have shown to be effective in a wide variety of uses.

With the advent of quantum computers, some modern research has been investigating quantum analogues of classical neural networks.

One proposed framework is that in Kerstin Beer, et al's *Training Deep Quantum Neural Networks*, which shall be considered here.

$$\{|\phi_i^{in}\rangle, |\phi_i^{out}\rangle\}$$

$$\downarrow$$



$$\downarrow$$

$$\max_U \left(\sum_{i=1}^{N} \langle \psi_i^{out} | \rho_{out} | \psi_i^{out} \rangle \right)$$

$$\downarrow$$

$$U \to e^{-\epsilon K} U$$

Motivation: Quantum Operations

Imagine we have some small system we're interested in, but it inevitably interacts with some larger system we'll term the environment.

Motivation: Quantum Operations

Imagine we have some small system we're interested in, but it inevitably interacts with some larger system we'll term the environment.

$$\rho \rightarrow \rho \otimes \rho_{Env.}$$

Motivation: Quantum Operations

Imagine we have some small system we're interested in, but it inevitably interacts with some larger system we'll term the environment.

$$\rho \rightarrow \rho \otimes \rho_{Env.}$$

It then evolves in concert with this larger system according to a unitary transformation:

$$\rho \otimes \rho_{Env.} \rightarrow U(\rho \otimes \rho_{Env.})U^{\dagger}$$

Motivation: Quantum Operations

Imagine we have some small system we're interested in, but it inevitably interacts with some larger system we'll term the environment.

$$\rho \to \rho \otimes \rho_{Env.}$$

It then evolves in concert with this larger system according to a unitary transformation:

$$\rho \otimes \rho_{Env.} \to U(\rho \otimes \rho_{Env.})U^{\dagger}$$

However, we still only care about and measure the subsystem, hence we end up with a reduced density matrix:

$$U(\rho \otimes \rho_{Env.})U^{\dagger} \to \mathrm{Tr}_{Env.}\left[U(\rho \otimes \rho_{Env.})U^{\dagger}\right]$$

Training Data Structure

Data will be provided for the training of the network via a set of arbitrary states (inputs), and the set of these same states after having some common unitary action act upon them (outputs). Hence, we will assume some given data set of the following form:

$$\text{Training Data:} \quad \{(|\psi_i\rangle, V|\psi_i\rangle) \mid 1 \leq i \leq N\}$$

Training Data Structure

Data will be provided for the training of the network via a set of arbitrary states (inputs), and the set of these same states after having some common unitary action act upon them (outputs). Hence, we will assume some given data set of the following form:

$$\text{Training Data:} \quad \{(|\psi_i\rangle, V|\psi_i\rangle) \mid 1 \leq i \leq N\}$$

This is a reasonable set of data since the most general quantum network will apply an arbitrary unitary gate, and hence the most general circuit should be able to approximate such actions.

Architecture

The overall action of the network is composed of layer-by-layer composition of the transition map ϵ^ℓ for each layer ℓ s.t. $in \leq \ell \leq out$.

Architecture

The overall action of the network is composed of layer-by-layer composition of the transition map ϵ^{ℓ} for each layer ℓ s.t. $in \leq \ell \leq out$.

Each layer may have a different number of qubits M_ℓ. Explicitly, the ℓ-th layer's transition map takes the form:

$$\epsilon^{\ell}(\rho_{\ell-1}) =$$

$$\mathrm{Tr}_{\ell-1}\left[\left(\prod_{m=1}^{M_\ell} U_\ell^{m-M_\ell}\right)\left(\left(|0\rangle^{\otimes M_\ell}\langle 0|^{\otimes M_\ell}\right)_\ell \otimes \rho_{\ell-1}\right)\left(\prod_{m=1}^{M_\ell} U_\ell^{m\dagger}\right)\right]$$

$$= \rho_\ell$$

Architecture

The overall action of the network is composed of layer-by-layer composition of the transition map ϵ^ℓ for each layer ℓ s.t. $in \leq \ell \leq out$.

Each layer may have a different number of qubits M_ℓ. Explicitly, the ℓ-th layer's transition map takes the form:

$$\epsilon^\ell \left(\rho_{\ell-1}\right) =$$

$$\mathrm{Tr}_{\ell-1}\left[\left(\prod_{m=1}^{M_\ell} U_\ell^{m-M_\ell}\right)\left(\left(|0\rangle^{\otimes M_\ell}\langle 0|^{\otimes M_\ell}\right)_\ell \otimes \rho_{\ell-1}\right)\left(\prod_{m=1}^{M_\ell} U_\ell^{m\dagger}\right)\right]$$

$$= \rho_\ell$$

And, hence, a total circuit of L layers returns ρ_{out}, defined below, for some given input state ρ_{in}.

$$\rho_{out} = \epsilon^{out}\left(\epsilon^L\left(\epsilon^{L-1}\left(\ldots\epsilon^1\left(\rho_{in}\right)\ldots\right)\right)\right)$$

Architecture: Step-by-step

For each layer ℓ,

Architecture: Step-by-step

For each layer ℓ,

1. The next layer's M qubits are prepared in the initial state $|0\rangle^{\otimes m}\langle 0|_{\ell}^{\otimes m}$ and tensor producted with the previous layer's output $\rho_{\ell-1}$.

$$\rho'_{\ell} = \left(|0\rangle^{\otimes M}\langle 0|^{\otimes M}\right)_{\ell} \otimes \rho_{\ell-1}$$

Architecture: Step-by-step

For each layer ℓ,

1. The next layer's M qubits are prepared in the initial state $|0\rangle^{\otimes m}\langle 0|_{\ell}^{\otimes m}$ and tensor producted with the previous layer's output $\rho_{\ell-1}$.

$$\rho'_{\ell} = \left(|0\rangle^{\otimes M}\langle 0|^{\otimes M}\right)_{\ell} \otimes \rho_{\ell-1}$$

2. The ℓ-th layer's M associated unitary matrices U_{ℓ}^{m} are applied to this tensor product state (from top to bottom).

$$\rho''_{\ell} = \left(\prod_{m=0}^{M-1} U_{\ell}^{M-m}\right)(\rho'_{\ell})\left(\prod_{m=1}^{M} U_{\ell}^{m\dagger}\right)$$

Architecture: Step-by-step

For each layer ℓ,

1. The next layer's M qubits are prepared in the initial state $|0\rangle^{\otimes m}\langle 0|_{\ell}^{\otimes m}$ and tensor producted with the previous layer's output $\rho_{\ell-1}$.

$$\rho'_{\ell} = \left(|0\rangle^{\otimes M}\langle 0|^{\otimes M}\right)_{\ell} \otimes \rho_{\ell-1}$$

2. The ℓ-th layer's M associated unitary matrices U_{ℓ}^{m} are applied to this tensor product state (from top to bottom).

$$\rho''_{\ell} = \left(\prod_{m=0}^{M-1} U_{\ell}^{M-m}\right)(\rho'_{\ell})\left(\prod_{m=1}^{M} U_{\ell}^{m\dagger}\right)$$

3. The partial trace over the $(\ell-1)$th layer's Hilbert space is taken, resulting in the output state ρ_{ℓ} of the ℓ-th layer.

$$\rho_{\ell} = \mathrm{Tr}_{\ell-1}[\rho''_{\ell}]$$

Cost

The metric by which we will judge the performance of the network on the training data is the cost, here taken as the average fidelity between the networks output state and the corresponding state given in training and explicity defined as:

$$C = \frac{1}{N} \sum_{i=1}^{N} \langle \psi_i^{out} | \rho_{out} | \psi_i^{out} \rangle$$

Note that this cost function is only applicable for training data based on pure states, for which the fidelity takes an especially nice form.

Cost

The metric by which we will judge the performance of the network on the training data is the cost, here taken as the average fidelity between the networks output state and the corresponding state given in training and explicity defined as:

$$C = \frac{1}{N}\sum_{i=1}^{N}\langle\psi_i^{out}|\rho_{out}|\psi_i^{out}\rangle$$

Note that this cost function is only applicable for training data based on pure states, for which the fidelity takes an especially nice form.

For input mixed states, we may replace the above with an averaged fidelity between output and target states of the form:

$$C = \frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{Tr}\left[\sqrt{\sqrt{\rho_i}\rho_i^{out}\sqrt{\rho_i}}\right]\right)^2$$

Training

We now wish to maximize the previously defined cost function (which has a maximum value of 1). This may be accomplished through training.

Training

We now wish to maximize the previously defined cost function (which has a maximum value of 1). This may be accomplished through training.

Training may be performed by evolving each unitary via the following map:

$$U_m^\ell \rightarrow e^{-\varepsilon K_m^\ell} U_m^\ell$$

which is parameterized by the step size ε, and where K_m^ℓ is derived from the derivative of the cost function and takes the following form:

$$K_m^\ell = \eta \frac{2^{m_\ell - 1}}{N} \sum_{i=1}^{N} \mathrm{Tr}_{\neg \ell} \Bigg[\left(\prod_{n=0}^{m-1} U_{m-n}^\ell \right) \left(\left(|0\rangle^{\otimes m_\ell} \langle 0|^{\otimes m_\ell} \right)_\ell \otimes \rho_i^{\ell-1} \right) \left(\prod_{n=1}^{m} U_m^{\ell\dagger} \right), \left(\prod_{n=m+1}^{m_\ell} U_n^{\ell\dagger} \right) \left(\sigma_i^\ell \otimes \mathbb{I}_{\ell-1} \right) \left(\prod_{n=1}^{m_\ell - (m+1)} U_{m_\ell - n}^\ell \right) \Bigg]$$

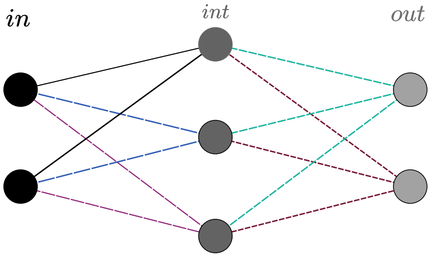
where the square brackets denote a commutator and $\sigma_i^\ell = \mathcal{F}^{\ell+1}(\ldots \mathcal{F}^{out}(\rho_i^{out})\ldots)$ is the adjoint channel to the layer-to-layer transition map ϵ^ℓ for layer ℓ.

Simple Example: $2 \times 3 \times 2$ QNN

As a simple example, we consider a QNN with one hidden layer of three qubits, and a two qubit input and output.

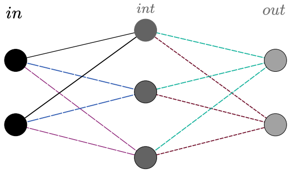
Simple Example: $2 \times 3 \times 2$ QNN

As a simple example, we consider a QNN with one hidden layer of three qubits, and a two qubit input and output.



$$\mathrm{Tr}_{int}\Big[U_2^{out}U_1^{out}\Big(\mathrm{Tr}_{in}\Big[U_3^{int}U_2^{int}U_1^{int}(\rho_{in}\otimes|000\rangle\langle 000|_{int})U_1^{int\dagger}U_2^{int\dagger}U_3^{int\dagger}\Big]\Big)U_1^{out\dagger}U_2^{out\dagger}\Big] = \rho_{out}$$

$2 \times 3 \times 2$ Example



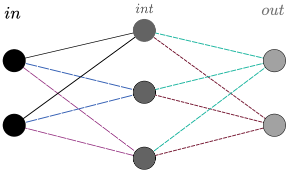
$$\mathrm{Tr}_{int}\Big[U_2^{out}U_1^{out}\Big(\mathrm{Tr}_{in}\Big[U_3^{int}U_2^{int}U_1^{int}(\rho_{in}\otimes|000\rangle\langle 000|_{int})U_1^{int\dagger}U_2^{int\dagger}U_3^{int\dagger}\Big]\Big)U_1^{out\dagger}U_2^{out\dagger}\Big] = \rho_{out}$$

In this case we have $3_{(int)} + 2_{(out)} = 5$ constituent unitaries composing the QNN:

$U_1^1,\ U_2^1,\ U_3^1$ for intermediate layer;

$U_1^{out},\ U_2^{out}$ for final layer.

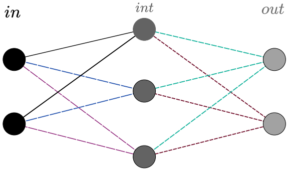
$2 \times 3 \times 2$ Example



$$\mathrm{Tr}_{int}\Big[U_2^{out}U_1^{out}\Big(\mathrm{Tr}_{in}\Big[U_3^{int}U_2^{int}U_1^{int}(\rho_{in}\otimes|000\rangle\langle000|_{int})U_1^{int\dagger}U_2^{int\dagger}U_3^{int\dagger}\Big]\Big)U_1^{out\dagger}U_2^{out\dagger}\Big] = \rho_{out}$$

Unitaries for the intermediate layer U_m^1 then act non-trivially on a state space of dimension $2^{2+1} \times 2^{2+1}$ but are tensored with identity in the rest, resulting in a matrix of dimension $2^{2+3} \times 2^{2+3}$.

$2 \times 3 \times 2$ Example



$$\mathrm{Tr}_{int}\Big[U_2^{out}U_1^{out}\Big(\mathrm{Tr}_{in}\Big[U_3^{int}U_2^{int}U_1^{int}(\rho_{in}\otimes|000\rangle\langle000|_{int})U_1^{int\dagger}U_2^{int\dagger}U_3^{int\dagger}\Big]\Big)U_1^{out\dagger}U_2^{out\dagger}\Big] = \rho_{out}$$

Unitaries for the intermediate layer U_m^1 then act non-trivially on a state space of dimension $2^{2+1} \times 2^{2+1}$ but are tensored with identity in the rest, resulting in a matrix of dimension $2^{2+3} \times 2^{2+3}$.

Similarly, unitaries for the output layer U_m^1 then act on a state space of dimension $2^{3+1} \times 2^{3+1}$ but are tensored with identity in the rest, resulting in a matrix of dimension $2^{3+2} \times 2^{3+2}$.

$2 \times 3 \times 2$ Example: Training

These 5 unitaries then require us to construct 5 training matrices K^ℓ_m, each corresponding uniquely to one of the above unitaries.

As an example, consider the following training matrices for the intermediate layer's unitaries:

$$K^1_1 = \eta \frac{2^2}{N} \sum_{i=1}^{N} \mathrm{Tr}_{2,3_{int}} \left[U^1_1 \left(\rho^{in}_i \otimes |000\rangle \langle 000|_1 \right) U^{1\dagger}_1 , U^{1\dagger}_2 U^{1\dagger}_3 \left(\mathbb{I}_{2^2} \otimes \sigma^1_i \right) U^1_3 U^1_2 \right]$$

$$K^1_2 = \eta \frac{2^2}{N} \sum_{i=1}^{N} \mathrm{Tr}_{1,3_{int}} \left[U^1_2 U^1_1 \left(\rho^{in}_i \otimes |000\rangle \langle 000|_1 \right) U^{1\dagger}_1 U^{1\dagger}_2 , U^{1\dagger}_3 \left(\mathbb{I}_{2^2} \otimes \sigma^1_i \right) U^1_3 \right]$$

Software Choice

Since most quantum computational SDKs don't explicitly allow for partial trace operations in circuits, we utilize QuTip, a QIS toolkit coded in python.

Software Choice

Since most quantum computational SDKs don't explicitly allow for partial trace operations in circuits, we utilize QuTip, a QIS toolkit coded in python.

This facilitates the manipulation of tensor product structure, and the taking of partial traces.

Rough Structure

The code essentially does the following:

1. Generate Data

Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training

Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training
   (a) Forward Pass

Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training
   (a) Forward Pass
   (b) Calculate Training Matrices

Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training
   (a) Forward Pass
   (b) Calculate Training Matrices
   (c) Update Constituent Unitaries

Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training
   (a) Forward Pass
   (b) Calculate Training Matrices
   (c) Update Constituent Unitaries
   (d) Calculate and Store Costs

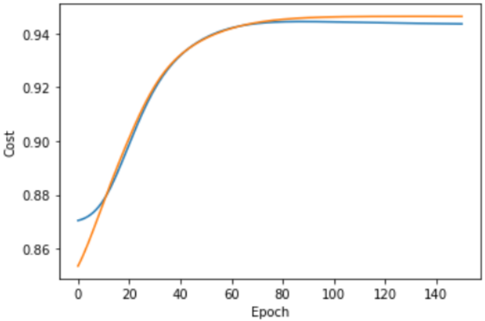
Rough Structure

The code essentially does the following:

1. Generate Data
2. Loop Through Training
   (a) Forward Pass
   (b) Calculate Training Matrices
   (c) Update Constituent Unitaries
   (d) Calculate and Store Costs
3. Plot Test Data Cost vs. Training Epoch

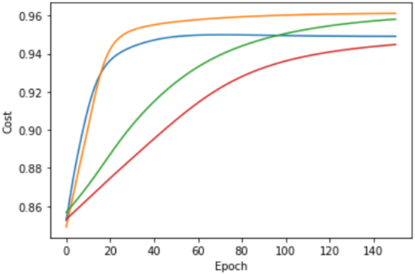
Simulation Results: 2x3x2 QNN I

Below is the example $2 \times 3 \times 2$ QNN's average fidelity on the test data after being trained on clean data (based on the same underlying unitary transformation).



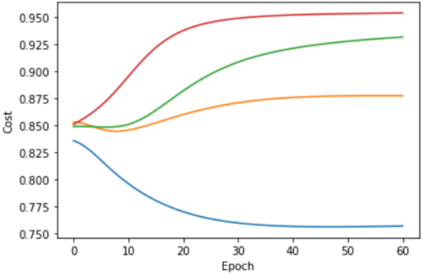
The fidelity clearly increased over time towards some asymptotic limit. Here, blue is the test data's cost whereas the orange represents training data cost.

Simulation Results: 2x3x2 QNN II



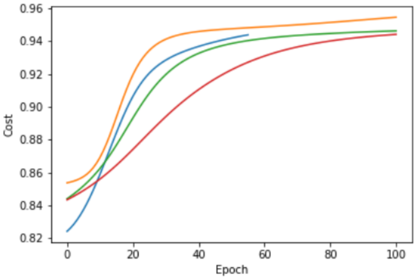
Above is plotted the cost vs. training epoch for the simulated network for 50 training pairs over 150 epochs with $\eta = 1$. Note that the orange represent the cost for a learning rate of $\epsilon = 1$, blue $\epsilon = 0.5$, green $\epsilon = 0.25$, and red $\epsilon = 0.1$; all the costs are for the same training data

Simulation Results: 2x3x2 QNN III



Cost vs. training epoch for the simulated network for a different number of a few training pairs over 60 epochs with $\epsilon = 0.5$, $\eta = 1$. Note that the orange represent the cost for 5 training pairs, blue 10 training pairs, green 25 training pairs, and red 50 training pairs; all the costs are for an independent set of 50 test pairs.

Simulation Results: 2x3x2 QNN IV



Cost vs. training epoch for the simulated network for 200 (with varied noisy) clean training pairs over 100 epochs with $\epsilon = 0.5$, $\eta = 1$. Orange represents 0 noisy pairs in the training, blue 50 noise, green 150 noise, and red 200 noise. The cost displayed is for an independent set of 30 test pairs corresponding to the clean data's operation. Note further that the training was stopped at a training data cost of .95 (since a perfect match wouldn't correspond exactly to the unknown unitary, given noise).

So what could we do?

Quantum Simulation of Material Systems? Need reliable channel for data (input/output), maybe not attainable short term

Generalize to graph states? Something that maintains connectivity/entanglement properties, could be interesting

Thanks!

Thanks for your time!

For more details, see https://alexheilman.com/res/qis/qml
and the original paper: https://www.nature.com/articles/s41467-020-14454-2

For a full report on the simulation see https://alexheilman.com/products/complete/qnn_232.pdf