CS230

Effectiveness of Quantum Neural Networks Utilizing Quantum Dropout

Ricky Young
Department of Engineering Physics
Stanford University
ryoung98@stanford.edu

Maxim Mints
Department of Computer Science
Stanford University
mmints@stanford.edu

Hiroki Nakayama
Department of Applied Physics
Stanford University
nahi5058@stanford.edu

Abstract

Fervent research has been accomplished in recent years to see if quantum computing can help with enhancing the performance in machine learning. While the current NISQ (noisy intermediate scale quantum) computers have limited access to coherent qubits and quantum gates, there is growing consense of their potential in reducing the parameter space and outperforming classical neural networks. In this paper, we demonstrate a quantum ready dropout regularization on Tensorflow Quantum using variational circuits. We also suggest alternative techniques similar to dropout regularization that is suited for quantum machine learning.

1 Introduction

Quantum machine learning is known to provide atypical calculation patterns as well as relatively strong performances with small parameters compared to their classical counterparts. In addition, vanishing and exploding gradients do not apply to quantum machine learning [2] making the growing field of applying quantum machine learning algorithms to classical data an intriguing field to conduct regularization experiements. We consider the classical dropout method on VQC (varitaional quantum circuits) on down-sampled classical data translated into quantum data and see if any regularization can occur as proposed by Schuld et al.[8]. Our work requires implementing an overfitted model by scratch considering the constraints of modern NISQ machines and realistic quantum simulation, as the two main constraints in our development of a regularization techniqe. Our paper involves developing a Quantum Tensorflow version of the work presented in Schuld et al. [8] with the addition of a novel quantum dropout.

2 Related work

We closely follow the implementation of quantum classifiers described in Farhi et al. [4] and the works of Schuld et al. [8]

CS230: Deep Learning, Spring 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Preprocessing Dataset

The MNIST handwritten digits were used for our implementations of the Farhi and Schuld quantum classifiers. We focus our efforts on MNIST256, where it is transformed into real-valued data vectors of dimension 256 given [5] that both [4] and [8] use it as a benchmark. Although both works use the same dataset, their preprocessing vary greatly from one another due to distinctions in quantum embeddings.

The Farhi et al. usage of basis embedding requires a severely downsampled 4 x 4 MNIST dataset with the unconventional removal of any contradicting labels. After downsampling, a threshold is applied to create the computational basis with $N = 16$ inputs each representing a pixel. The pixels with values above the threshold are noted as "1" and "0" creating our computational basis state $|0, 1, .n_i..n_N\rangle$. This basis is encoded into the qubits with a rotation through an X gate for $n_i = 1$ where the dataset is a superposition of the computation basis states [9].

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{i}^{N} |x^{(m)}\rangle$$

We next describe the state preparation scheme done by Schuld et al. [8] Let the input vector be $\mathbb{R}^N$, where N represents the dimension of the vector. If N is a power of 2, all the amplitudes created by the n-qubit system will be utilized, and so the input vector will be straightforwardly normalized to unity. If N is not a power of 2, Schuld et al. pads the input vectors with non-zero padding terms, c_i until the padded vector is of dimension that is a power of 2. The padded vector with a total of L padded terms will then be normalized to unity as follows: [8]

$$(x_1, \cdots, x_N)^T \rightarrow \frac{1}{\sqrt{\sum_i x_i^2 + \sum_j |c_j|^2}} (x_1, \cdots, x_N, c_1, \cdots, c_L)^T$$

After normalization, Schuld et al. [8] implements the amplitude encoding; that is, the normalized input vectors are associated with the amplitudes of the n-qubit state: [7]

$$|\Psi\rangle = \sum_{i}^{N} x_i |i\rangle$$

Here, $|\Psi\rangle$ is interpreted as the quantum state, and x_i are the the elements within the N-dimensional input vector.

This by itself is non-trivial, and requires the use of a specialized state-preparation circuit, which creates a quantum state where the amplitudes of the basis states (i.e. $|i\rangle \in \{|00000001\rangle, |00000010\rangle, |00000011\rangle ...\}$ as an 8-qubit example) are given by the provided normalized values (in our algorithm these are the pixel values of the downscaled/padded/deskewed input images). We create this circuit by using the Mottonen algorithm [6].

Due to the requirements listed above, the simplest implementation will then be to choose the input vector to have a dimension of power of two. Schuld et al. uses the MNIST dataset by first deskewing them and coarse-graining the original (28,28) dimensional examples into (16,16) so that when flattened, the images can be represented as a 256-dimensional input vector, which is essentially the dimension of a 8-qubit system. [8] Thus, the input vector is straightforwardly normalized to unity.

4 Models

The models that we have considered are Farhi et al. [4] and Schuld et al. [8]. Both apply VQC (variational quantum circuits) where the decomposition of the unitary operation U_θ is expressed as:

$$U = U_L ... U_l ... U_1$$

where each U_l is a single qubit or two qubit quantum gate. The quantum gates act as parameters which are optimized and each "block" of quantum gates can be considered equivalent to a classical linear layer. To focus on the dropout regularization we will discuss the architecture of both models briefly below.

4.1 QNN by Farhi et al

For the Farhi et al. approach several unitary matrices are applied to each channel of the quantum circuit. Unitary matrices are generalized orthogonal matrices in the complex domain which is norm preserving i.e. a unitary matrix U fulfills the condition of $U^\dagger U = UU^\dagger = I$ where $U^\dagger$ is the conjugate transpose[1]. The unitary consists of several parity gates, which apply the tensor product of the specified gates. $G_1 \otimes G_1$ where G_1 is an arbitray unitary single qubit gate. Our model which follows the Tensorflow tutorial closely uses the XX and ZZ parity gate which are parameter dependent and allow for learning. With the given loss function:

$$loss(\vec{\theta}z) = 1 - l(z) \langle z, 1| U^\dagger(\vec{\theta}) Y_{n+1} U\vec{\theta} |z, 1\rangle$$

where Y_{n+1} is the readout qubit , $\langle z, 1|$ is the complex conjugate encoded computational basis $|z, 1\rangle$, $l(z)$ is the true label $[-1, 1]$. The expected readout is between $[-1, 1]$ and therefore an improvement of parameters would result in a reduction in the loss function. The QNN by Farhi is much friendlier in terms of its pre-processing given that each qubit represents a pixel and is the slimmest using the least amount of paramters1.

4.2 CC (Circuit Centric Quantum Classifier) by Schuld et al.

The Schuld implementation utilizes a four step model architecture which consists of pre-processing, learning, measuring, and post processing steps. We focus our attention to the unitary transformation applied in Step 2 and the measurement in Step 3. Step 2 is the equivalent of several classical linear layers when decomposed into gates and effectively is a difference in gate choice compared to Farhi et al[4]. The architecture involves several "blocks",denoted as r, where r represents a hyperparameter for the cyclic control gate creation. It is best to consider each block as equivalent to a "layer" in classical machine learning. To create a block single qubit gates are applied to all channels where for our implementation we specifically choose 3 Rotation gates around the X,Y,Z gates which allows for the greatest parameterization around each of the axes of the Bloch sphere. The single qubit gates are followed by two qubit control gates where for any qubit index $j \in [0, \#ofchannels]$ the block must have one controlled single-qubit gate with the jth qubit as the target and the qubit with the index $k = (j + r)(\#ofchannels)$[8].

Much of the Schuld et al. quantum classifier was developed upon our best interpretation and the closest quantum Tensorflow equivalent. Current implementations of quantum networks are very much theoretical.

The algorithm given in Schuld et al. required the use of 2 types of gates - generic one-qubit gates with 3 parameters (i.e. representing any possible gate) and controlled by similarly generic two-qubit gates. As per "B. Optimising the architecture" of the Schuld paper, the two-qubit gates can be reduced to a parameterized phase shift and a parameterized X-rotation gate, which are easily enough implemented in tfq/cirq (the cirq equivalent of the controlled phase-shift gate is the controlled-Z gate). As for the generic one-qubit gate, while tfq doesn't support specifying a gate via a parametric Pauli matrix, we were able to use the fact that any gate can be decomposed into 3 parametric rotations, around Z, then Y, and another around Z, as per Slepoy [10]; thus, we still use the same number of learnable parameters in our circuit as Schuld et al.

4.3 Overfitting

For Quantum Circuits we are limited in the number of implementable qubits and quantum gates due to current NISQ computational and runtime costs. Therefore, increasing the number of parameters to make the the model excessively flexible to exhibit overfitting proved difficult. For MNIST, we have generated irregularities in the data by filtering the true and false classes (the numbers 3 and 6 that we attempt to differentiate, inspired by Farhi et al.[4]) and then limiting the allocation of each class to 30% for "3" (and 70% for "6") in the 500-element training data and, conversely, 70% for "3" (and 30% for "6") in the 500-element testing data (all data is entries randomly picked from the MNIST database). This generates noticeable overfitting on the baseline Schuld algorithm, making the test set fit approx. 1% better on average across 10 runs.

5 Dropout Method

In our research we have discovered that because of the many approaches in embedding classical data into quantum data that regularization techniques such as dropout are difficult to generalize and implement.

To motivate the dropout algorithm described in Schuld et al. [8], we need to understand the correlation between the QNN/quantum classifier and a classic feed-forward neural network. Since quantum circuits are essentially all representable as linear operations in the Hilbert space, it is easy to see that in the classical ANN model they correspond to a linear layer (e.g. weight matrix and bias vector), while the measurement operator can be seen as corresponding to the nonlinear activation. Essentially, a quantum classifier circuit like ones built by Schuld et al. [8] or Farhi et al. [4] is similar to a feed-forward neural network with one hidden layer.

To further investigate this similarity, it is useful to think back to the amplitude encoding in the input used by Schuld et al. With that example, it's easy to see that a neuron of a classic ANN will, in this model, essentially correspond to the amplitudes of the basis states at a given point ("moment") of the circuit. Therefore, mirroring dropout from classic neural networks, a quantum dropout mechanism might focus on eliminating some of these amplitudes from the final part of the circuit, e.g. from the first and only hidden layer. Schuld achieves this by measuring a randomly chosen qubit (different from the output qubit) over several epochs. By the properties of quantum physics, measuring a qubit can collapse its state to either $|0\rangle$ or $|1\rangle$; thus, the amplitudes of all basis states of the full circuit involving the other value become 0, and measuring them would be impossible. It's easy to see that this is equivalent to applying dropout with keep-probability of 50% in a classic neural network.

Interestingly, we found that this is not possible to directly implement in Tensorflow-Quantum. The main issue is the fact that building an expectation value of the output qubit, which is the overall output of the quantum circuit and used to calculate the prediction, requires (in a "noisy" device or simulation) several measurements of the output qubit. The problem is that, if we measure the dropout qubit at the end of the circuit as well, we might get different amplitudes dropped out (depending on the dropout measurement being $|0\rangle$ or $|1\rangle$), so the different values measured for the output qubit would effectively be drawn from different circuit states; we would want to build two expectation values of the output qubit then, one corresponding to the dropout qubit measuring $|0\rangle$ and one corresponding to the dropout qubit measuring $|1\rangle$. However, that would require modifications to Tensorflow Quantum's noisy simulation backend which would be too complex for the current work.

Instead, we settled on using pure analytical simulation, having TFQ measure the expectation of both the output and dropout qubit; then, instead of feeding the measured expectation value to further layers of the hybrid quantum-classic architecture directly, we add a dense classic ANN 2x1 layer with learnable parameters (which also invokes the learnable bias term mentioned by Schuld) which takes the measured expectations of the output and the randomly chosen dropout qubit as inputs. Intuitively, this can be seen as the dense layer learning to match the analytic expectation calculations to the example output expectation measurement logic described above. This works surprisingly well on our small dataset, minimizing the generalization error without noticeably decreasing classification error for the training dataset.

6 Results

Results for MNIST Prediction using QNN, Circuit Centric			
Model Type	Validation	Test	Parameters
QNN by Farhi	85.75 %	89.92%	32
C.C. by Schuld	97.44 %	96.85%	124
C.C. w dropout	99.2%	98%	124
Fair Classical	91.31%	91.31%	32
Classical CNN	99.95%	99.95%	1198721

Table 1: Higher resolution input and a more powerful model make this problem easy for the CNN. While a classical model of similar power (32 parameters) trains to a similar accuracy in a fraction of the time [4]

Figure 1: **Accuracy and loss plot of C.C. without dropout** Our validation (blue) and training accuracy and loss (orange) for the Schuld implementation suggests high performance.

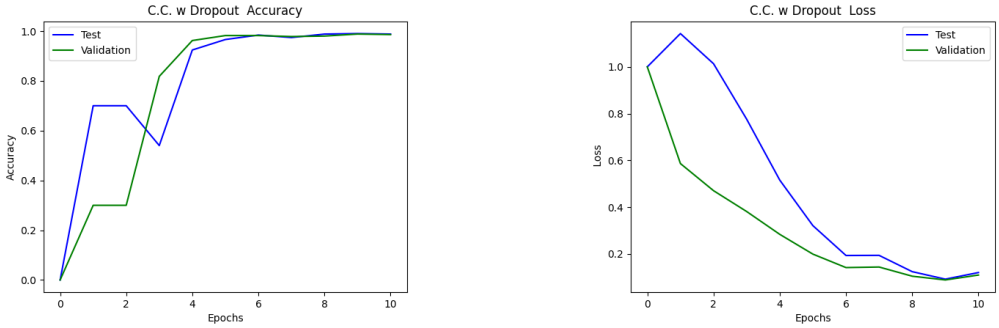

Figure 2: **Accuracy and loss plot of C.C. with dropout** Our validation (blue) and training accuracy and loss (orange) for the Schuld implementation suggests high performance representative of Schuld's implementation.

After developing an amplitude encoding according to the proposed strategy by Mottonen et al. [6] we were able to achieve sound results in both our dropout and non dropout Schuld implementation. While the implementation with dropout was able to train slightly better than the original model, we consider this to not be statistically significant and also did not see any regularization effect thus requiring further research.

7 Conclusion and Outlook

We have developed the Circuit Centric Quantum Classifier by Schuld and a quantum dropout regularization technique on Quantum Tensorflow, both firsts to the authors knowledge. Encouragingly the model performs comparatively with the paper with only a small subset of the MNIST dataset and shows to be resilient against any strategized dropout techniques. Given the current landscape of QML we see that there is much potential in creating structure and conducting research to appreciate the effects of our fully generalized quantum dropout regularization. Therefore, in the near future we intend to develop and outline a more generalized dropout technique for the Tensorflow Quantum API and to apply it to several other benchmark datasets. We also intend to investigate other regularization methods and QML architecture given our challenges in applying dropout to the basis embedding architecture by Farhi et al [4]. One possible step for the future work is to incorporate operational quantum dropout scheme proposed by by Verdon et.al. [11] as well as to investigate Quantum Convolutional Networks [3].

8 Contributions

Maxim worked on the drop out implementation for both Schuld et. al. and Farhi et. al. Ricky and Hiroki worked on the Schuld et. al. implementation including the pre-processing of the datasets.

References

[1] Martín Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. *CoRR*, abs/1511.06464, 2015.

[2] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, Nov 2019.

[3] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, Aug 2019.

[4] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors, 2018.

[5] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[6] Mikko Mottonen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. *arXiv preprint quant-ph/0407010*, 2004.

[7] Maria Schuld. *Supervised learning with quantum computers*. Springer, 2018.

[8] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101, 04 2018.

[9] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122(4), Feb 2019.

[10] Alexander Slepoy. Quantum gate decomposition algorithms. *Sandia Report*, 2006.

[11] Guillaume Verdon, Jason Pye, and Michael Broughton. A universal training algorithm for quantum deep learning. *arXiv preprint arXiv:1806.09729*, 2018.

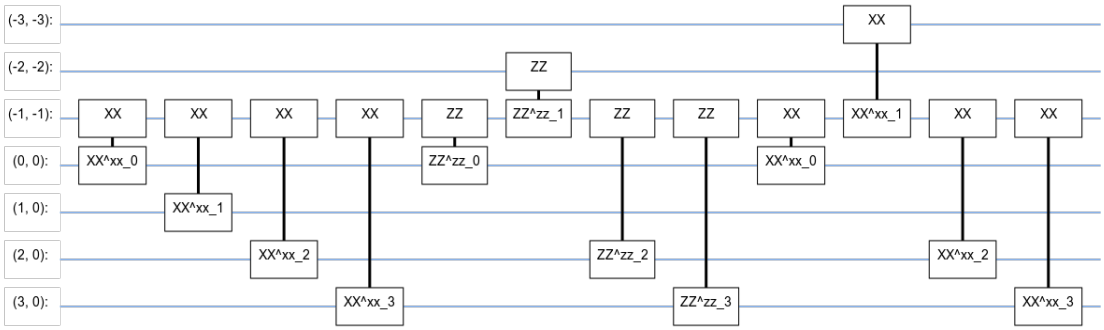

Figure 3: Simplified sample 3-layer circuit with ancillary-qubit Schuld-like quantum dropout applied. You can see the ancillary qubits, marked with negative indices, being included in each layer to replace dropped qubits.

```
958s 3s/step - loss: 0.8657 - hinge_accuracy: 0.7293 - val_loss: 0.4193 - val_hinge_accuracy: 0.8856
938s 3s/step - loss: 0.4110 - hinge_accuracy: 0.8638 - val_loss: 0.3662 - val_hinge_accuracy: 0.8911
943s 3s/step - loss: 0.3700 - hinge_accuracy: 0.8823 - val_loss: 0.3524 - val_hinge_accuracy: 0.9022
```

Figure 4: We see that without dropout that our performance is around 89% and that our epochs take around 15minutes (950 seconds).

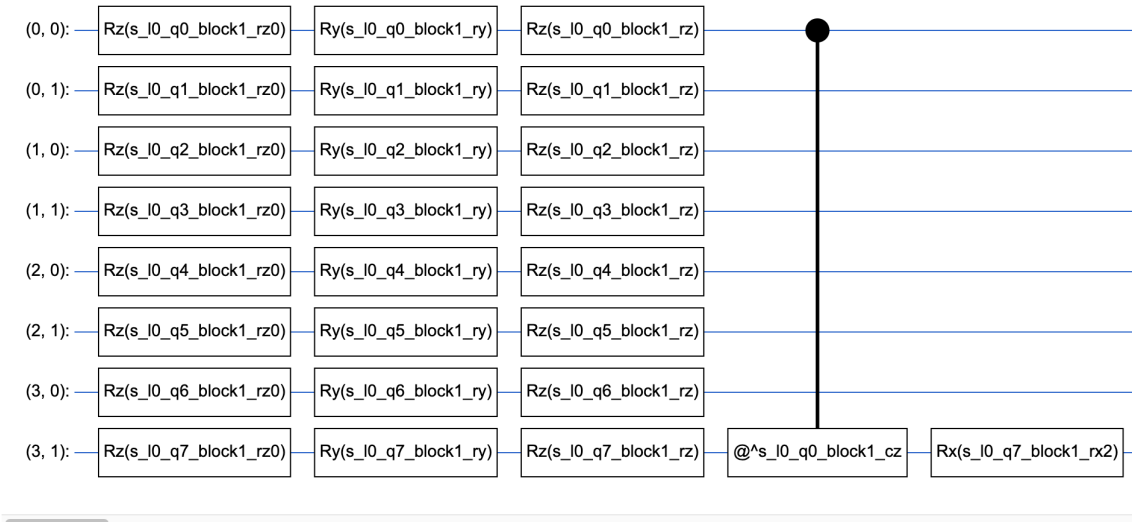

Figure 5: Fragment of Schuld circuit on 8 qubits implemented in Cirq.

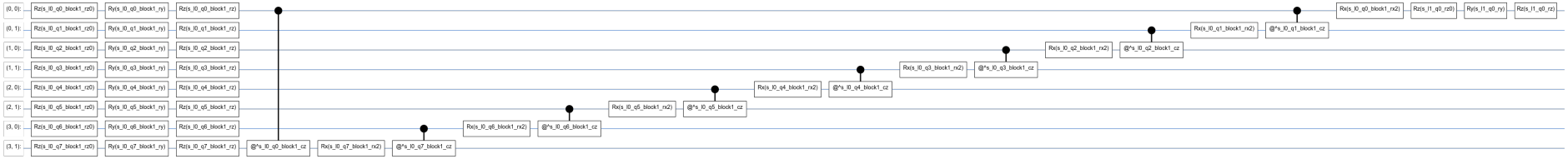

Figure 6: Full one-block (r=1) representation of Schuld QNN circuit on 8 qubits in Cirq (not counting state preparation circuit).