ROOT2020 Final Research Report

Title: Toward the establishment of new image recognition method based on machine learning by quantum computers.

Author: Keiichiro HAYASHI, Nada High School

Prepared as Final Research Report for the Advanced Stage in the GSC/ROOT Program.

Advisors: Satofumi SOUMA, Kobe University

Date: February 28, 2022

Abstract:

古典的なニューラルネットワークを模した量子回路を組み立てることにより、量子画像認識機械学習を行った。本研究の特色としては、同時摂動法を用いて計算回数を極力減らしたこと、重みパラメータの最適化のために勾配ベクトルを求めて学習精度と学習効率を高める、使用する量子ビットの数や重み層が増えた場合にも対応したアルゴリズムを設計したしたことだ。6 qubit を用いて重み層の数も何通りか試しながら検証を行った。$C^{\otimes n}Z$ ゲートだけでは当手法の有用性までは示せなかったが、新たな量子画像認識機械学習手法の大枠を完成させるに至った。今後$C^{\otimes n}RZ$ ゲートの導入で、入力値の高い表現性と勾配ベクトルを生かた学習精度と学習効率の向上が期待できる。

1. Abstract
2. Introduction (the background and the significance of this study)
3. Method (trajectory of thinking and about the source code)
4. Result (the outcome of the execution)
5. Discussion (consideration based on the result)
6. Conclusions
7. Acknowledgements
8. References
9. Appendix

Introduction

本研究では、近年注目されている量子コンピュータの応用アルゴリズムの一つとして下記論文で提案されている量子機械学習、特に量子ニューロンの考え方に基づき、この量子ニューロンに対する入出力部分の具体化に関する未検討課題と、機械学習による重みの最適化取り組む。

先行研究において量子機械学習の原理が提案されているが、実用例としてこれを画像認識に応用しようとした場合に、入力画像をどのように量子ニューロンへの入力とするのか、また、量子的なプロセスの結果からどのように「画像を認識できた」と判断するのか、といった「具体的な問題に適用しようとした場合に顕在化する課題」についてはこれまで十分に検討されているとはいえない。このような現実的な問題に向き合うことによって、新たに従来の手法との類似性や差異を顕在化させることが出来ると考えられる。また、画像判別のプロセスにおいて重み層を複数層用意し、より実践的な画像認識機械学習を試みた。本研究ではノイズの影響を一旦無視し、量子プログラミングのためのオープンソースである Qiskit を用いてこの課題に取り組み、実用化の基盤を構築する。

Materials and Methods / Theory / (or any appropriate name of the section)

古典的な機械学習におけるニューラルネットワークでは、入力情報をノードにマッピングし、重みを掛け合わせ、それが閾値を上回れば発火するというプロセ

スを何度も行い、最終的な出力から元の入力情報を分類したり、値を予測したりする(fig 1).本研究において入力する情報をどのように量子状態にマッピングするか、またどのように重みを掛け合わせるかについての着想は、先行研究から得た[1].入力情報がマッピングされたベクトルを $\vec{i}$ とし、それに掛け合わせる重みのベクトルを $\vec{w}$ とする.

$$\vec{i}=\begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w}=\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} \tag{1.1}$$

$|\psi_i\rangle, |\psi_w\rangle$.このとき $i_j, w_j \in \{-1,1\}$ であり、これらの量子状態、$|\psi_i\rangle, |\psi_w\rangle$ を定義する.

$$|\psi_i\rangle = \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_j |j\rangle, |\psi_w\rangle = \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_w |j\rangle \tag{1.2}$$

$|j\rangle \in \{|00...00\rangle, |00...01\rangle, ..., |11...11\rangle\}$ であり、0 と 1 だけの数列は 10 進数である j の 2 進数表記であると解釈する.量子ビットはそれぞれ $|0\rangle, |1\rangle$ であり、N 個全ての量子状態が $|0\rangle$ であるとき、量子状態は $|00...00\rangle \equiv |0\rangle^{\otimes N}$ である.$|\psi_i\rangle, |\psi_w\rangle$ はこれらの複数の量子状態の重なり合わせで表現される.この考え方で簡易的な量子回路の構造を考える(fig 2).ここで、U_i は初期状態 $|0\rangle^{\otimes N}$ から $|\psi_i\rangle$ の量子状態を作るユニタリゲートである.また U_w は $|\psi_w\rangle$ で表される重みを掛け合わせるユニタリゲートである.最終的に発火を $C^{\otimes N}X$ ゲートによる Ancilla qubit の反転に委ねるとする.

$$U_i|0\rangle^{\otimes N} = |\psi_i\rangle, U_w|\psi_w\rangle = |1\rangle^{\otimes N} = |m-1\rangle \tag{1.3}$$

このとき $m = 2^N$ である.U_w が $|\psi_w\rangle$ という重みをかけるとき、U_w が適用される直前の量子状態が $|\psi_w\rangle$ ならば、U_w 適用後の N 個の量子ビットが全て $|1\rangle$ になり $C^{\otimes N}X$ ゲートは Ancilla qubit を反転し、測定値は 1 になる.入力情報に対応した $|\psi_i\rangle$ に U_w が適用されるとき、

$$|\phi_{i,w}\rangle \equiv U_w|\psi_i\rangle = \sum_{j=0}^{m-1} c_j |j\rangle \tag{1.4}$$

(Eq 1.3)と(Eq 1.4)の定義を用いると、二つの量子状態の内積は、

$$\langle\psi_w|\psi_i\rangle = \langle\psi_w|U_w^\dagger U_w|\psi_i\rangle = \langle m-1|\phi_{i,w}\rangle = c_{m-1} \tag{1.5}$$

である.$|c_j|^2$ は $C^{\otimes N}X$ ゲートが適用される直前に N 個の量子ビットの量子状態が $|j\rangle$ である確率を示す.そのため $|c_{m-1}|^2$ は量子状態が $|11...11\rangle$ である確率、つまり Ancilla qubit が反転する確率を示す.

U_i, U_w の内訳を考える.簡単のために $N = 2$ の場合を考える. 4 マスの白黒画像の各マスが黒なら -1, 白なら 1 を i_n に割り当て $i_0, i_1, i_2, i_3, \in \{-1, 1\}$ とし, $|\psi_i\rangle = \frac{1}{\sqrt{4}}\left(i_0|00\rangle + i_1|01\rangle + i_2|10\rangle + i_3|11\rangle\right)$ と表す.また各マスが黒なら 0,白なら 1 を j_n に割り当てると、各画像は $k_i \equiv j_3 j_2 j_1 j_{0(2)}$ で表現できる(fig 3).初期状態 $|0\rangle^{\otimes 2}$ に $H^{\otimes 2}$ を適用した直後は $j_0 = j_1 = j_2 = j_3 = 1$ である。ここで $C^{\otimes n}Z$ ゲートを導入する。

$$C^{\otimes n}Z_{\vec{l}}\left|j\right\rangle=\begin{cases}-\left|j\right\rangle \ if \bigcap_{i=0}^{N-1} j_i \geq l_i \\ \left|j\right\rangle \ otherwise\end{cases} \tag{1.6}$$

$\vec{l}$ は N 個の要素を持ち、$l_i \in \{0,1\}$ である。本来マルチコントロールゲートは特定の qubit が全て $|1\rangle$ であるとき対象の qubit に操作を適用するというものである。Z ゲートはブロッホ球において Z 軸周りに π 回転させる操作であり、$|0\rangle$ の量子状態に適用しても変化はなく、$|1\rangle$ の量子状態に適用すると位相の逆転が起きる(fig 4)。そのためマルチコントロール Z ゲートは、”特定の qubit”と対象の qubit の全ての量子状態が $|1\rangle$ であるとき位相が逆転する(fig 5).”特定の qubit”と対象の qubit の場所を 1,それ以外を 0 とした配列を $\vec{l}$ とした。N=2 のとき $C^{\otimes 0}Z_{\{1,0\}}$ ゲート(fig 6.1)は $|10\rangle,|11\rangle$ の、$C^{\otimes 0}Z_{\{0,1\}}$ ゲート(fig 6.2)は $|01\rangle,|11\rangle$ の、$C^{\otimes 1}Z_{\{1,1\}}$ ゲート(fig 6.3)は $|11\rangle$ の符号を反転させる。(Eq 1.6)ではこれを一般化した。$n+1$ は $\vec{l}$ の中の 1 の個数である。ただこれらのゲートでは $|00\rangle$ の符号を反転できないため、X ゲートを用いてビットの係数を交換して調整する。これを元に $N=2$ の場合において U_i, U_w の中身を書き出した(fig 7)。試しに $k_i = k_w = 2$ の回路を例とする(fig 8). U_i の中で，$C^{\otimes 0}Z_{\{0,1\}}$ ゲート，$C^{\otimes 1}Z_{\{1,1\}}$ ゲートの後の量子状態は $\frac{1}{\sqrt{4}}\left(|00\rangle+|01\rangle-|10\rangle+|11\rangle\right)$, $X^{\otimes 2}$ ゲートが適用された後は $|00\rangle \leftrightarrow |11\rangle, |01\rangle \leftrightarrow |10\rangle$ と入れ替わるため $\frac{1}{\sqrt{4}}\left(-|00\rangle-|01\rangle+|10\rangle-|11\rangle\right)$ となっている($0010_{(2)}=2$)。その後 U_w で $C^{\otimes 0}Z_{\{0,1\}}$ ゲート，$C^{\otimes 1}Z_{\{1,1\}}$ ゲートが適用されると $\frac{1}{\sqrt{4}}\left(-|00\rangle+|01\rangle+|10\rangle-|11\rangle\right)$,

$H^{\otimes 2}$ のあとは $|11\rangle$ であり、最終的に Ancilla qubit は反転され 1 が観測される。$k_i = k_w$ のときだけ、観測値は 1 になる。$N = 2$ のときだからこそ全てのパターンを書き出せたが、パターン数は 2^{2^N} 通りあるため、qubit 数が増えたときのために関数化する必要があった(Appendix)。

最終的な観測値は必ず 0 か 1 で、どちらが観測されるかは確率によるものだ。観測値の平均値をとり、これを内積の期待値とした。

$$f\left(k_i, k_w\right) \equiv E\left(\left\langle \psi_w \middle| \psi_i \right\rangle\right) = \frac{1}{\kappa}\sum_{k=1}^{\kappa}\left\langle \psi_w \middle| \psi_i \right\rangle \tag{1.7}$$

今回は $\kappa = 1024$ として行った。

実行時間との兼ね合いにより、至極情報の少ないデータセットで画像認識機械学習の検証をする必要があった。8×8 マスの〇と×の画像をそれぞれ 100 枚ずつ用意し(fig 9)、これらから特徴量を学習して重みを最適化し、〇か×かを識別する問題を設定した。〇ならば 1,×ならば 0 を出力するモデルを目指すため、損失関数を次のように設計した。

$$L\left(k_w\right) = \frac{1}{2m}\sum_{j=1}^{m}\left[\left\{1 - f\left(k_{i,j}^{circle}, k_w\right)\right\}^2 + \left\{f\left(k_{i,j}^{cross}, k_w\right)\right\}^2\right] \tag{1.8}$$

k_i は画像の各マスの白と黒から 8×8 桁の 2 進数を作りそれを愚直に 10 進数に変換したものとした。適当に $k_{w,0}$ に初期値を与え、損失関数が最小化する方向に k_w を更新する。

$$k_{w,t+1} = k_{w,t} - \eta\frac{dL}{dk_{w,t}} \tag{1.9}$$

$$L'\left(k_{w,t}\right)=\frac{L\left(k_{w,t}+dk_{w,t}\right)-L\left(k_{w,t}-dk_{w,t}\right)}{2dk_{w,t}} \tag{1.10}$$

重みの変化は bit に見立てた$\left|00...00\right\rangle$~$\left|11..11\right\rangle$の各量子状態の係数ごとに偏微分しなければならない。

$$L'\left(k_{w,t}\right)=\left(\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,0}},\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,1}},\cdots,\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,2^N-1}}\right)^{\mathrm{T}} \tag{1.11}$$

$i_{t,j}$はt回の更新を経た$k_{w,t}$の表す量子状態の内の$\left|j\right\rangle$の係数である。$\partial i_{t,j}$を各マスの変化、つまり白が黒へ、黒が白へ変化する様子を示すこととする。jの2進数表記の各桁の数字について位相が2の有限体を考える。

$$\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,j}}=\frac{L\left(\left(k_{w,t(2)}+e_{t,j}\right)_{(10)}\right)-L\left(k_{w,t}\right)}{1}=L\left(\left(k_{w,t(2)}+e_{t,j}\right)_{(10)}\right)-L\left(k_{w,t}\right) \tag{1.12}$$

$e_{t,j}$はj番目の桁だけ1でそれ以外は0の二進数である。(Eq 1.10)の差分近似の考え方ではこの場合不都合が生じるため(Eq 1.12)の形をとる。1度のk_wの更新のために$L\left(k_{w,t}\right)$の勾配ベクトルを求めるには、Lの値を$\left(2^N+1\right)$回求める必要があるが、これでは計算量が莫大になる。そこで同時摂動最適化手法を導入して計算量の軽減を図る。

$$k_{w,t+1}=k_{w,t}-\eta\Delta k_{w,t} \tag{1.13}$$

$$\Delta i_{t,j}=\frac{L\left(k_{w,t}+c_t\right)-L\left(k_{w,t}-c_t\right)}{2c_{t,j}} \tag{1.14}$$

$$\Delta k_{w,t} = \Delta i_{t,2^N-1}\Delta i_{t,2^N-2}\cdots\Delta i_{t,1}\Delta i_{t,0} \tag{1.15}$$

$\Delta k_{w,t}$ は $k_{w,t}$ における偏微分の推定値を表している。$c_t, c_{t,j}$ はそれぞれ摂動値とその 2 進数変換時の j 桁目を表す。2 進数に変換し各桁について位相 2 の有限体として $k_{w,t}-\Delta k_{w,t}, k_{w,t}+c_t$ の処理を行う。摂動 c_t の生成は、確率分布を元にしたベルヌーイ乱数を用いた。$c_{t,j}$ は t, j に対して i.i.d.(独立同分布)で、その分布は 0 を中心に対称で大きさは一様有解である。(Eq 1.14)では全ての j に関して $L\left(k_{w,t}+c_t\right)-L\left(k_{w,t}-c_t\right)$ が一定であるため、一度の k_w の更新のために $L\left(k_{w,t}\right)$ の勾配ベクトルを求めるには、L の値を 2 回求めるだけで良いため、(Eq 1.12)と比べ計算量を軽減できた。今回の研究では $\Delta i_{t,j}, c_{t,j}$ がそれぞれ 0 か 1 に限定されていることから、(Eq 1.16)を採用する。

$$\Delta i_{t,j} = \begin{cases} 1 \ \textit{if} \ \left(L\left(k_{w,t}+c_t\right)-L\left(k_{w,t}\right)\right)<0 \cap \ c_{t,j}=1 \\ 0 \ \textit{otherwise} \end{cases} \tag{1.16}$$

また一般的にニューラルネットワークにおける中間層を増やすことで学習精度や学習効率が向上することが示唆されている。$k_w \in \Re^{\gamma}$ の場合、つまり重みが γ 層に多層化された場合でも検証を行った(fig 10)。(Eq 1.4)に則ると次のように定義される。

$$\left|\phi'_{i,w}\right\rangle \equiv \prod_{s=1}^{\gamma} U_{w,s}\left|\psi_i\right\rangle \tag{1.17}$$

U_w をそのまま γ 回繰り返す構造を設計した。$\overrightarrow{k_w}$ は要素数 γ の 1 次元ベクトルで

γ個のパラメータは互いに独立である。

$$g\left(k_i,\overrightarrow{k_w}\right)\equiv E\left(\left\langle\psi'_w\middle|\psi_i\right\rangle\right)=\frac{1}{\kappa}\sum_{k=1}^{\kappa}\left\langle\psi'_w\middle|\psi_i\right\rangle \tag{1.18}$$

便宜的に(Eq 1.18)のように表記した。

$$L\left(\overrightarrow{k_w}\right)=\frac{1}{2m}\sum_{j=1}^{m}\left[\left\{1-g\left(k_{i,j}^{circle},\overrightarrow{k_w}\right)\right\}^2+\left\{g\left(k_{i,j}^{cross},\overrightarrow{k_w}\right)\right\}^2\right] \tag{1.19}$$

$$k_{w,s,t+1}=k_{w,s,t}-\eta\Delta k_{w,s,t} \tag{1.20}$$

$$\Delta i_{s,t,j}=\frac{L\left(\overrightarrow{k_{w,t}}+\overrightarrow{c_t}\right)-L\left(\overrightarrow{k_{w,t}}-\overrightarrow{c_t}\right)}{2c_{s,t,j}} \tag{1.21}$$

$$\Delta k_{w,s,t}=\Delta i_{s,t,2^N-1}\Delta i_{s,t,2^N-2}\cdots\Delta i_{s,t,1}\Delta i_{s,t,0} \tag{1.22}$$

この場合、層を増やしても、損失関数 L 自体の計算は長くなるが、1 度の$\overrightarrow{k_w}$の更新のために$L\left(\overrightarrow{k_{w,t}}\right)$の勾配ベクトルを求めるには、$L$の値を 2 回計算するだけで良い。やはり本研究では$\Delta i_{s,t,j},c_{s,t,j}$がそれぞれ 0 か 1 に限定されていることから、(Eq 1.24)を採用する。

$$\Delta i_{s,t,j}=\begin{cases}1\ \ if\ \ \left(L\left(\overrightarrow{k_{w,t}}+\overrightarrow{c_t}\right)-L\left(\overrightarrow{k_{w,t}}\right)\right)<0\cap\ c_{s,t,j}=1\\ 0\ \ otherwise\end{cases} \tag{1.23}$$

$\gamma=1,3,5$について各パラメータの初期値$k_{w,s,0}=0$としてそれぞれ 500 回重みを更新しながら画像認識機械学習を試み、比較した。

Results

epoch を横軸に、(Eq 1.19)の計算結果を Energy としてプロットした。重みはそれまでの損失関数が最小値をとったときのものを採用し更新する。損失関数の最小値の変動も Minimum Energy としてプロットした(fig 11)。その際の初期値と最終値をまとめた(fig 12)。

Discussion

検証結果より、重みを更新し損失関数を減少させることに成功した。今回の検証では重み層を増やすことによる学習精度の向上は認められなかった。精度に関して機械学習の成功と言うには及ばないが、少ないテストデータと最小限のパラメータで行っての検証としては成功したと言える。

今回の研究は著者なりの量子画像認識機械学習の手法についての骨格を作ることに重きを置いた。量子コンピュータで演算を行う利点は量子の重ね合わせ状態を活かした計算能力を駆使できることにある。

$|j\rangle \in \{|00...00\rangle, |00...01\rangle, ..., |11...11\rangle\}$ と表すように、N 個の qubit の 0 と 1 の組み合わせを、H ゲートによって確率的に重ねあわせることが出来る。量子回路でのゲート操作で、これらの量子状態の係数、観測確率を操作できる。今回使った $C^{\otimes n}Z$ ゲートは特定の qubit が 1 のときブロッホ球上で Z 軸周りに位相を π 回転させる、つまり係数の符号を入れ替える操作のため、(fig 9)における最後の $H^{\otimes N}$

がなければ、どのタイミングで$C^{\otimes n}X$を適用し観測しても 1 が発火する確率は$\frac{1}{2^N}$である。今後位相の回転を連続値として設定できる$C^{\otimes n}RZ$ゲートを用いることで、表現性が増すと考えられる。係数が 1 または-1(厳密には$\pm 2^{-\frac{N}{2}}$)の二択の今回の場合は、N 個の qubit を用いると$2^{\frac{N}{2}} \times 2^{\frac{N}{2}}$マスまでの白黒画像を量子状態で表現できた。この場合はk_wは整数値しかとらず、強引に微分で変化量を図ろうにも、微小変化量の最小値は 1 である。またk_wの値によってはこの変化量には大きすぎる差がある。一連の内積計算の手法を踏まえると、k_wの 1 の変化は$|11...11\rangle$の係数として割り当てられた 1 マスだけの変化であるときと、二進数の繰り上げの際に特定の複数のマスが変化する場合があることが想像できる。古典的なプロセスにおいては各ノードの重みについてそれぞれ偏微分をして微小変化量を求めるが、今回の場合は bit に見立てた$|00...00\rangle$~$|11..11\rangle$の各量子状態の係数が-1 か 1 かに変化するだけであり、複数の意味で微分不可能であったため、無理矢理に(Eq 1.21)から(Eq 1.23)を作った。本研究は今後$C^{\otimes n}RZ$ゲートを導入した際にも生きるメソッドを採用している。

そもそも二進数に変換して各桁について有限体の考え方を使うという強引な手順を踏んでいるのも、$C^{\otimes n}RZ$ゲートを導入すれば解決できる。本研究では各桁を位相 2 の有限体上で微分するという、一見支離滅裂と思えるプロセスを採用しているが、係数i_jは$C^{\otimes n}Z$ゲートではなく$C^{\otimes n}RZ$ゲートを使えば連続値に設定

できるからだ。結局”位相”θを使えば2πで一周するから有限体の考え方は不要になる。ランダムにパラメータを更新してそれまでより適切なものがあれば採用するという手荒なやり方になってしまったが、連続値を用いて偏微分を可能にすることで、学習精度や学習効率が向上すると確信している。その場合、今度は2進数変換をして各桁を連続値とすることが意味不明であるが、これについては今後の検討課題と言える。

摂動値はベルヌーイ乱数を用いて期待値を0.5と設定して検証を行ったが、他に最適な数値があるはずだ。また量子コンピュータに発生するノイズを摂動に落とし込めないかという発想にも至った。ゲイン係数ηも変化させる先行研究も見当たり、今後更なる検討をする必要がある。

Conclusions

今回の採用した量子画像認識機械学習の手法の有用性を示すには至らなかったものの、当手法の大枠を作ることが出来た。この手法の長所は同時摂動法を用いて計算量を軽減したこと、重み層を多層化したことである。今後$C^{\otimes n}RZ$ゲートの導入で、入力値の表現性と勾配ベクトルを生かた学習精度と学習効率の向上が期待できる。

Acknowledgements

This work was supported by ROOT program. I am grateful to Prof. Satofumi Souma for his kind guidance and advice in this research.

References/Bibliography

[1] F. Tacchino, C. Macchiavello, D. Gerace, D. Bajoni, “An artificial neuron implemented on an actual quantum processor,” npj Quantum Information 5, 1-8 (2019).

[2] S. Mangini, F. Tacchino, D. Gerace, C. Macchiavello, D. Bajoni, “Quantum computing model of an artificial neuron with continuously valued input data,”Mach. Learn.: Sci. Technol. 1 045008 (2020).

[3] F. Tacchino, C. Macchiavello, D. Gerace, D. Bajoni, “Variational learning for quantum artificial neural networks”

[4] F. Tacchino, P. Kl. Barkoutsos, C. Macchiavello, D. Gerace, I. Tavernelli, D. Bajoni, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), 130-136 (2020).

[5] F Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, “Quantum implementation of an artificial feed-forward neural network,” Quantum Sci. Technol. 5 044010 (2020).

[6] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, C. Macchiavello, “Quantum computing models for artificial neural networks” EPL (Europhysics Letters) 134, 10002 (2021).

[7] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, “Parameterized quantum circuits as machine learning models,” Quantum Sci. Technol. 4 043001 (2019).

[8] A. Chalumuri, R. Kune, B. S. Manoj, “Training an Artificial Neural Network Using Qubits as Artificial Neurons: A Quantum Computing Approach,” Procedia Computer Science, 171, 568-575 (2020).

[9] Y. Li, R-G. Zhou, R. Xu, J. Luo, and W. Hu, “A quantum deep convolutional neural network for image recognition” Quantum Sci. Technol. 5 044003 (2020).

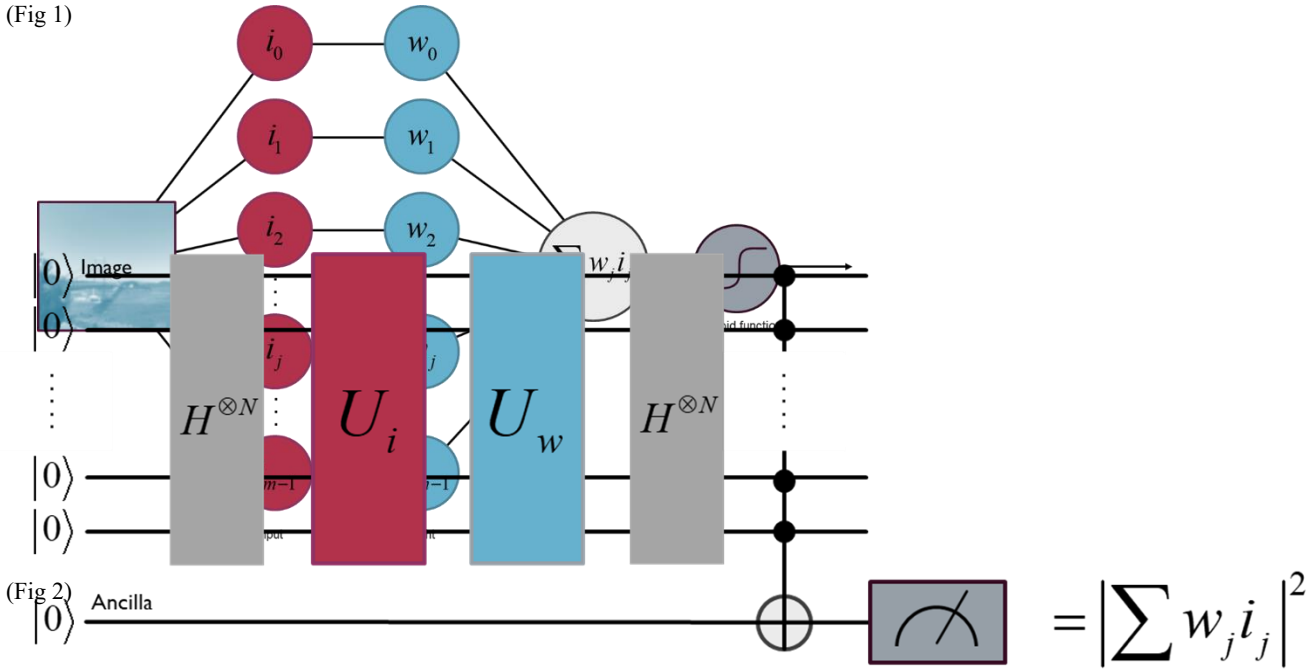
Appendix

The algorithm for automatic generation of quantum circuits is shown. The function itself returns the inner product value.量子回路の自動生成のアルゴリズムを示す。関数自体は内積値を返す。

```
def function_n(n,ki,kw,kw_n):
    #n は使用する qubit 数,ki は入力値,kw は長さγの 1 次元ベクトル,kw_n は層数γ
  #入力値、重みパラメータにはそれぞれ0 ~ 2^(2^N) −1の数値を与えられる。
    #bit は要素数n×2^nの配列
    #gate は要素数n×2^nの配列
    #A は要素数2^n×2^nの配列
    #B は要素数2^nの配列
    #C は要素数2^nの配列
    #D は要素数2^n×kw_nの配列
    #E は要素数2^n×kw_nの配列
    #FLAG は要素数2^nの配列
    #X は要素数nの配列
    for i in range(0,pow(2,n)):
        for j in range(0,n):
            bit[i][n-1-j]=bin4(i,n,j)
    #bit 配列は 0-pow(2,n)の 2 進数表記
      for i in range(0,pow(2,n)):
        for j in range(0,n):
            bit[i][n-1-j]=bin4(i,n,j)

    a=1
    #a は正方行列 A の横の番号.予め全て初期値 1 を入れてある.0 行目は全て初期値の 1.
    for j in range(1,n+1):
        for i in range(0,pow(2,n)):
            if sum(bit[pow(2,n)-1-i])==j:
    #bit 配列を下から、和が 1 からだんだん増やして見ていく.これによってゲートの順になる.
                gate[a]=bit[pow(2,n)-1-i]
    #ついでに gate 配列作成、gate[0]は意味ない
                for k in range(0,pow(2,n)):
                    for l in range (0,n):
                        if((bit[k][l]-bit[pow(2,n)-1-i][l])<0):
    #任意の bit 配列が上の if 文を満たす配列を包含しないなら
                            A[k][a]=0
    #包含しない bit 配列はそのゲートで反転できない
                            break
                a+=1
    #まず縦に|00...0>-|11...1>順番に、横にゲートを順に考えて、それがそのビットを反転させるなら 1。一番左は便宜上で無視。

    for i in range(pow(2,n)):
        C[pow(2,n)-1-i]=1-bin4(ki,pow(2,n),i)
    #反転するなら 1,反転しないなら 0。H ゲート適用後は全ビットが 1 で、0 にするべきビットは 1、そのまま 1 のビットは 0。
    if C[0]==1:
    #C[0](0 だけのビット)は(C^(0-(n-1)))Z ゲートでは反転できないから、他のビットと入れ替えるしかない。
        for i in range (1,n+1): #bit 配列の和が 1 から n の間で
            for j in range(pow(2,n)):
                if sum(bit[pow(2,n)-1-j])==i and C[pow(2,n)-1-j]==0:
    #下から順に考えて bit 配列の和が i と一致してそのビットを反転させなく良いとき
                    for k in range(pow(2,n)):
                        if FLAG[decimal(n,bit[k])]==0:
    #まだ交換がなされてないとき
                            C[decimal(n,bit[k])],C[decimal(n,(bit[pow(2,n)-1-j]-bit[k])%2)]=C[decimal(n,(bit[pow(2,n)-1-j]-bit[k])%2)],C[decimal(n,bit[k])]
    #和が(差でもいいけど)bit[pow(2,n)-1-j]になる組み合わせで交換し合う
```

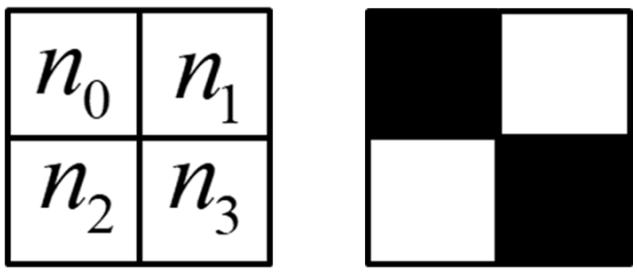
```
                                FLAG[decimal(n,bit[k])],FLAG[decimal(n,(bit[pow(2,n)-1-j]-bit[k])%2)]=1,1
#交換済みのマーカー
                        for k in range(n):
                            X[k]=bit[pow(2,n)-1-j][k]
                        break
                else:
                    continue
            if C[0]==0:
#C の入れ替えが済んだなら終了
                break
B=solve(A,C)
#連立方程式で解を求める。
for i in range(pow(2,n)):
    B[i]=round(B[i])%2
#位相 2 の有限体
for h in range(kw_n):
    for i in range(pow(2,n)):
        D[h][i]=(sum(bit[i])-(1-bin4(kw[h],pow(2,n),pow(2,n)-i-1)))%2
#反転するなら 1,反転しないなら 0
    if(D[h][0]==1):
        for i in range(pow(2,n)):
            D[h][i]=1-D[h][i]
#|00...0>が反転すべきなら、するしないを入れ替える
    E[h]=np.dot(np.linalg.inv(A),D[h])
    for i in range(pow(2,n)):
        E[h][i]=round(E[h][i])%2

q = QuantumRegister(n+1, 'q')
c = ClassicalRegister(1, 'c')
qc = QuantumCircuit(q, c)

for i in range(n):
    qc.h(i)
    qc.barrier(i)
if B[0]==1:
    qc.z(0)
    qc.x(0)
    qc.z(0)
else:
    for i in range(1,pow(2,n)):
        if B[i]==1:
            MCZ(gate[i],sum(gate[i]),n,qc,q)
for i in range(n):
    if X[i]==1:
        qc.x(i)
for h in range(kw_n):
    for i in range(1,pow(2,n)):
        if E[h][i]==1:
            MCZ(gate[i],sum(gate[i]),n,qc,q)
for i in range(n):
    qc.h(i)
MCX(n+1,qc)
qc.measure(q[n],c[0])

shots=1024
simulator = Aer.get_backend('qasm_simulator')
result = execute(qc, backend = simulator, shots=shots).result()
counts = result.get_counts(qc)
num_ones = counts.get('1', 0)/shots

qc.draw('mpl')
return num_ones

#MCX(m,qc)で m 番目の qubit に X ゲートを適用する関数を用意.
#MCZ(bit,m,n,qc,q)で bit[n]の 1 に当たる m 個の qubit 間で CZ ゲートを適用する関数を用意.
```

(Fig 1)



(Fig 2)

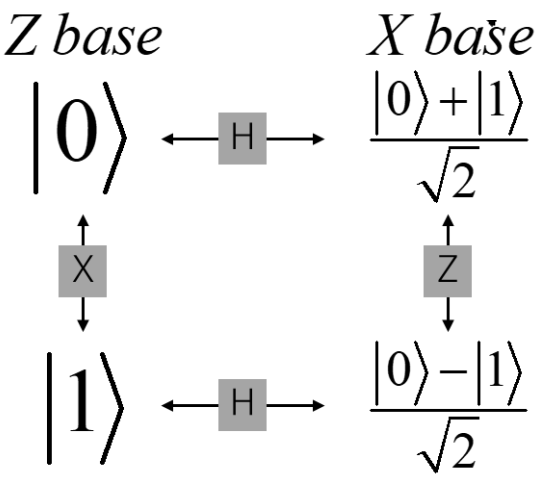
(Fig 3)



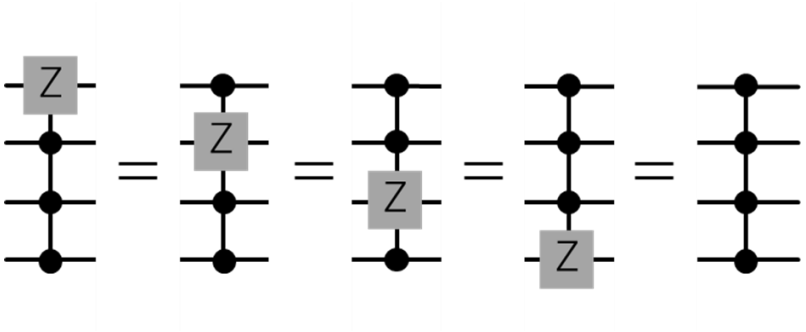
$$k = n_3 n_2 n_1 n_0 \quad ex) k = 6$$

$$black : 0, white : 1$$

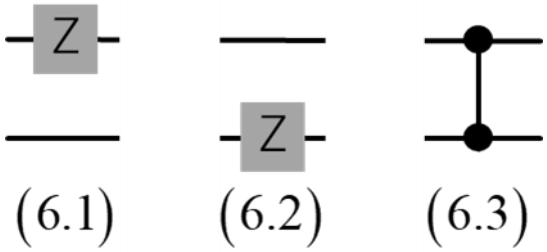
(Fig 4)



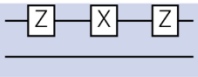
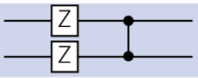
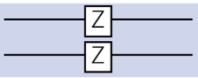
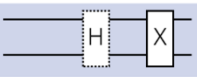
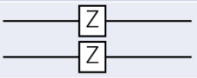
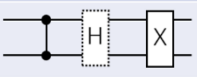
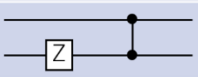
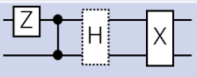
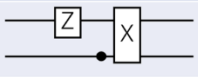
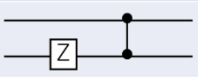
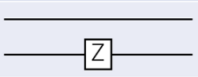
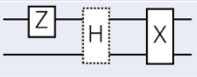
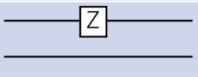
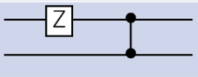
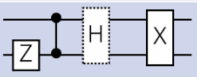
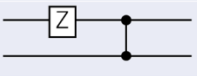
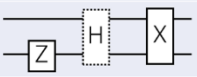
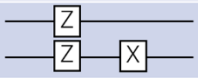
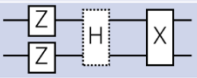
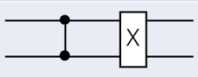
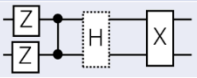
(Fig 5)



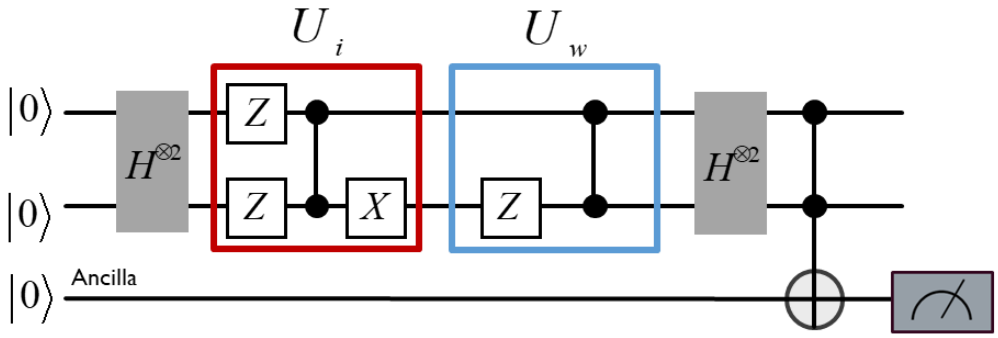
(Fig 6)



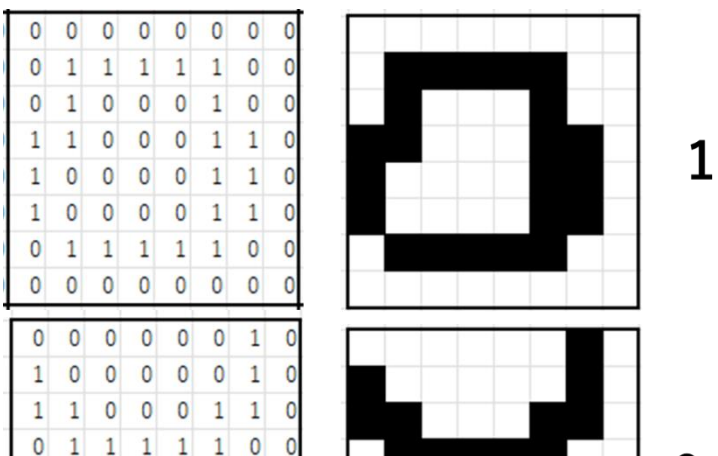
(Fig 7)

k_i	U_i	k_i	U_i	k_w	U_w	(U_w)
0		8		0,15		
1		9		1,14		
2		10		2,13		
3		11		3,12		
4		12		4,11		
5		13		5,10		
6		14		6,9		
7		15		7,8		

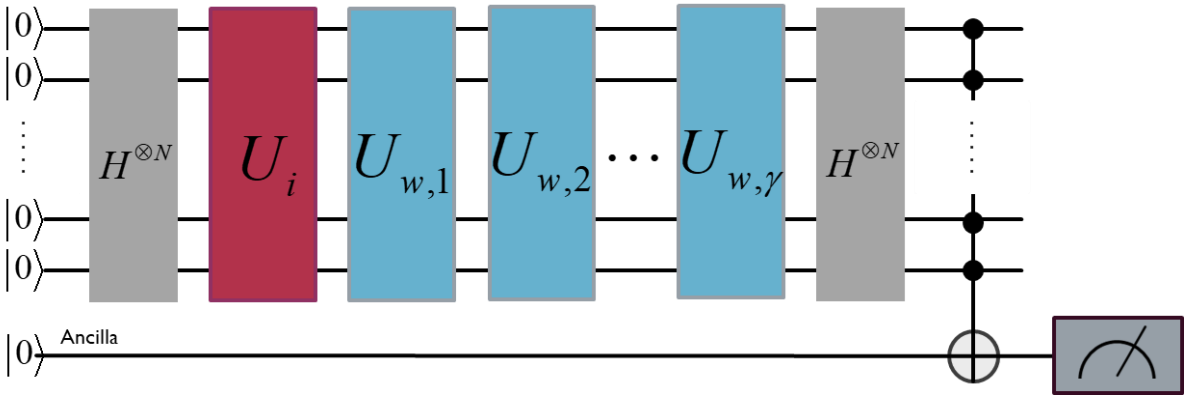
(Fig 8)



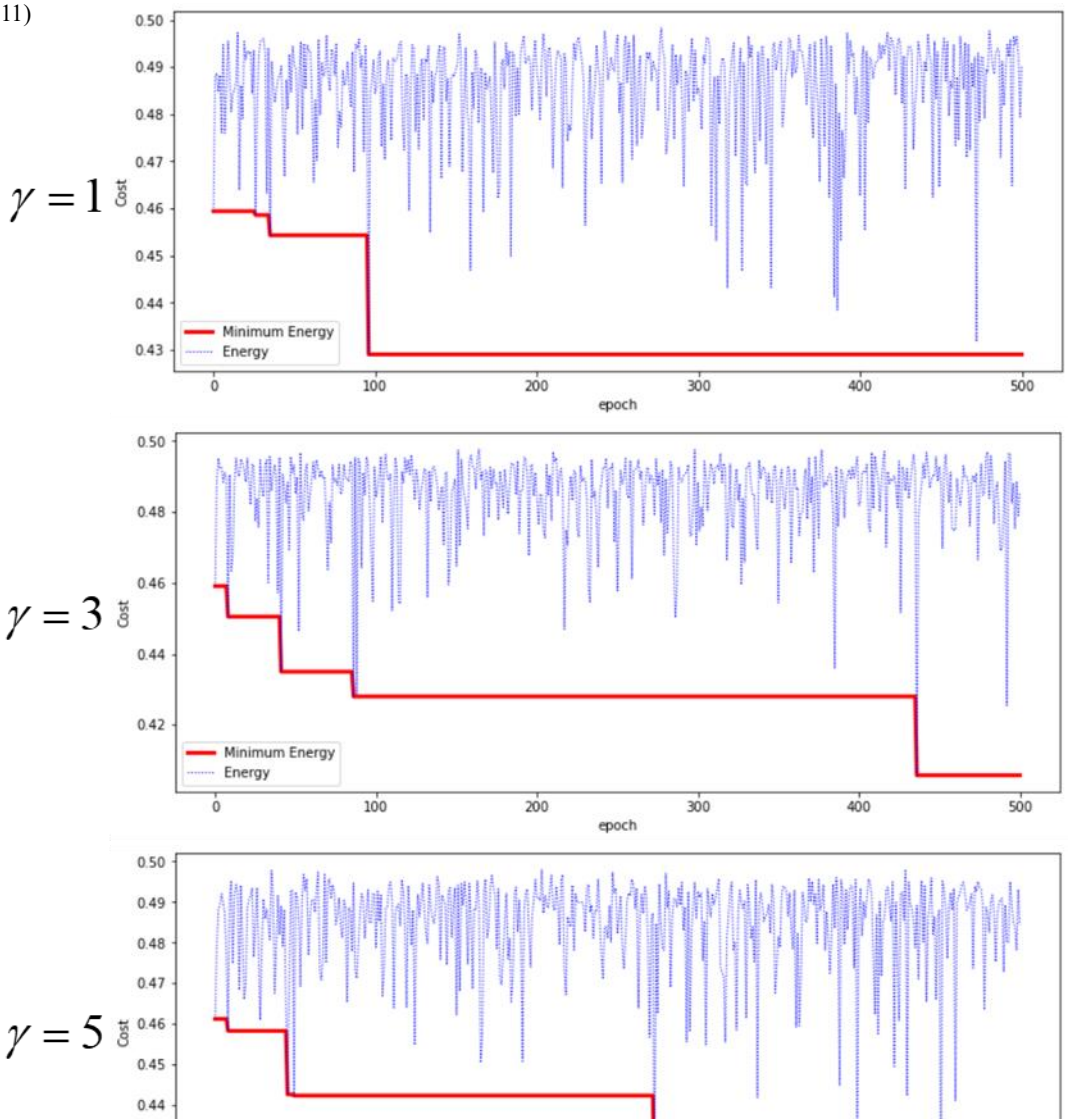
(Fig 9)



(Fig 10)


$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
Ancilla
$H^{\otimes N}$
U_i
$U_{w,1}$
$U_{w,2}$
$U_{w,\gamma}$
$H^{\otimes N}$

(Fig 11)


$\gamma = 1$
$\gamma = 3$
$\gamma = 5$
Cost
epoch
Minimum Energy
Energy
0.50
0.49
0.48
0.47
0.46
0.45
0.44
0.43
0.42
0
100
200
300
400
500

(Fig 12)

	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$
initial value	0.4594	0.4591	0.4612
final value	0.4290	0.4057	0.4182