Interim Report

1. Introduction

a) Background knowledge

Compression of quantum states could have vast potential applications. One such example would be in quantum communication and quantum networking field. Transmission of quantum states is not a trivial task as one needs to fight decoherence and noise that threat the fidelity of the information of interest. On other hand, it costs much less recourses to keep the coherence of the quantum states if there is a representation of the states in lower dimension. Therefore, we would like to use machine learning technique to learn such a representation under certain conditions. The related field of studies are quantum information theory, machine learning and constrained optimization.

Machine learning has been a rapidly growing field in computer science and engineering in the last decade, with steady progress in image recognition, computer vision, voice recognition, medical diagnosis, search engines and so forth. Machine learning algorithms falls under the following categories: supervised training, unsupervised training and reinforcement learning. Recently, there has been intensive theoretical and in-principle experimental proofs of applying quantum mechanics to machine learning. This furry of activities has been fuelled in part by the promise of quantum technologies to provide some advantage over the classical counterparts.

An autoencoder uses machine learning to represent data in a lower dimensional space, in short, autoencoders compress data so as to reduce memory requirements. Recently, Pepper *et al* has considered a simple experimental realization of an autoencoder to reduces qutrits to qubits with low error rates. In this project, we will follow closely an experimental realization of a quantum autoencoder with qutrits [1] or more generally with a compression of qudits to qunits ($d > n$). We will look at the possible extension of the scheme in terms of inputs. The study will be extended to applications of machine learning to other possible quantum information tasks. A perspective from information theory will be adapted for investigation as well.

- An overview of quantum information

In physics and computer science fields, quantum information corresponds to the state of a quantum system. It is quantum information theory's basic study element [2]. Quantum information processing methods can be used to. Quantum Information can be processed with computers, transferred from location to location, controlled with algorithms, and examined with mathematics.

- Artificial neural network

Artificial neural networks (ANN) are computing systems that resemble biological neural networks what animal brains are composed of. However, they not identical.

Such systems "learn' to perform tasks by training through a large number of examples in order to find out certain pattern from it. The artificial neutral network generally is programmed with task-specific rules.

- Constrained optimisation

The process of objective function optimisation in respect of some constrained variables is known as constrained optimisation. For this process, it aims to minimise the cost or energy function, and to maximise the reward or utility function which are the potential candidates of objective function. Constraints can be hard constraints or soft constraints. Hard constraints set conditions for the variables that are to be satisfied. If the conditions on the variables are not satisfied, variable values of soft constraints are penalized in the objective function.

- Gradient descent

Gradient descent is a first-order recursive optimisation algorithm for locating the minima of a concave or convex function. It was initially proposed by Cauchy in 1847 [3][4]. At each point in the iteration, steps proportionate to the negative of the gradient (or approximate gradient) of the function should be chosen to compute a local minimum of a function. If steps proportionate to the positive of the gradient are taken, a local maximum of that function is approached instead. Such a protocol is termed as gradient ascent.

- Autoencoder

An autoencoder is one form of artificial neural network. It is adapted to find out faithful data compressions in an unsupervised setting, typically in terms of dimensionality reduction[5]. The target of an autoencoder is to learn a representation (encoding) for a set of data. This could be done by training the network to ignore unwanted data components. A reconstructing phase is learnt along with the reduction phase. For this stage, the autoencoder tries to generate a representation as close as possible to its original input from the reduced encoding. There are several variants being proposed aside the basic model. They have a unanimous goal of assigning the learned representations of the input with presumably meaningful attributes [6]. Some examples of autoencoder include the regularised autoencoders (proven effective in learning representations for relevant classification tasks) [7] and variational autoencoders (with their recent applications as generative models) [8]. Autoencoders are effective in solving many real-world tasks, ranging from understanding the semantic meaning of sentences to face recognition [9, 10].

b) Timeline

Considering NTU FYP schedule and studying on EEE full-time program, the project planning is roughly shown below.

Date	Tasks
Jun 2019 - Oct 2019	Literature Review and the acquiring relevant knowledge
Sep 2019 - Oct 2019	Meeting with supervisor and determine the specified FYP topic and requirements
Sep 2019	Start the project.
11 Nov 2019	Submission of Interim Report
Nov 2019 -Mar 2020	Complete most of project requirements.
Mar 2020	Completion of the draft final report and meeting with supervisor for advanced requirement to improve the project's performance
Mar 2020	Continue and modify the project
27 Mar 2020	Submission of Draft FYP report
28 Mar 2020 - 8 Apr 2020	Meeting with supervisor and modify the report and project.
9 Apr 2020	Submission of FYP report
13 - 17 Apr 2020	Project Demonstration
18 Apr 2020 - 10 May 2020	Preparation for Oral Presentation
11 - 13 May 2020	Oral Presentation

In addition, there will be a short face-to-face discussion or meeting with the supervisor every month.

c) Motivation

The purpose of the interim report is to closely monitor as well as to keep on track of the progress of the work been done in a systematic manner.

2. Work Conducted

a) Tool learnt and used: Theano

As a Python library and an optimizing compiler, Theano is commonly used for evaluating and manipulating mathematical expressions, especially matrix-valued ones [11]. In Theano, computations are expressed using a NumPy-esque syntax and compiled to run efficiently on either CPU or GPU architectures.

b) Progress made

Formulate a constrained optimisation pipeline

So far, the physical problem has been formulated in terms of a constrained optimisation problem. First, we need to define the problem:

Given a set of vectors which do not span the entire Hilbert space, find a unitary transformation that can use fewer number of basis to describe these vectors.

'Do not span the entire Hilbert space' simply means that these vectors are in principle compressible in a lossless manner. The real reason that we require a unitary transformation is that all manipulations on quantum states must be equivalent to some form of unitary operation to ensure the reversibility and to preserve the vector norm. On the other hand, unitary transformation can be interpreted in a geometrical sense as some rotation on vectors. Hence it is not difficult to see if the given set of vectors does not space the entire space in which all possible vectors exist, we can always find a rotation to re-align their bases to the canonical bases and we do not need all the canonical bases to describe them, ergo, lossless compression. However, if we are to solve the problem in constrained optimization paradigm, we will not have a closed-form solution and the final result is an approximation of the real lossless compression. How close it is to the optimal result depends on how good our strategy is.

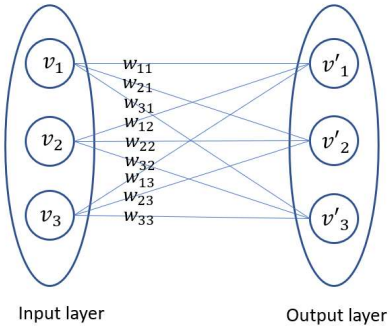

Fig. 1 Neural network for constrained optimisation

We define a two-layer perceptron as shown in Fig. 1, each layer with three nodes for simplicity. The first layer is the input layer and the three nodes have the values of each entry of the initial 3-dimential input vector. The second layer is the output layer and the three nodes have the values of each entry of the final output vector. Since the input-output layers are fully connected, there are 9 connections and 9 weights in total. Our objective is to compress the input vector, so we can set one constraint as one of the output nodes to have a minimum value (ideally 0). The second constraint is that the output vector has the same norm as the input vector as unitary transformation preserves the norm. In the case of state vector for a quantum system, the norm is always 1. Now the output vector V_{out} can be represented by the following relation with the input vector V_{in}:

$$V_{out} = \begin{bmatrix} v'_1 \\ v'_2 \\ v'_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = WV_{in}$$

Notice that all the weights collectively can be thought as entries of some square matrix W that operates on V_{in}. This coincides with out objective of finding some specific transformation, which can be viewed as a matrix under certain assigned bases. In fact, it can be proved that if we ensure that V_{in} and V_{out} have the same norm, W is

indeed unitary. Further notice that in the context of deep learning, the activation function is always one in order to have the relation in equation (1).

Now it is rather straight forward to specify the objective function to be optimized. It would be to minimize such a squared-error cost function:

$$L = \sum([w_{11} \quad w_{12} \quad w_{13}]V_{in})^2 + \lambda\left([w_{21} \quad w_{22} \quad w_{23}]{V_{in}}^2 + [w_{31} \quad w_{32} \quad w_{33}]{V_{in}}^2 - 1\right)^2$$

The first term is to minimize the first entry of V_{out}(ideally 0). The second term is to impose the unit norm constraint. λ can be considered as the Lagrange multiplier which tunes the relative importance of the second term with respect to the first term (i.e. Which constraint has the priority in the optimization scheme).

Now perform gradient descent technique on the cost function, differentiating it with respect to all the weights in the hope of finding a local minimum.

c) Caveat

The gradient descent could and has been implemented in 2 following 2 forms:

1) Gradient descent in real space

This is the normal type of gradient descent which can be seen in many places.

2) Gradient descent in complex space

Since Theano itself does not support complex space gradient descent, we have to formulate the problem in a different way:

Define a complex number $z = x + iy$. Then

$$dz = dx + idy$$

and

$$\frac{\partial}{\partial \mathrm{z}} = \frac{\partial}{\partial \mathrm{x}} + i\frac{\partial}{\partial \mathrm{y}}$$

since differentiation is a linear operation. It means that we can treat the real part and the complex part of the weights separately.

However, things get more complicated for complex number multiplication. Assume $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$, then

$$\begin{aligned} z_1 z_2 &= (x_1 + iy_1)(x_2 + iy_2) \\ &= x_1x_2 + iy_1x_2 + ix_1y_2 - y_1y_2 \\ &= (x_1x_2 - y_1y_2) + i(y_1x_2 + x_1y_2) \end{aligned}$$

Therefore, by abiding to this rule, we can define to variables x and y to going through the gradient descent process and combine the final result to give the adjusted complex number z.

d) Tasks completed

The results of compressing 2-dimensional vectors and 3-dimentional real and complex vectors been obtained.

results obtained

1) Real-space gradient descent (3-D input vector)

The unitary matrix obtained

$$W = \begin{bmatrix} -0.000417519 & -0.00521565 & 0.0116655 \\ 0.264319 & 0.411594 & 0.473851 \\ 0.896402 & 0.351494 & 0.00274371 \end{bmatrix}$$

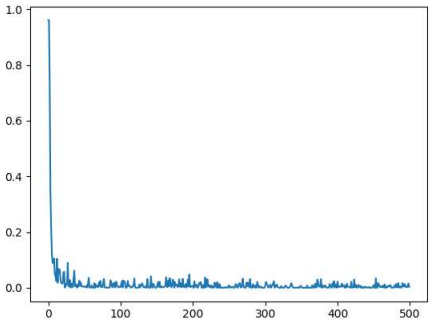

Fig. 2(a) Loss versus epoch

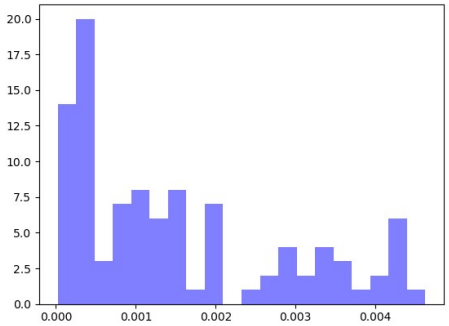

Fig. 2(b) Histogram of the first entry of the output vector during testing phase

norm variance: 0.005286588038536152
norm mean: 1.0290477032330845

2) Complex-space gradient descent (3-D input vector)

The unitary matrix obtained (rounded to 6 significant figures):

$$W = \begin{bmatrix} -8.84134\text{e}-05 + 0.000184444\text{j} & -0.00473946 - 0.00283093\text{j} & 0.00807089 + 0.00407159\text{j} \\ -0.294443 + 0.577051\text{j} & 0.645851 + 0.332697\text{j} & 0.400190 - 0.368305\text{j} \\ 0.749248 + 0.177904\text{j} & -0.154940 + 0.581374\text{j} & 0.441941 + 0.318514\text{j} \end{bmatrix}$$

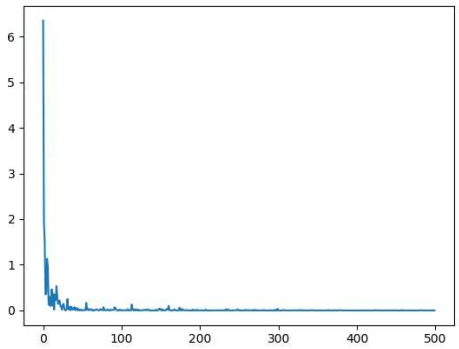

Fig. 3(a) Loss versus epoch

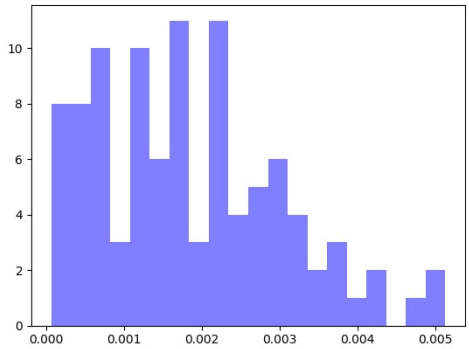

Fig. 3(b) Histogram of the first entry of the output vector during testing phase

norm variance: 0.0008801749392840856
norm mean: 1.0015020011237408

e) In view of information entropy

We can view the compression problem in terms of its entropy. We try to compute the information entropy of our initial and final data. Von Neumann entropy is a generalization of Shannon entropy for quantum states. Shannon entropy is defined as

$$H(X) \equiv H(p_1, p_2, .., p_n) \equiv -\sum_x p_x \, log_2 p_x$$

While Von Neumann entropy is defined as

$S(\rho) \equiv -\mathrm{tr}(\rho \, log \, \rho) = -\sum_x \lambda_x \, log_2 \lambda_x$

Where ρ is the density matrix of the quantum system and defined as

$$\rho \equiv \sum_i p_i \, |\psi_i\rangle\langle\psi_i|$$

where $|\psi_i\rangle$ is the input state vector and $\langle\psi_i|$ is its conjugate transpose (transpose in the case of real vectors). λ_x are the eigenvalues of ρ.

In the 3-dimensional complex vector case (with constraint),

$$\rho \equiv \sum_i p_i \, |\psi_i\rangle\langle\psi_i| = \frac{1}{3}\begin{bmatrix}1\\0\\0\end{bmatrix}[1 \quad 0 \quad 0] + \frac{1}{3}\begin{bmatrix}0\\1\\0\end{bmatrix}[0 \quad 1 \quad 0] + \frac{1}{3}\begin{bmatrix}0\\0\\1\end{bmatrix}[0 \quad 0 \quad 1] = \frac{1}{3}\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

where the bases are not canonical. The Von Neumann entropy of the uniformly distributed collection of input vectors input is

$$S(\rho) \equiv S(\rho) = -\sum_x \lambda_x \, log_2 \lambda_x = -3\left(\frac{1}{3} log \frac{1}{3}\right) \approx 1.585$$

Since the output could be fully described by three canonical axes, the Von Neumann entropy would be the same (this point will be explained more in detail in the final report).

f) Potential problems identified

From Shannon's source coding theorem, in principle such a compression process could be lossless. However, since the unitary operation we found is rather an approximation since gradient descent is a numerical method, there will be inaccuracy.

The norm of each entry of the initialized random unit vectors for training weren't generated uniformly (see Fig.) despite fulfilling the normalization condition. This might cause the training to be biased. One possible solution is to reverse engineer uniformly distributed vectors. However, in this case how to decompose the vectors into their standard normal basis is not immediately obvious.

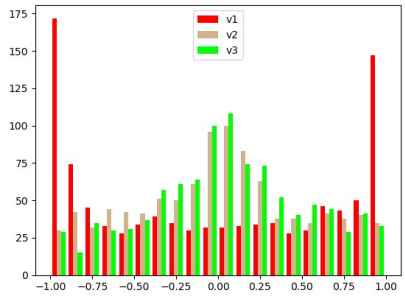

Fig. 4(a) Histogram of real input vectors (1000 samples)

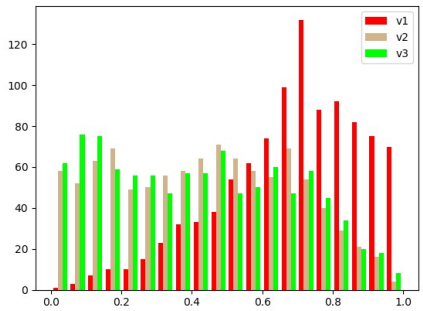

Fig. 4(b) Histogram of the norms of complex input vectors (1000 samples)

3. Future Work

a) Compare to PCA

Principle component analysis is a very commonly practised data dimensionality reduction technique. In the future work we will investigate whether such a technique is useful for our task and whether is it possible to generalize our approach to other applications and scenario. Compared to a general problem, the additional constraints we have here is unit norm of the input and output vectors and the process is equivalent to a unitary operation.

b) Mixed state system

So far, we only have a pure state quantum system. The classical probability in the density matrix arose from the fact that we chose different pure states to transmit different message in practical setting. However, the same method presumably could be generalized into for mixed quantum states as well. For future work, we are to investigate how to adapt the constrained optimisation technique we have been using so far to compress mixed states.

c) Changes compared to the original approach

Originally, as proposed by one of the papers which we have been following closely, we planned to use a well-studied neural network structure, namely auto-encoder, to compress the state vectors and re-construct them to compute the loss. However, it

turned out that the compression process must be a unitary process the at the middle part we cannot reduce the number of nodes as a convectional auto-encoder does. Moreover, we proved that as long as we ensure the two constraints mentioned before, the output vector will be the one of our interest. We can decompress the output vector simply by applying the inverse of the trained unitary matrix since unitary transformation is fully reversible. Therefore, there is no need to implement the decoder part. Our final neural network structure simply consists of one input layer and one output layer without hidden layer, which is capable of handling our defined task.

4. Conclusion

So far, the project is in the right track of progressing. The task is clearly defined and the approach is rather effective. As shown in the results, the lost versus epoch plots looks promising and the histogram of the first entry shows the compression is quite effective as well. Norm mean is very close to 1 and norm variance is rather small. All that means our two constraints are satisfied to a considerable extent after tuning the hyperparameters such as the learning rate and Lagrange multiplier. As mentioned before, this numerical solution won't be as accurate as an analytical result, so the next step is to find the unitary matrix that is closest to the trained one.

Reference

[1] A. Pepper, N. Tischler, and G.J. Pryde, Experimental Realization of a Quantum Autoencoder: The Compression of Qutrits via Machine Learning, Phys. Rev. Lett., 122, 060501 (2019).

[2] Nielsen, Michael A. (2010). Quantum computation and quantum information. Chuang, Isaac L. (10th anniversary ed.). Cambridge: Cambridge University Press. ISBN 978-1107002173. OCLC 665137861.

[3] Dimitri P. Bertsekas, Nonlinear Programming, Athena Scientific 1999, 2nd edition, pp. 187.

[4] Cauchy, Augustin. "Méthode générale pour la résolution des systemes d'équations simultanées." Comp. Rend. Sci. Paris 25.1847 (1847): 536-538.

[5] Kramer, Mark A. (1991). "Nonlinear principal component analysis using autoassociative neural networks" (PDF). AIChE Journal. 37 (2): 233–243. doi:10.1002/aic.690370209

[6] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). Deep Learning. MIT Press. ISBN 978-0262035613.

[7] Vincent, Pascal; Larochelle, Hugo (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". Journal of Machine Learning Research. 11: 3371–3408.

[8] Diederik P. Kingma, Max Welling, An Introduction to Variational Autoencoders, arXiv:1906.02691

[9] Liou, Cheng-Yuan; Huang, Jau-Chi; Yang, Wen-Chie (2008). "Modeling word perception using the Elman network". Neurocomputing. 71 (16–18): 3150. doi:10.1016/j.neucom.2008.04.030.

[10] Hinton GE, Krizhevsky A, Wang SD. Transforming auto-encoders. In International Conference on Artificial Neural Networks 2011 Jun 14 (pp. 44-51). Springer, Berlin, Heidelberg.
[11] Bergstra, J.; O. Breuleux; F. Bastien; P. Lamblin; R. Pascanu; G. Desjardins; J. Turian; D. Warde-Farley; Y. Bengio (30 June 2010). "Theano: A CPU and GPU Math Expression Compiler" (PDF). Proceedings of the Python for Scientific Computing Conference (SciPy) 2010.