Quantum Algorithms for Deep Convolutional Neural Networks

Alejandro Rosales

Research Paper Reference

Title: Quantum Algorithms for Deep Convolutional Neural Networks

Authors: Iordanis Kerenidis, Jonas Landman, and Anupam Prakash

Research Centers: CNRS, IRIF, Universite Paris Diderot, Paris, France

Dated: November 15, 2019

Link: https://arxiv.org/pdf/1911.01117.pdf

Motivation/Goal

Current problem: "...it is difficult to implement non linearities with quantum unitaries."

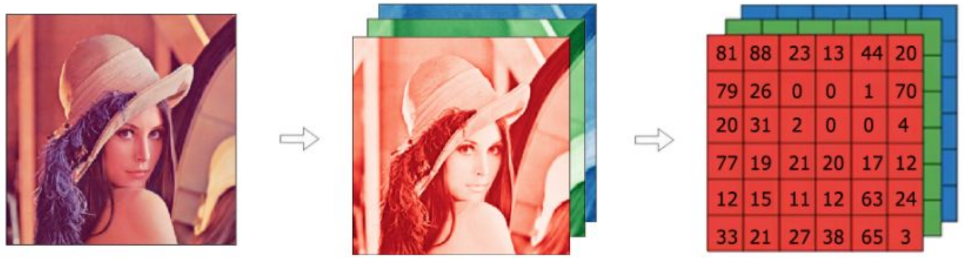
Suggested solution: "a new quantum tomography algorithm with norm guarantees, and new applications of probabilistic sampling in the context of information processing."

Goal: "The QCNN is particularly interesting for deep networks and could allow new frontiers in image recognition, by using more or larger convolution kernels, larger or deeper inputs"

Quick Presentation Overview

Preliminaries and Refreshers

Representation of a Classical CNN layers and procedures



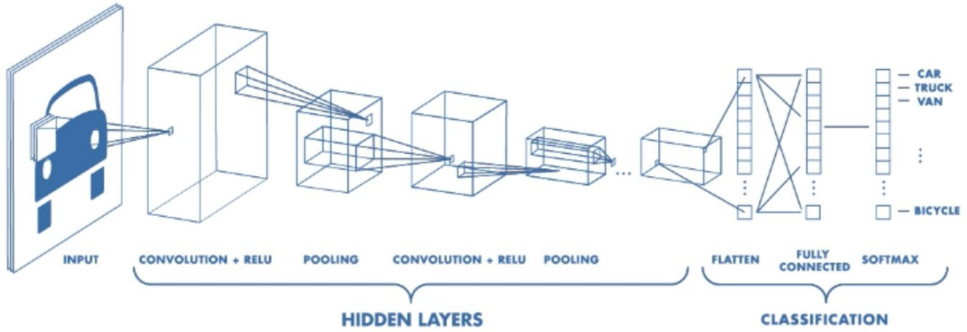
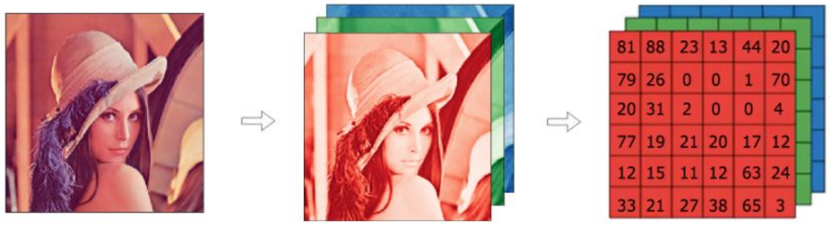
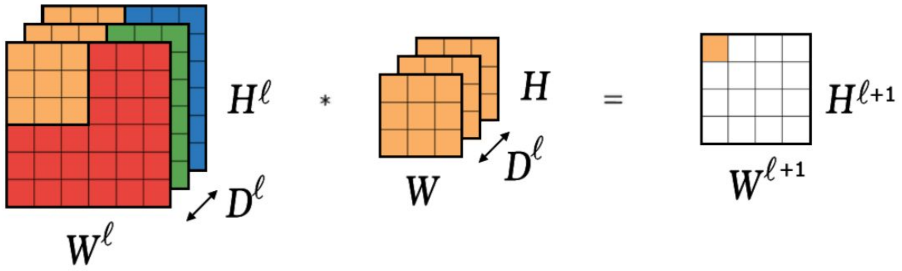


Figure 2: Representation of a CNN layers and procedures [1]

Representation of a Classical CNN layers and procedures (Cont.)





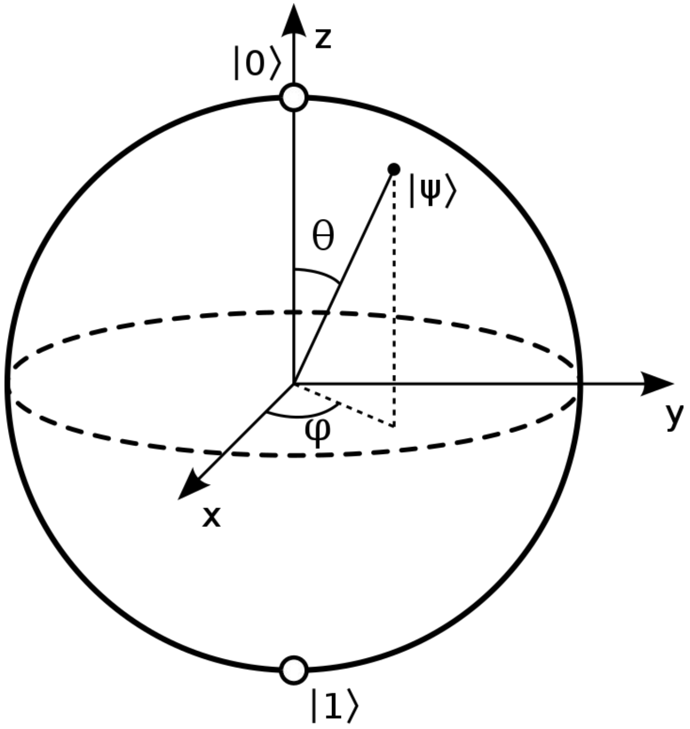
Probability Amplitudes and the Bloch Sphere

A qubit can be at any surface of the bloch sphere.

$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$,

where $|\alpha|^2 + |\beta|^2 = 1$

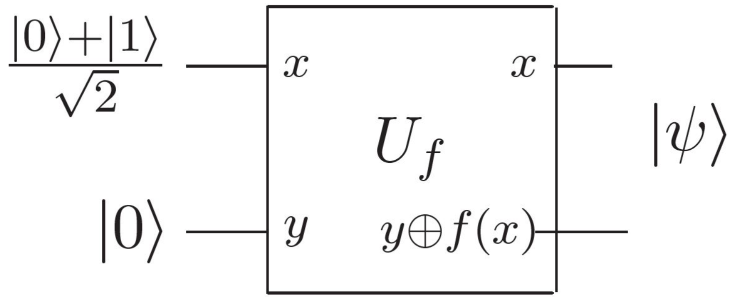
Where α and β are the probabilities of the system being in the representative states



Quantum Parallelism

Fundamental feature of many quantum algorithms

- Unlike classical parallelism, where multiple circuits each built to compute *f(x)* are executed simultaneously, here a single *f(x)* circuit is employed to evaluate the function for multiple values of *x* simultaneously,
- Done by quantum computers exploiting one qubit to be in superposition of different states at once



Quantum Convolution Layer

Definitions

Input and Kernel

input for the quantum convolution layer is a 3D tensor input $X^\ell \in \mathbb{R}^{H^\ell \times W^\ell \times D^\ell}$

weights layer, filter layer, or kernel layer is a 4D tensor $K^\ell \in \mathbb{R}^{H \times W \times D^\ell \times D^{\ell+1}}$

Precision and Quantum Tomography

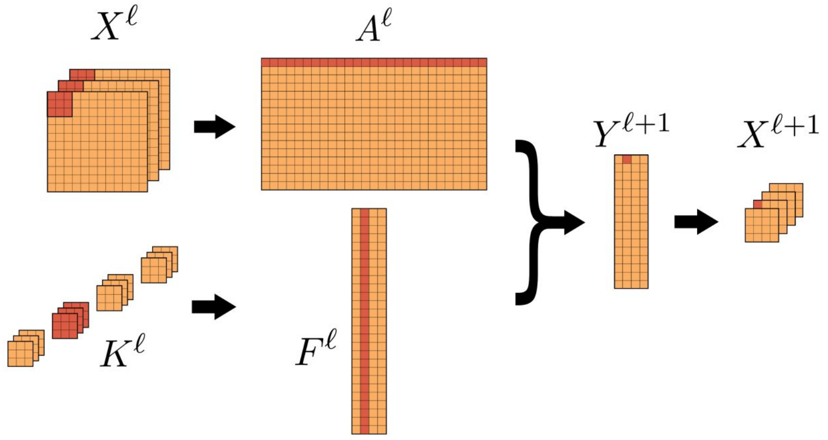
Given precision parameters $\epsilon, \Delta > 0$, there exists a quantum algorithm that computes a quantum state that is Δ close to $|f(\bar{X}^{\ell+1})\rangle$ where $X^{\ell+1} = X^\ell * K^\ell$,

Reformulate Convolution Product as Matrix Product

Given that a convolution product is equivalant to a matrix-matrix multiplication and the convolutional product between X^ℓ and K^ℓ is:

$$X^{\ell+1}_{i^{\ell+1},j^{\ell+1},d^{\ell+1}} = \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{d=0}^{D^\ell} K^\ell_{i,j,d,d^{\ell+1}} X^\ell_{i^{\ell+1}+i,j^{\ell+1}+j,d}$$

it is possible to reformulate this convolution product equation as a matrix product. A diagram of this reshaping of the input and kernel is given by the authors and depicited below:



How We Will Proceed

Quantum Convolution Layer Overview:

Quantum Convolution

Quantum Sampling

Quantum Random Access Memory and Pooling

OHIO UNIVERSITY

Inner Product Estimation

First we load input row vector A_p^ℓ and kernel vector F_q^ℓ into quantum states by quering QRAM in the following manner:

$$\begin{cases} |p\rangle|0\rangle \mapsto |p\rangle \left|A_p^\ell\right\rangle \\ |q\rangle|0\rangle \mapsto |q\rangle \left|F_q^\ell\right\rangle \end{cases}$$

This is done so that following mapping can be perfromed with the two vectors:

$$\frac{1}{K}\sum_{p,q}|p\rangle|q\rangle \mapsto \frac{1}{K}\sum_{p,q}|p\rangle|q\rangle|\bar{P}_{pq}\rangle\,|g_{pq}\rangle$$

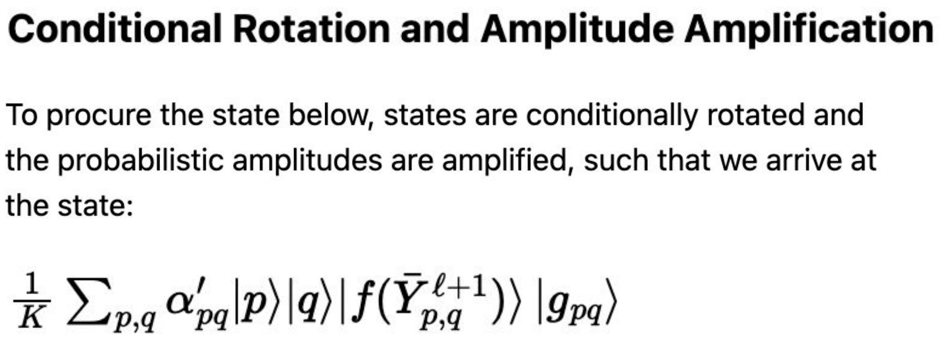
where

$$K = \sqrt{H^{\ell+1}W^{\ell+1}D^{\ell+1}}$$

“True” inner product is $P_{pq} = \dfrac{1 + \langle A_p^\ell \mid F_q^\ell \rangle}{2}$

FOREVER OHIO

Quantum Feature Mapping



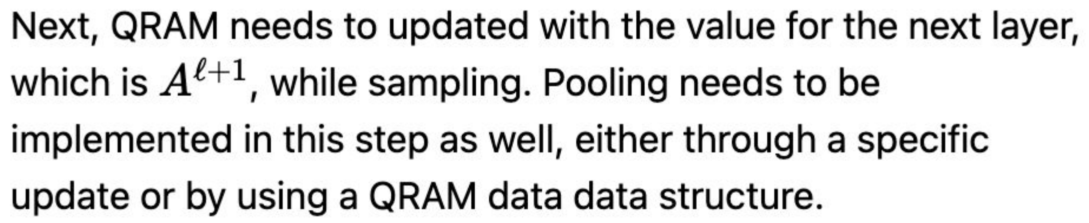
Conditional Rotation and Amplitude Amplification

To procure the state below, states are conditionally rotated and the probabilistic amplitudes are amplified, such that we arrive at the state:

$$\frac{1}{K}\sum_{p,q}\alpha'_{pq}|p\rangle|q\rangle|f(\bar{Y}^{\ell+1}_{p,q})\rangle\,|g_{pq}\rangle$$

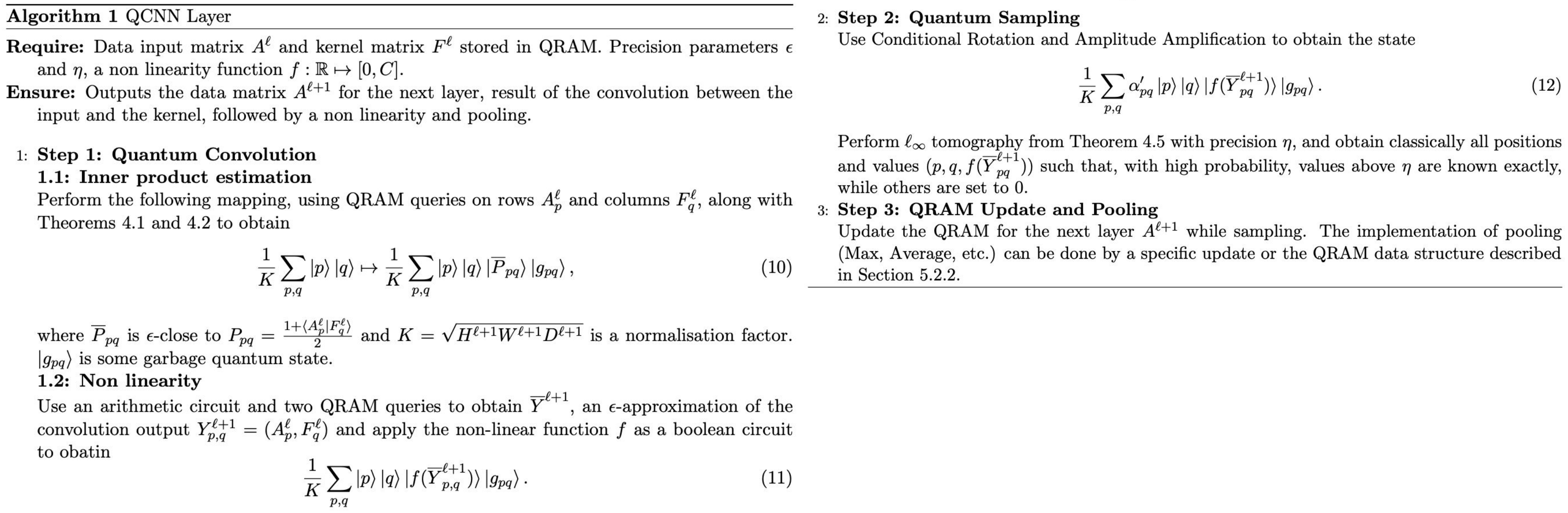
Quantum tomography is performed with precision η so that all values and positions $(p,q,f(\bar{Y}_{pq}^{\ell+1}))$ are obtained with a high probability.

QRAM Update and Pooling



Next, QRAM needs to updated with the value for the next layer, which is $A^{\ell+1}$, while sampling. Pooling needs to be implemented in this step as well, either through a specific update or by using a QRAM data data structure.

QC Algorithm Overview



Algorithm 1 QCNN Layer

```
Require: Data input matrix $A^\ell$ and kernel matrix $F^\ell$ stored in QRAM. Precision parameters $\epsilon$
    and $\eta$, a non linearity function $f : \mathbb{R} \mapsto [0, C]$.
Ensure: Outputs the data matrix $A^{\ell+1}$ for the next layer, result of the convolution between the
    input and the kernel, followed by a non linearity and pooling.

1: Step 1: Quantum Convolution
    1.1: Inner product estimation
    Perform the following mapping, using QRAM queries on rows $A_p^\ell$ and columns $F_q^\ell$, along with
    Theorems 4.1 and 4.2 to obtain
        $$\frac{1}{K}\sum_{p,q} |p\rangle |q\rangle \mapsto \frac{1}{K}\sum_{p,q} |p\rangle |q\rangle |\overline{P}_{pq}\rangle |g_{pq}\rangle ,  \quad (10)$$
    where $\overline{P}_{pq}$ is $\epsilon$-close to $P_{pq} = \frac{1+\langle A_p^\ell | F_q^\ell \rangle}{2}$ and $K = \sqrt{H^{\ell+1}W^{\ell+1}D^{\ell+1}}$ is a normalisation factor.
    $|g_{pq}\rangle$ is some garbage quantum state.
    1.2: Non linearity
    Use an arithmetic circuit and two QRAM queries to obtain $\overline{Y}^{\ell+1}$, an $\epsilon$-approximation of the
    convolution output $Y_{p,q}^{\ell+1} = (A_p^\ell, F_q^\ell)$ and apply the non-linear function $f$ as a boolean circuit
    to obatin
        $$\frac{1}{K}\sum_{p,q} |p\rangle |q\rangle |f(\overline{Y}_{p,q}^{\ell+1})\rangle |g_{pq}\rangle .  \quad (11)$$
2: Step 2: Quantum Sampling
    Use Conditional Rotation and Amplitude Amplification to obtain the state
        $$\frac{1}{K}\sum_{p,q} \alpha'_{pq} |p\rangle |q\rangle |f(\overline{Y}_{pq}^{\ell+1})\rangle |g_{pq}\rangle .  \quad (12)$$
    Perform $\ell_\infty$ tomography from Theorem 4.5 with precision $\eta$, and obtain classically all positions
    and values $(p, q, f(\overline{Y}_{pq}^{\ell+1}))$ such that, with high probability, values above $\eta$ are known exactly,
    while others are set to 0.
3: Step 3: QRAM Update and Pooling
    Update the QRAM for the next layer $A^{\ell+1}$ while sampling. The implementation of pooling
    (Max, Average, etc.) can be done by a specific update or the QRAM data structure described
    in Section 5.2.2.
```

Amplitude Amplification

Where the notation $\mathbb{E}_{p,q}(f(\overline{Y}_{pq}))$ represents the average value of the matrix $f(\overline{Y})$. It can also be written $\mathbb{E}(f(\overline{X}))$ as in Result 1:

$$\mathbb{E}_{p,q}(f(\overline{Y}_{pq})) = \frac{1}{HWD} \sum_{p,q} f(\overline{Y}_{pq}) \tag{32}$$

Amplitude Amplification (Cont.)

At the end of these iterations, we have modified the state to the following:

$$|f(\overline{Y})\rangle = \frac{1}{\sqrt{HWD}} \sum_{p,q} \alpha'_{pq} \, |p\rangle \, |q\rangle \, |f(\overline{Y}_{pq})\rangle$$,where $\alpha'_{pq} = \frac{\alpha_{pq}}{\sqrt{\sum_{p,q} \frac{\alpha_{pq}^2}{HWD}}}$

Result of One Forward Pass

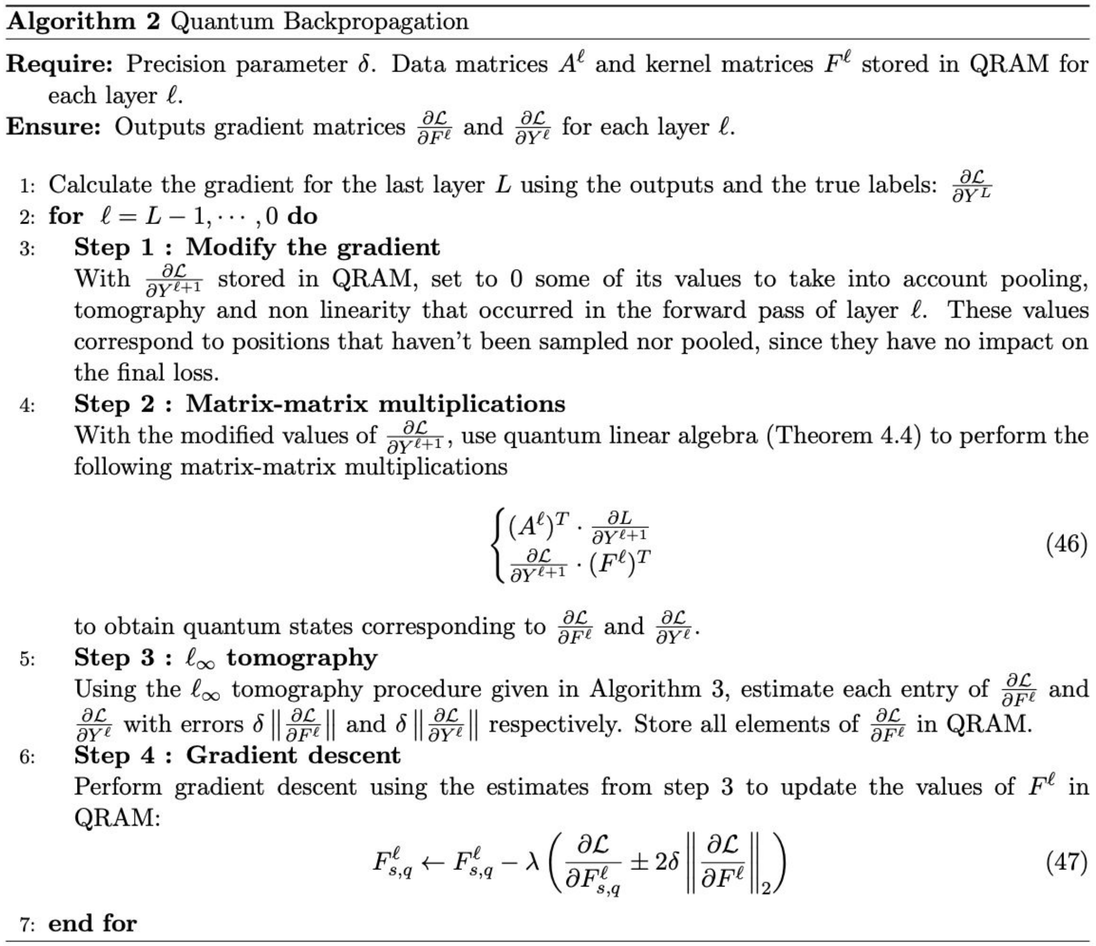
We can rewrite the final quantum state obtained in (33) as

$$|f(\overline{Y}^{\ell+1})\rangle = \frac{1}{\sqrt{\sum_{p,q} f(\overline{Y}^{\ell+1}_{pq})}} \sum_{p,q} \sqrt{f(\overline{Y}^{\ell+1}_{pq})}\,|p\rangle\,|q\rangle\,|f(\overline{Y}^{\ell+1}_{pq})\rangle \tag{35}$$

We see here that $f(\overline{Y}^{\ell+1}_{pq})$, the values of each pixel, are encoded in both the last register and in the amplitude. We will use this property to extract efficiently the exact values of high magnitude

More QCNN Fun:

Quantum CNN Backpropagation!



Algorithm 2 Quantum Backpropagation

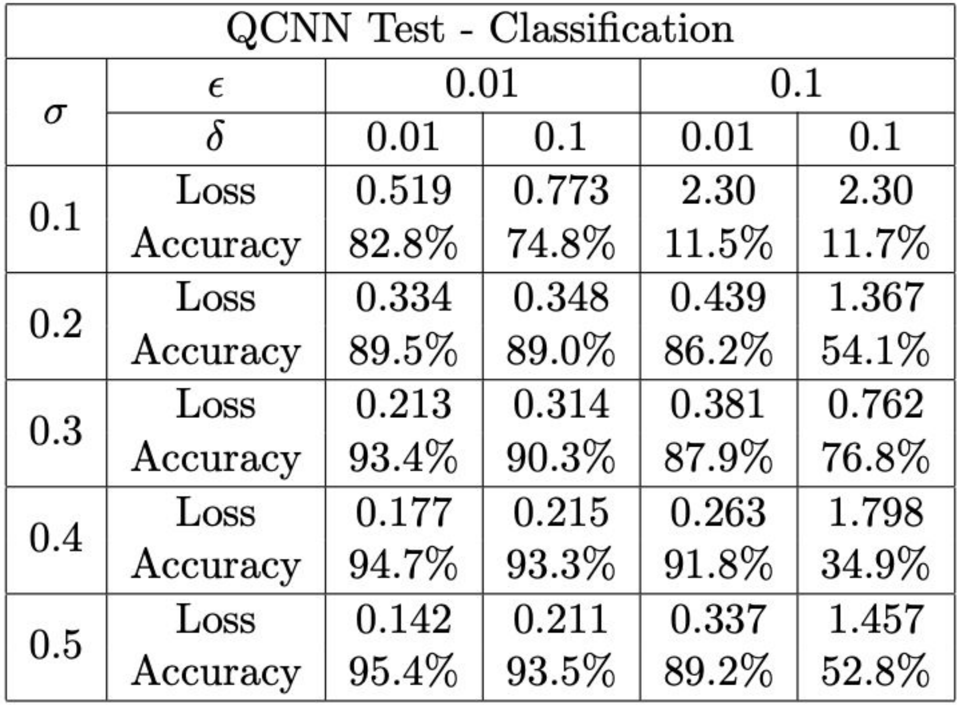
Require: Precision parameter δ. Data matrices A^ℓ and kernel matrices F^ℓ stored in QRAM for each layer ℓ.

Ensure: Outputs gradient matrices $\frac{\partial \mathcal{L}}{\partial F^\ell}$ and $\frac{\partial \mathcal{L}}{\partial Y^\ell}$ for each layer ℓ.

```
1: Calculate the gradient for the last layer $L$ using the outputs and the true labels: $\frac{\partial \mathcal{L}}{\partial Y^L}$
2: for $\ell = L-1, \cdots, 0$ do
3:     Step 1 : Modify the gradient
       With $\frac{\partial \mathcal{L}}{\partial Y^{\ell+1}}$ stored in QRAM, set to 0 some of its values to take into account pooling, tomography and non linearity that occurred in the forward pass of layer $\ell$. These values correspond to positions that haven't been sampled nor pooled, since they have no impact on the final loss.
4:     Step 2 : Matrix-matrix multiplications
       With the modified values of $\frac{\partial \mathcal{L}}{\partial Y^{\ell+1}}$, use quantum linear algebra (Theorem 4.4) to perform the following matrix-matrix multiplications
           $$\begin{cases} (A^\ell)^T \cdot \frac{\partial \mathcal{L}}{\partial Y^{\ell+1}} \\ \frac{\partial \mathcal{L}}{\partial Y^{\ell+1}} \cdot (F^\ell)^T \end{cases} \qquad (46)$$
       to obtain quantum states corresponding to $\frac{\partial \mathcal{L}}{\partial F^\ell}$ and $\frac{\partial \mathcal{L}}{\partial Y^\ell}$.
5:     Step 3 : $\ell_\infty$ tomography
       Using the $\ell_\infty$ tomography procedure given in Algorithm 3, estimate each entry of $\frac{\partial \mathcal{L}}{\partial F^\ell}$ and $\frac{\partial \mathcal{L}}{\partial Y^\ell}$ with errors $\delta \left\| \frac{\partial \mathcal{L}}{\partial F^\ell} \right\|$ and $\delta \left\| \frac{\partial \mathcal{L}}{\partial Y^\ell} \right\|$ respectively. Store all elements of $\frac{\partial \mathcal{L}}{\partial F^\ell}$ in QRAM.
6:     Step 4 : Gradient descent
       Perform gradient descent using the estimates from step 3 to update the values of $F^\ell$ in QRAM:
           $$F^\ell_{s,q} \leftarrow F^\ell_{s,q} - \lambda \left( \frac{\partial \mathcal{L}}{\partial F^\ell_{s,q}} \pm 2\delta \left\| \frac{\partial \mathcal{L}}{\partial F^\ell} \right\|_2 \right) \qquad (47)$$
7: end for
```

OHIO UNIVERSITY

Results



QCNN Test - Classification

σ	ϵ	0.01		0.1	
	δ	0.01	0.1	0.01	0.1
0.1	Loss	0.519	0.773	2.30	2.30
	Accuracy	82.8%	74.8%	11.5%	11.7%
0.2	Loss	0.334	0.348	0.439	1.367
	Accuracy	89.5%	89.0%	86.2%	54.1%
0.3	Loss	0.213	0.314	0.381	0.762
	Accuracy	93.4%	90.3%	87.9%	76.8%
0.4	Loss	0.177	0.215	0.263	1.798
	Accuracy	94.7%	93.3%	91.8%	34.9%
0.5	Loss	0.142	0.211	0.337	1.457
	Accuracy	95.4%	93.5%	89.2%	52.8%

: QCNN trained with quantum backpropagation on MNIST dataset. With $C = 10$ fixed.

FOREVER OHIO

Takeaways

Pros

“Our Quantum CNN is complete in the sense that almost all classical architectures can be implemented in a quantum fashion: any (non negative an upper bounded) non linearity, pooling, number of layers and size of kernels are available”

“The running time presents a speedup compared to the classical algorithm, due to fast linear algebra when computing the convolution product, and by only sampling the important values from the resulting quantum state. This speedup can be highly significant in cases where the number of channels D in the input tensor is high… allowing deep architectures for CNN, which was the case in the recent breakthrough of DeepMind AlphaGo algorithm.”

Cons

“...despite our new techniques to reduce the complexity, applying a non linearity and reusing the result of a layer for the next layer make register encoding and state tomography mandatory, hence preventing from having an exponential speedup on the number of input parameters.

Extra Resources:

https://contextswitching.org/tcs/quantumcnn.html

Thank you for coming to my TED talk

No questions!!

Are there any questions?

End scene.