

修士論文

演算容易性を考慮した CNN における 内部数値表現の最適化

令和 3 年 1 月 28 日提出

指導教員

工藤 知宏 教授

電気系工学専攻 融合情報コース

37-196495 富永 一輝

要旨

近年, CNN のサイズと複雑さはますます増大している. その結果, CNN 演算の計算コストの増大が問題となっている. 特に計算資源の限られたエッジデバイス上で高度な推論を行うため, 認識精度を落とさずに計算コストを低減する手法の研究が多くなされている. これら研究の 1 つの動きとして, FPGA 等のハードウェアの活用が進められている. しかし, これらハードウェアの演算性能, 特に浮動小数点での演算性能を考慮した際に, CNN の推論での演算をいかにシンプルにして認識性能を上げるかが課題となっている.

機械学習モデルの要求するメモリ量や計算コストを低減する手法の 1 つに, CNN の内部数値表現の量子化形式の変更がある. CNN においては 32bit の浮動小数点形式 (FP32) 数値表現が用いられることが多いが, これをより bit 幅の小さな量子化形式に変更することで, メモリ量や計算コストの削減を行う. しかし単純に bit 幅の小さな固定小数点形式や浮動小数点形式への変換を行うだけでは, 推論時の認識精度が落ちてしまう. そのため量子化によってメモリ使用量や計算コストを減らしつつ, 量子化後の CNN の認識精度を維持できる手法について多くの研究がなされてきた.

本論文では, FPGA 等を用いた CNN 推論に適した, メモリフットプリントと計算コスト双方を削減可能な量子化手法について提案した. 具体的には, LQ-Nets 量子化を対象として, 量子化基底の最適化基準に推論精度を用いる手法を提案し, 従来手法との比較・検討を行った. CNN に VGG-like を, データセットに CIFAR-10 を用い, レイヤごとに, 重み w とアクティベーション a の LQ-Nets 量子化を最適化した結果, 量子化誤差が最小となるように最適化する従来手法と比べて約 3% 推論精度が向上した. 提案手法は, 従来手法と比べ量子化形式が大幅に制約されているのにも関わらず, 提案手法を用いて最適化した量子化 CNN の認識精度が向上したことから, CNN の内部数値表現の量子化において, 提案手法での最適化が効果的であることが示された.

目次

第 1 章 Introduction.....	7
1.1 背景	7
1.2 当研究における貢献事項.....	8
1.3 当論文の構成	9
第 2 章 関連研究	10
2.1 畳み込みニューラルネットワーク (CNN)	10
2.2 量子化	12
2.3 Variable-Binsize-Quantization(VBQ).....	13
2.4 Accuracy-based Variable-Binsize-Quantization (AC-based VBQ).....	14
2.5 Learned Quantized Networks (LQ-Nets).....	15
2.5.1 LQ-Nets における代表値の制限.....	16
2.5.2 LQ-Nets における量子化の最適化.....	18
2.6 関連研究のまとめ.....	19
第 3 章 Accuracy-based と Quantize Error Minimization (QEM)の比較.....	20
3.1 背景	20
3.2 実験内容	20
3.3 実験結果と考察	22
第 4 章 Python based AC-based VBQ の作成	23
4.1 背景	23
4.2 実験内容	24
4.2.1 遺伝的アルゴリズム (GA) の Python 実装	24
4.2.2 Python 化した AC-based VBQ の並列処理化	24
4.3 実験結果と考察	25
4.3.1 遺伝的アルゴリズム (GA) の Python 実装の結果と考察	25
4.3.2 Python 化 AC-based VBQ の並列処理化の結果と考察	26
第 5 章 AC-based LQ-Nets.....	29
5.1 背景	29
5.2 提案手法	29

5.2.1 選択	29
5.2.2 突然変異	30
5.2.3 交叉	30
5.3 実験内容	33
5.4 実験結果と考察	34
5.4.1 提案手法と従来手法の認識性能比較.....	34
5.4.2 重み w とアクティベーション a の考察	35
5.4.3 遺伝的アルゴリズムの世代遷移とその考察.....	38
5.4.4 量子化の条件変更.....	38
5.4.5 量子化前後の中間層の出力.....	42
第 6 章 結論と今後の課題.....	45
6.1 結論	45
6.2 今後の課題	46
謝辞	47
参考文献	48
学会発表	52

圖一覽

Figure.1. The structure of the Le-Net, one of the famous example of CNN models	10
Figure.2. The structure of MNIST, one of the famous example of datasets.....	11
Figure.3. An example where an image in MNIS transforms to CNN inputs	11
Figure.4. Various types of simple quantization	12
Figure.5 An example of Various Bin-size Quantization	14
Figure.6 Relationship between the number of bin boundaries and Top-1 Accuracy in some quantization methods.	15
Figure.7 The example of inference calculation.....	16
Figure.8 Representation values in 2-bits Quantization of LQ-Net.....	17
Figure.9 Multiply operation in LQ-Net.....	18
Figure.10 Sum of products in LQ-Net	18
Figure.11 Dataset of CIFAR-10	21
Figure.12 Relationship between the number of bin boundary and Top-1 accuracy	22
Figure.13 The structure of original AC-based VBQ	23
Figure.14 The comparison of original AC-based VBQ and Python AC-based VBQ.....	25
Figure.15 Relationship between the number of process and time in Python AC-based VBQ	27
Figure.16 The distribution of weights and representative values in 2 methods	28
Figure.17 An array P of base-value	30
Figure.18 Mutation method in our method	30
Figure.19 General 2-points crossover.	31
Figure.20 Crossover method in our method.....	31
Figure.21 The structure of GA in our method.....	32
Figure.22 The weight values in AC-based LQ-Nets and Non learning LQ-Nets.....	36
Figure.23 The activation values in AC-based LQ-Nets and Non learning LQ-Nets.....	37
Figure.24 The transition of base values in weight GA.....	40
Figure.25 The transition of base values in activation GA.....	41
Figure.26 The input image of this experiment.	42
Figure.27 The deliberation of activation output of each layer.	44

表一覽

Table.1 Example impacts of bit-width on LQ-Net.....	16
Table.2 Comparison between AC-based VBQ and LQ-Nets	19
Table.3 VGG-like model	21
Table.4 The structure of ABCI (1 node).....	21
Table.5 The conditions in the multiprocessing experiment.....	25
Table.6 The comparison of symmetry and non-symmetry.....	26
Table.7 1 task in Python AC-based VBQ.....	27
Table.8 The conditions in GA	34
Table.9 The comparison of the 3 methods.	34
Table.10 The result of AC-based LQ-Nets.....	35
Table.11 The structure of multiply calculation units.....	39
Table.12 Comparison between 1 Unit 3-bit quantization and FP32 in CNN accuracy.	39
Table.13 Changing bit-width in AC-based LQ-Nets.....	39

第 1 章 Introduction

1.1 背景

近年、ディープニューラルネットワーク(DNN) は様々な用途で実用化されている。特に画像認識の分野では、畳み込みニューラルネットワーク(CNN) と呼ばれるモデルの認識性能が年々向上している。しかしながら、認識性能の向上と共に CNN のモデルサイズと複雑さが増し、CNN の計算コストが高くなってきている。大規模画像認識のコンクールである ImageNet large scale visual recognition challenge(ILSVRC)においては、2012 年に CNN の一種である AlexNet [1]が優勝して以来、ほぼ全てのシステムが CNN ベースに置き換わっている。AlexNet は畳み込み層 5 層、全結合層 3 層の合計 8 層という構成であったが、2014 年度に優勝した Google-Net[3]は 22 層、2017 年に優勝した SENet[4]は 154 層と、年々 CNN の層が深まり、大規模なモデルとなっている。大規模なモデルは、限られたハイエンドサーバーのみでしか実行できず、計算コストがかかる[5, 6, 7]。大規模モデルの一例として ResNet-50 と呼ばれる CNN があるが、これは GPU を 8 台(Tesla-P100) を導入しても、ImageNet と呼ばれるデータセットを学習し終えるまでに約 29 時間要する[8]。また三上氏らが行った研究においては、前例と同様の学習を 224 秒で終えることに成功しているが、最大 2176 台もの GPU TeslaV100) が使用されており、計算コストが高くなっている[7]。CNN の推論時にかかる計算コストは、学習にかかるコストの約 3 分の 1 であるものの、推論は学習時と比較してより少ないリソース下で実行されるケースが多く、推論においても CNN の計算コスト削減が求められている。特に、計算資源の限られたエッジデバイス上で高度な推論を行うため、認識精度を落とさずにモデルを減量化する手法について、様々な提案がなされている。その中の 1 つの動きとして、FPGA 等のハードウェアを使用して、CNN の演算における計算コストを削減する動きが世の中に存在している[9,10]。FPGA を用いることで、低消費電力・低レイテンシな推論を実現できる。しかしながら、FPGA 等のハードウェアは、CNN の演算で使用される浮動小数点演算に難点を持っている。このようなハードウェアを考慮した際、どのように CNN での演算を単純化し、かつモデルの認識性能を向上させるかが課題となっている。CNN の演算における計算コスト削減のために、学習済みの CNN のサイズを縮小することに焦点を当てた研究が多く存在している[20,21]。このためのモデルの圧縮技術の 1 つとして、CNN 内部数値表現の量子化形式を変更する技術が存在している[11]。量子化

によって、CNN の演算に使われる数値表現の bit 幅を減らすことで、CNN の演算に必要なメモリ量・計算コスト・通信量を削減できる。CNN 内部数値表現の量子化形式を変更する手法は、大きく 2 つの手法に分けることができる。1 つは CNN の学習を行いながら量子化する方法で、もう 1 つは学習済みの CNN の数値表現を量子化する方法である。前者の方法では、CNN 数値表現の量子化形式の変更後にモデルの再学習が行われる[13, 14, 15, 16, 17]。前者の手法を用いることで、量子化前的小数点 32 ビット(FP32)CNN での認識性能をある程度維持して量子化することが可能になる[19]。しかしながら、学習済みの CNN を量子化する際は再学習を必要とする。その上、CNN 学習時のパラメータを調整する作業には多くの労力と時間を必要とする。一方で後者の方法においては、既に学習を終えた CNN を使用し、このモデルの内部数値表現を量子化する[23, 24, 25]。この手法では、学習なしで量子化 CNN を作成することができるため、モデルの再学習を必要としない。しかし、量子化の際に量子化誤差が蓄積する可能性があり、CNN の認識性能を維持するのは比較的難しいとされている。

当論文では、後者の量子化手法に注目し、FPGA 等のハードウェアを用いた推論を考慮した、新たな量子化 CNN 作成手法を提案する。

1.2 当研究における貢献事項

当研究において貢献した事項は、以下のとおりである。

- 量子化 CNN の認識精度を維持する上で、当研究室で研究されてきた AC-based VBQ で用いられている AC ベースの最適化(後述)が、従来手法の多くで使用されている QE ベースの最適化より効果があることを実証した。
- AC-based VBQ のプログラムコードを、一律 Python のコードに書き換えた。これにより AC-based VBQ による量子化の最適化を他のコードに移植・適用することを容易にした。
- AC-based LQ-Nets と呼ばれる新手法を提案した。当手法は、先行研究である AC-based VBQ と LQ-Nets における量子化の最適化手法を組み合わせたものであり、双方の長所が生かされた手法となっている。当手法によって、CNN での推論演算をよりシンプルにし、かつ量子化 CNN の認識性能を維持することが可能になった。
- AC-based LQ-Nets を使用することで、Weight と Activation がそれぞれ 3bit に量子化された CNN において、元の CNN の認識性能の低下を 1%未満に抑えることに成功した。
- 先行研究である LQ-Nets や Non-Learning LQ-Nets で作成した量子化 CNN と比較して、

AC-based LQ-Nets で作成した量子化 CNN がより良い認識性能を維持することを確認した.

- 量子化 CNN の認識性能を維持するという目的において, 量子化の最適化の際, **Weight** の代表値を, 量子化誤差を最小にするように最適化するよりも大きい値にしたほうが効果的であることを確認した. 加えて, **Activation** の代表値の大小と認識精度の相関があまり認められないことを確認した.

1.3 当論文の構成

当論文は, 以下で説明するような構成となっている. 第 2 章では, 当研究についての関連研究について記載する. 第 3 章では, 第 2 章で述べる AC-based での最適化が量子化 CNN の認識性能を維持する上で機能するかどうかについて, 確認実験を行う. 第 4 章では, 先行研究である AC-based VBQ を一体化 Python にコードを書き直し, 先行研究と性能を比較する実験を行う. 第 5 章では, AC-based LQ-Nets と呼ばれる提案手法を実装し, 実験の結果を記載する. これと共に実験から得られた知見について述べる. 第 6 章において結論を述べ, 締めくくる.

第 2 章 関連研究

2.1 畳み込みニューラルネットワーク (CNN)

当論文では、畳み込みニューラルネットワーク(CNN) モデルの量子化とその最適化に焦点を当てた研究についての成果をまとめている。この項では、CNN の概要について説明する。畳み込みニューラルネットワーク(CNN)は、畳み込み層と呼ばれる、画像の濃淡パターンを検出する役割を持つレイヤを含んだ、ニューラルネットワークモデルの 1 つである。他のニューラルネットワークモデルと比較して、画像認識に強みを持つのが CNN の特徴である。CNN には様々な種類が存在しているが、最も有名な CNN の 1 つは、Y. L. Cun らによって提案された Le-Net [26] である。Le-Net の構造は Figure.1 の通りである [26]。

CNN において、入力から出力にかけて各レイヤを伝播していく値は、アクティベーションと呼ばれる。畳み込み層や全結合層のアクティベーションを乗算するパラメータは、重みと呼ばれる。画像を入力として受け取り、被写体は何であるかを予測し、それを確率として出力する操作を推論と呼ぶ。また推論の結果を正解と比較し、CNN の重みを更新することで、よりよいモデルを作成していく行程は、学習と呼ばれる。CNN の入力は、通常データセットから入力される。データセットは、CNN を学習するために設けられた画像のセットであり、多くの種類のデータセットが存在している。

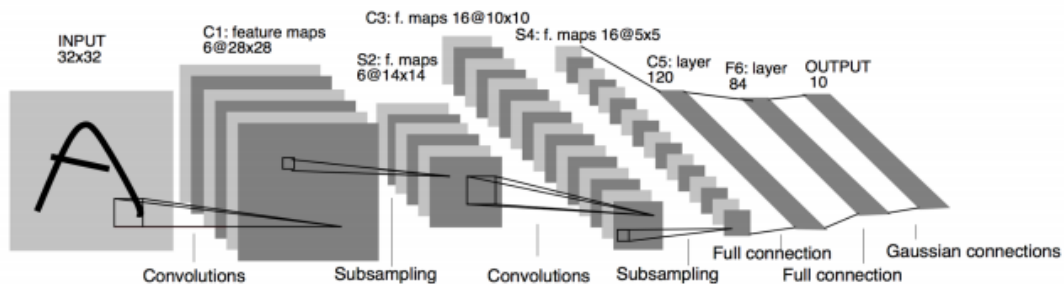


Figure.1. The structure of the Le-Net, one of the famous example of CNN models [26]

有名なデータセットとして、Figure.2 に図示する MNIST[27] がある。MNIST は、0 から 9 までの 10 クラスの手書き番号の画像を集めて作られたデータセットである。すべての画

像はグレースケールであり、サイズは $28 \times 28 \text{px}$ に統一されている。クラス数が少なく、グレースケール画像であり、画像サイズも比較的小さいため、MNIST は最も単純なデータセットの 1 つとされている。

CNN の推論・学習においては、Figure.3 のように、データセット内の画像の画素値が CNN の入力として使用される。学習においては、入力された画像による出力と、データセットの画像 1 枚ごとに紐づけられている正解ラベルとを比較し、CNN の重み w が更新されていく。



Figure.2. The structure of MNIST, one of the famous example of datasets[49]

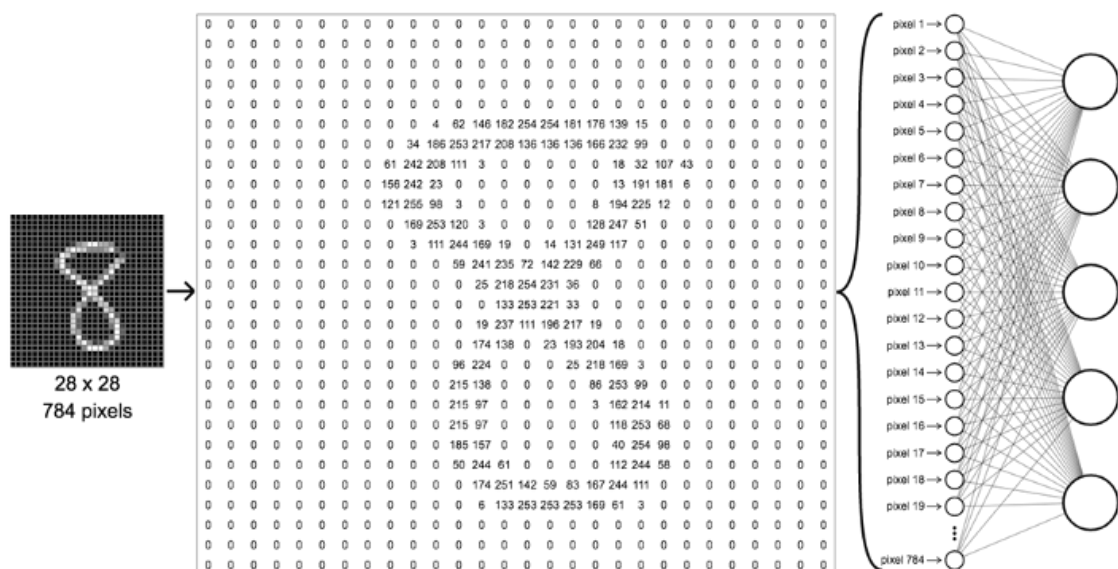


Figure.3. An example where an image in MNIS transforms to CNN inputs [48]

2.2 量子化

量子化とは、任意の範囲にある値全てを 1 つの代表値に置き換える操作である。具体的には、Figure.4[28] のように、数直線上に複数の閾値 (Bin boundaries) を設定し、2 つの異なる閾値の範囲中にある値を 1 つの代表値に置き換える操作のことを量子化という。当論文においては、閾値の範囲のことをビン、範囲の幅のことをビンサイズ、閾値のことをビン境界と表す。

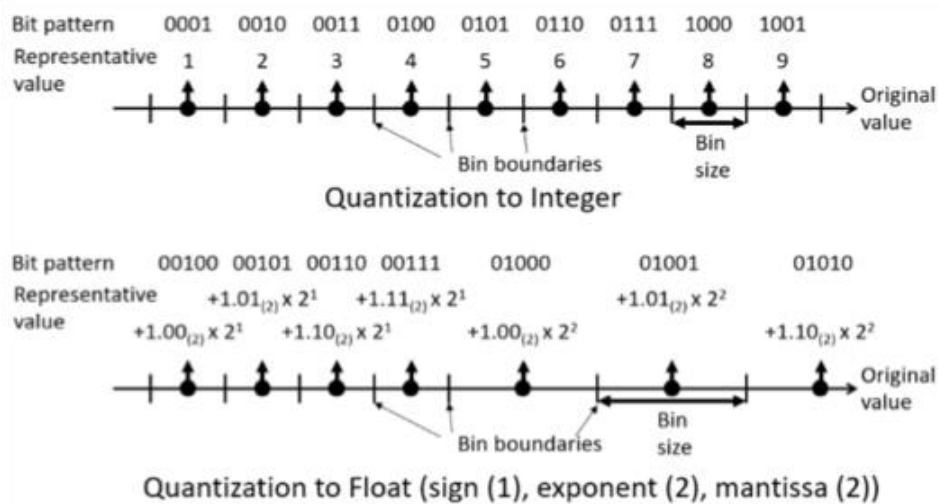


Figure.4. Various types of simple quantization [28]

コンピュータにおいて実際に使用されている数値は、Figure.4 の数直線のような連続的な値ではなく、断続的な値である。コンピュータは 2 ビットの組み合わせで表現できる式しか扱えない為、Figure.4 に示すような、ビン境界を一定の間隔で決める線形量子化や Log スケールごとにビン境界を配置する対数量子化、あるいはその組み合わせによって断続的になった数値を使用している。量子化が施される前の CNN において、実際は重み w やアクティベーション a 等の数値表現は既に量子化された状態になっており、これらは主に 32bit (FP32) で表されている。従って、CNN の量子化では、主に 32bit の数値表現で構成された CNN の量子化形式を変更し、更に小さい bit 幅に量子化する操作が行われる。CNN の量子化により、CNN のサイズと計算コストを縮小できるが、CNN の内部数値表現が限定される。量子化形式の bit 幅が小さくなるにつれて、量子化前後の数値表現の間での誤差が大きくなり、これが量子化後の CNN における画像認識性能を劣化させる可能性がある。

CNN の量子化には主に 2 種類の手法が存在している。1 種類目が CNN の学習時に量子化を伴う方法、2 種類目が学習済みの CNN を量子化する方法である。前者の手法について、

10 クラス分類を行うデータセットを用いて、量子化 CNN を作成した先行研究が存在している[13,24,29]. これらの研究においては、推論時の認識性能を維持しつつ、重みとアクティベーションを2値化することに成功している. しかしながら前者の方法は量子化 CNN の作成に学習を要するため、学習に関わるパラメータの設定等、多くの労力を要する. 一方で後者の手法では、学習済みの CNN を、様々なデータセットを用いて量子化する先行研究が存在している[23,24,25]. これらの研究では、コンピュータの数値表現で用いられている線形量子化や対数量子化等を用いて CNN の内部数値表現の量子化形式の変更を行い、量子化 CNN の性能を測定している. しかし、このようなビン境界を一定の規則で決める、単純な量子化では、量子化後の CNN の認識性能を維持するためにあまり効果がなかったことが示されている. 従って、量子化により計算コスト削減とメモリ使用量削減を実現しつつ、量子化 CNN の性能を維持できる不規則な量子化について、多くの研究がされてきた[20, 30, 31].

2.3 Variable-Binsize-Quantization(VBQ)

前述した不規則な量子化の 1 つの例が、Figure.5 [28] で示される Variable-Bin-size-Quantization (VBQ) と呼ばれる、学習済みの CNN に量子化を施す手法である. VBQ は、線形量子化や対数量子化等の単純な規則の量子化とは異なり、閾値の配置箇所に関する規則を定めず、量子化したい CNN に応じてビンを自由に決定することにより、量子化 CNN の性能を維持することを試みた手法である. ビンの数を 2^n で表せる数に限定することで、学習済み CNN に n bit の量子化を施すことができる.

VBQ には主に 2 つの問題が存在している. 1 つ目の問題は、ビン境界の決定方法に関する問題である. ビン境界の決定方法として、ビン境界を量子化前後の数値表現の誤差が最小になるように決定する方法 (Quantized Error Method, QEM) を使用するのが自然である. しかし、この QEM が、量子化 CNN の認識性能を維持するのに最適な手法であるかは定かではない. 2 つ目の問題は、量子化後の演算方法に関する問題である. VBQ は、量子化 CNN を推論する際の計算方法まで考慮しきれていない. 従って VBQ を用いた量子化 CNN は、FPGA のような浮動小数点計算に弱みを持つハードウェアでは、CNN の推論において効率的な計算を実行できない.

以下で紹介する 2 つの手法は、前述した VBQ の 2 つの問題のそれぞれに対処する手法である.

- Accuracy-based Variable-Bin-size-Quantization (AC-based VBQ)
- LQ-Nets

次項において、2 つの手法について詳細を記述する.

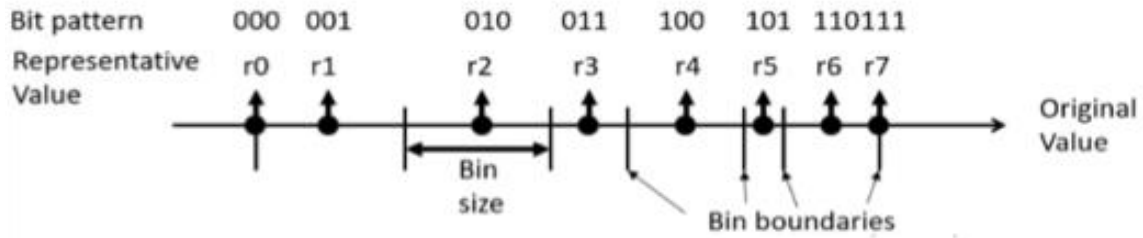


Figure.5 An example of Various Bin-size Quantization [28]

2.4 Accuracy-based Variable-Binsize-Quantization (AC-based VBQ)

Accuracy-based Variable-Bin size-Quantization (AC-based VBQ) [32] は、当研究室において 2019 年に発表された技術である。AC-based VBQ は、学習済み CNN の重み w を、レイヤ毎に量子化できる。当技術では、CNN を量子化する際、量子化後の CNN の認識精度 (Accuracy) が最良になるようにビン境界・ビンサイズを決定できる。Quantized Error Method (QEM) とは違い、量子化 CNN の性能を確実に向上させるように、量子化自体を最適化できる点が当技術の最も重要な特徴である。当手法における最適化では、全ての教師データに対して推論を実行した上で、最も適したビン境界の配置を決定している。しかし、ビン境界のセットが不連続な値であるため、勾配法をはじめとする一般的な最適化手法を用いることが難しく、この最適化をどのように行うのかは自明ではない。具体例として、AC-based VBQ を用いて、学習済み CNN に 3bit 精度での量子化とその最適化を施すことを考える。この際、ビンの数を $2^3 = 8$ 個に対応させるため、7つのビン境界を決定する必要がある。この7つのビン境界は実数値全体の値を取りうる上、断続的な値を取るため、どのような手法を用いて最適化するのかについては自明ではない。

そこで、AC-based VBQ では、ビン境界の最適化において、遺伝的アルゴリズム (Genetic-Algorithm, GA) [34] が使用されている。遺伝的アルゴリズムとは、解の候補を遺伝子と見立て、様々な遺伝子を持つ複数の個体に対し、選択・突然変異・交叉などの操作を繰り返すことで、優れた遺伝子を次世代へ継承するアルゴリズムである。

AC-based VBQ における 1つの遺伝子をビン境界の集合 $\{V_1, V_2, \dots\}$ とする。2つのビン境界に挟まれた区間の値は、2つのビン境界の平均値を代表値とし、ビン境界に挟まれない区間については、最も近いビン境界の値を代表値とすることで、全ての实数範囲をいずれかの代表値に対応させる事ができる。複数存在する遺伝子の候補に遺伝的アルゴリズムを施し、量子化 CNN の認識精度が最も良くなる遺伝子を探索する。AC-based VBQ では、重み w の量子化の最適化を CNN のレイヤ毎に行っている。

Figure.6 は、AC-based VBQ の成果の一例である。AC-based VBQ (Figure.6 中の青線部) を施した量子化 CNN が、単純な量子化を施した量子化 CNN よりも、認識性能が良くなることが示されている。特にビン境界が 7 つ(3bit) 以下の時、単純な量子化と AC-based VBQ で性能差が大きく出ているのが分かる。このように、AC-based VBQ は推論精度を維持したまま量子化表現のビット数を減らすことができ、モデルサイズの削減に成功している。しかし VBQ と同様、現状では推論に FP32 演算を用いており、計算コストは削減できていない。

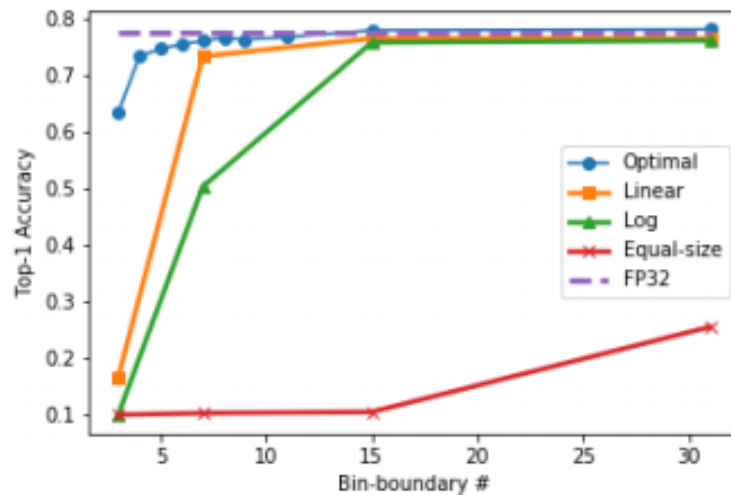


Figure.6 Relationship between the number of bin boundaries and Top-1 Accuracy in some quantization methods. [32]

2.5 Learned Quantized Networks (LQ-Nets)

Learned Quantized Networks (LQ-Nets) [19]は、2018 年に Microsoft 社によって発表された技術である。当技術は、CNN の学習を行いながら、重みとアクティベーションの数値表現を量子化する技術である。LQ-Nets において、これらの数値表現はレイヤよりも細かい括りであるチャンネル毎に量子化される。また量子化後の CNN の認識性能を維持するため、第 1 レイヤに存在する数値表現は量子化されていない。Table.1 は LQ-Nets の性能の一例を示したものである。

LQ-Nets の最も重要な特徴は、LQ-Nets 特有の量子化形式にある。LQ-Nets での量子化形式を適用することで、量子化 CNN の推論演算を重みとアクティベーションが量子化された状態のまま行うことができる。この特徴によって、LQ-Nets で作成した量子化 CNN は、推論時にシンプルな演算が可能になる。そのため LQ-Nets は、浮動小数点演算に弱みを持つ FPGA 等でのハードウェアでの推論に適した量子化モデルを作成できるといえる。LQ-Nets における代表値の制限について説明する。

Table.1 Example impacts of bit-width on LQ-Net.[19]

ResNet-20 (CIFAR-10)		VGG-Small (CIFAR-10)		ResNet-18 (ImageNet)	
Bit-width (W/A)	Acc. (%)	Bit-width (W/A)	Acc. (%)	Bit-width (W/A)	Acc. (%)
32/32	92.1	32/32	93.8	32/32	70.3
1/32	90.1	1/32	93.5	2/32	68.0
2/32	91.8	2/32	93.8	3/32	69.3
3/32	92.0	3/32	93.8	4/32	70.0
1/2	88.4	1/2	93.4	1/2	62.6
2/2	90.2	2/2	93.5	2/2	64.9
2/3	91.1	2/3	93.8	3/3	68.2
3/3	91.6	3/3	93.8	4/4	69.3

2.5.1 LQ-Nets における代表値の制限

LQ-Nets の量子化形式では、代表値にある種の制限を加えている。代表値に制限を加えた量子化 CNN を作成することで、量子化 CNN の推論時に、量子化された代表値を用いたシンプルな演算が可能になる。当項では、LQ-Nets における代表値の制限と推論時の演算について説明する。

初めに簡単のため、簡素なニューラルネットワークモデルにおける推論演算の仕組みについて説明する。推論を実行する際、学習時と同様に、推論したい画像の入力が第 1 レイヤに与えられ、Figure.7 に示すような演算が行われる。1 つのノード(Figure.7 中の赤円部)では、ノードの入力の積 (学習された重み w と、前レイヤのアクティベーション a の積) の合計が計算され、活性化関数を用いてアクティベーションを算出する。算出されたアクティベーションは、次のレイヤに存在するノードに渡される。全てのレイヤにおいて、このような重み w とアクティベーション a の積算が行われ、推論結果が出力される。LQ-Net は、この積算に注目し、代表値に制限を加えた量子化を施している。

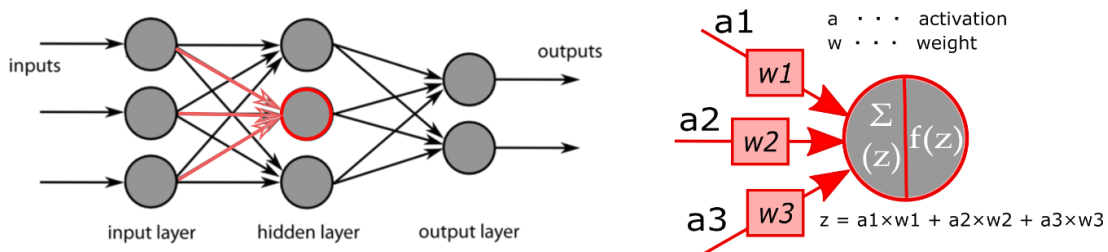


Figure.7 The example of inference calculation

LQ-Nets では、量子化の過程で基底値と呼ばれる要素が各ビットに配当される。基底値は、

量子化後の代表値を加減算のみで表現するために用いるパラメータであり、量子化後の代表値を制限するために使用される。n ビットの量子化を施す際、基底値は全部で n 個の要素を持つ。

具体的な例として、重み w とアクティベーション a の双方をそれぞれ 2 ビットに量子化する例を考える。2 ビットの量子化では、1 ビット目に V_1 、2 ビット目に V_2 という値を割り当てる。この際各ビットは、Figure.8 で示すように、 V_1 と V_2 に正または負の符号を追加するために使用される。重み w の基底値を $\{V_1^w, V_2^w\}$ 、アクティベーション a の基底値を $\{V_1^a, V_2^a\}$ とすると、式 (A) のように、双方の数値表現はそれぞれ 4 種類の代表値のいずれかに量子化される。

$$w = [-V_1^w - V_2^w, V_1^w - V_2^w, -V_1^w + V_2^w, V_1^w + V_2^w] \quad \dots \quad (A)$$

$$a = [-V_1^a - V_2^a, V_1^a - V_2^a, -V_1^a + V_2^a, V_1^a + V_2^a]$$

前述したように、CNN の推論では、重み w とアクティベーション a の積算が実行される。双方の数値表現が式 (A) で示すいずれかの値に量子化されている時、これらの積算の結果は Figure.9 で示す 16 種類のいずれかで表せる。ここで再度積算に注目すると、当積算の結果は Figure.9 中に存在する 4 つの青の値 ($V_1^a V_1^w$, $V_1^a V_2^w$, $V_2^a V_1^w$, $V_2^a V_2^w$) を加減算することによって計算できる。従って、Figure.9 中に存在する 4 つの青の値 ($V_1^a V_1^w$, $V_1^a V_2^w$, $V_2^a V_1^w$, $V_2^a V_2^w$) を事前に計算しておき、テーブルとして保持しておくことで、Figure.10 で示すように、1 つのノードの入力の積の合計を加減算のみで算出できるようになる。

このように、LQ-Nets の代表値の制限が、量子化によるメモリフットプリントの削減を行い、低計算コストかつシンプルな加減算とビット演算を用いた推論を可能にしている。従って、LQ-Nets の量子化方式は、FPGA 等のハードウェア上での推論に適した手法であるといえる。

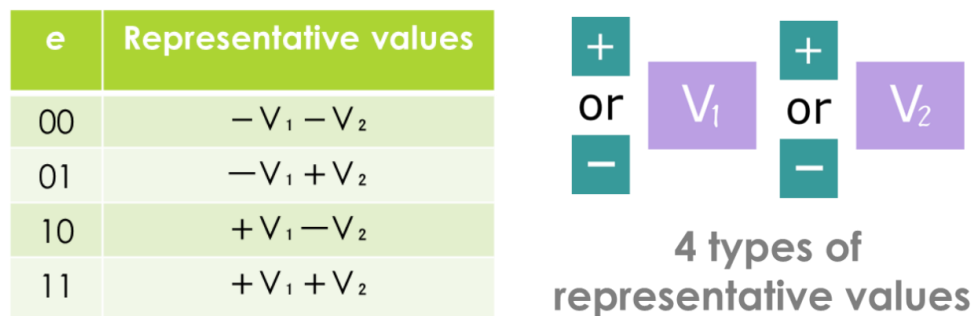


Figure.8 Representation values in 2-bits Quantization of LQ-Net

$$\begin{pmatrix} \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_1^w & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_2^w \end{pmatrix} \times \begin{pmatrix} \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_1^a & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_2^a \end{pmatrix} \\
 = & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_1^a V_1^w & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_1^a V_2^w & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_2^a V_1^w & \begin{matrix} + \\ \text{or} \\ - \end{matrix} V_2^a V_2^w
 \end{pmatrix}$$

Figure.9 Multiply operation in LQ-Net

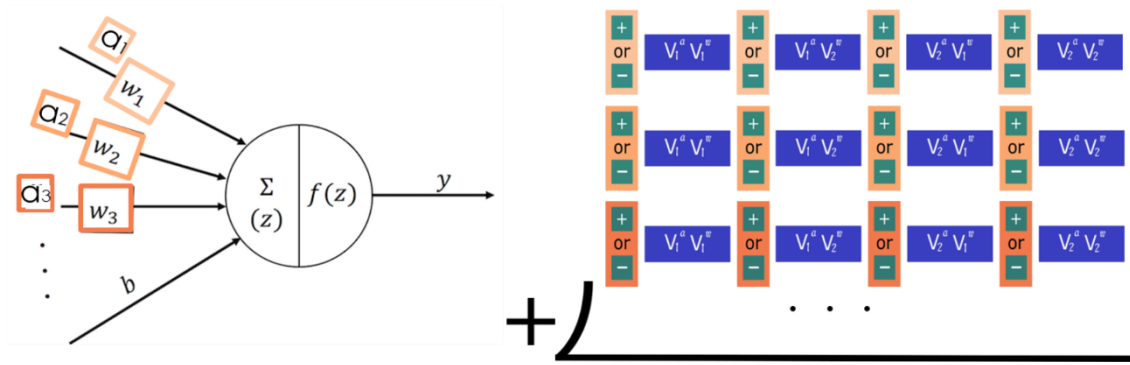


Figure.10 Sum of products in LQ-Net

2.5.2 LQ-Nets における量子化の最適化

LQ-Nets は前項で説明した特徴に加え、基底値の最適化が行われている。基底値は、量子化前後の CNN の数値表現の差分が最小になるように量子化する手法(QEM) を用いて最適化される。LQ-Nets では式(B)で示す最小二乗法の式を用いて、重み w とアクティベーション a の値の最適化が行われる。

$$Q^*(x) = \operatorname{argmin}_Q \int p(x)(Q(x) - (x))^2 dx \quad (\text{B})$$

式(B)によって、量子化 CNN の数値表現の最適化は簡単に行うことができるものの、AC-

based VBQ のように, CNN の認識精度を基準とした最適化手法は用いていない. 量子化誤差を最小にすることが, 量子化 CNN の認識精度を最良にするかどうかは明らかではなく, 当手法での最適化が最も適した手法であるかどうかは定かではない.

2.6 関連研究のまとめ

この章では, CNN と CNN の量子化, VBQ, AC-based VBQ, LQ-Net について述べた. AC-based VBQ と LQ-Nets について, Table.2 にまとめる.

Table.2 Comparison between AC-based VBQ and LQ-Nets

	AC-based VBQ	LQ-Nets
Retraining	No	Yes
Optimization criteria	Accuracy-based	Quantization Error-based
Arithmetic operation	No	Yes

AC-based VBQ は遺伝的アルゴリズム(GA)を用いることによって量子化の最適化を行い, モデルサイズの削減を行いつつ, 特に低 bit の量子化表現を用いた量子化 CNN のパフォーマンスを向上させる事ができる技術であるものの, CNN の推論において現状 FP32 での演算を用いているため, 計算量の削減は行えていない. 一方 LQ-Nets は, 基底値という概念を導入することで, 量子化 CNN の推論を簡単にする表現値へ, 量子化を最適化することができ, 使用メモリ量・計算量の双方で望ましい性質を持っている. しかし量子化の最適化には Quantization Error Minimization (QEM) を用いており, これが量子化 CNN の認識精度を向上するために最適な手法であるかは定かではない. これら 2 つの技術は, どちらも VBQ の課題のうち 1 つを対処する技術であるが, 双方に対処できる技術とはなっていない. そこで, これら 2 つの技術を組み合わせることによって, VBQ の双方の課題に対処できる量子化の最適化手法を実現できると考えた.

第3章 Accuracy-based と Quantize Error Minimization (QEM)の比較

3.1 背景

第2章において、LQ-Nets が Quantization Error Method (QEM) を用いて量子化の最適化を行っており、これが量子化 CNN のパフォーマンスを向上する上で最適な手法であるかは定かではないことを説明した。そこでまずは CNN の認識精度を向上させる目的において、CNN の認識精度 (Accuracy) をスコアとする、量子化を Accuracy based で最適化する手法 (AC-based での最適化) に効果があるのかについて明らかにする。そのため、AC ベースでの最適化と、QEM を利用する Quantization Error based の手法 (QE-based での最適化) の2手法の性能差を明らかにするための実験を行った。

3.2 実験内容

VBQ の代表値の最適化を AC-based と QE-based の2通りで実施し、結果を比較した。当実験では学習済み CNN として VGG-like モデルを用い、学習データセットとして CIFAR-10 を用いた。VGG-like モデルは、Table.3 のような畳み込み層 7 層、全結合層 3 層の計 10 層から構成されている。モデルの実装には、Keras と Tensorflow を用いた。学習した VGG-like モデルの認識精度は、CIFAR-10 のテストデータに対して 77.98%であった。この学習済みモデルが本実験の基準となる。CIFAR-10 は、約 6 万枚の画像(訓練用画像 50000 枚、テスト画像 10000 枚)に、10 クラスの解答ラベルを施した、Figure.11 のようなデータセットである。CIFAR-10 を採用した理由は、AC-based での最適化において遺伝的アルゴリズムを繰り返す際、スコアを繰り返し計算する必要があるため、比較的簡単なベンチマークの方が望ましいこと・少しの遺伝子の違いによって適応度に差が出る程度には難しいベンチマークである必要があることの2点である。以後、当実験で使用した、代表値の最適化を QEM によって行った VBQ のことを、QE-based VBQ と呼ぶ。この QE-based VBQ は、Python によって実装した。当実験では、計算を行うハードウェアとして産業総合研究所 (AIST) が運営するスーパーコンピュータシステム ABCI を用いた。Table.4 で示すように、ABCI は1つの計

算ノードに 4 つの GPU を持つハードウェアである.

Table.3 VGG-like model [32]

Layer #	Layer	Param #
Layer1	Conv2D	$3 \times 3 \times 3 \times 64$
Layer2	Conv2D	$3 \times 3 \times 64 \times 64$
MaxPooling		
Layer3	Conv2D	$3 \times 3 \times 64 \times 128$
Layer4	Conv2D	$3 \times 3 \times 128 \times 128$
MaxPooling		
Layer5	Conv2D	$3 \times 3 \times 128 \times 256$
Layer6	Conv2D	$3 \times 3 \times 256 \times 256$
Layer7	Conv2D	$3 \times 3 \times 256 \times 256$
MaxPooling		
Layer8	FC	4096×1024
Layer9	FC	1024×512
Layer10	FC	512×10

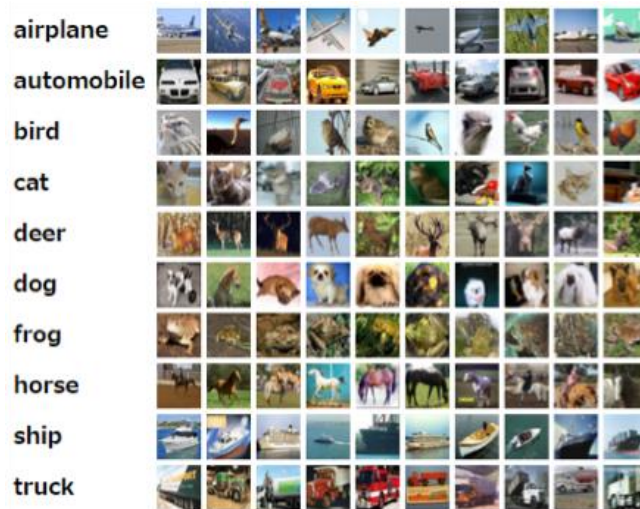


Figure.11 Dataset of CIFAR-10 [50]

Table.4 The structure of ABCI (1 node)

CPU	Intel Xeon Gold 6148(27.5M cache, 2.40GHz, 20Core) $\times 2$
GPU	NVIDIA Tesla V100(SXM2) $\times 4$
Memory	384GiB

3.3 実験結果と考察

量子化の精度, すなわち VBQ の代表値の数を変えながら計算した量子化 CNN の認識精度を Figure.12 に示す. どちらの手法でも, 代表値の数が 8 個以上, すなわち 4 bit 以上であれば, FP32 のモデルと同等の認識精度が得られた. しかし代表値の数が 7 つ以下, すなわち 3bit 以下になると, QE-based の手法において認識精度が下がり始めているのがわかる. また代表値の数が 4 つ以下, すなわち 2bit 以下になると, QE-based VBQ の認識精度は 10% 程度まで低下する. これはランダムで解答を選択した場合と同程度となることから, 推論が機能していないことがわかる. 一方, AC-based VBQ では 2bit の量子化でも認識精度は低下していない. 以上の結果から, 特に低ビットの量子化では AC-based の最適化が優位であると考えられる.

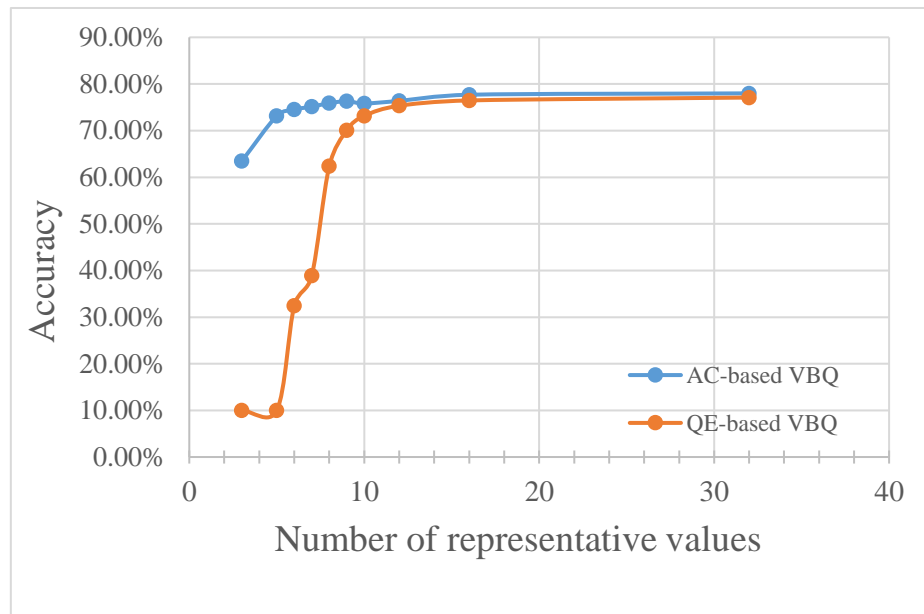


Figure.12 Relationship between the number of bin boundary and Top-1 accuracy

第4章 Python based AC-based VBQの作成

4.1 背景

AC-based VBQ は, Figure.13 のような構成になっている. AC-based VBQ では, 遺伝的アルゴリズム(GA)を行うプログラムと量子化 CNN の評価を行うプログラムが独立しており, 双方のプログラム間の通信手段として gRPC を用いている. しかし AC-based VBQ は, 先述した gRPC のオーバーヘッドが重いこと, C++から不透明なプロトコルを通して Python を立ち上げていたため, バグが不透明であること等の問題を抱えていた. これらの問題に対処するため, AC-based VBQ を一体化の Python で構成し直した. AC-based VBQ での遺伝的アルゴリズム (GA) を一体化 Python で書き直すことにより, C++と Python で書かれた AC-based VBQ を Python のみで構成することができ, 1つの Python で Tensorflow を複数のインスタンスで回すことができる. また複雑な元のプログラムを簡略化する意味や, 今後様々な量子化手法と遺伝的アルゴリズム (GA) を組み合わせる際の利便性を高める意味も持つ.

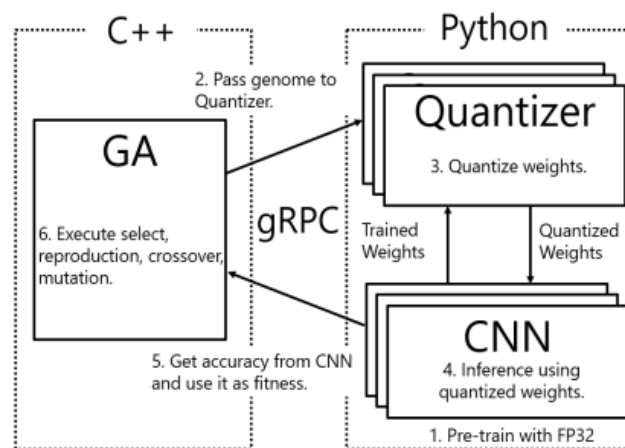


Figure.13 The structure of original AC-based VBQ[32]

4.2 実験内容

4.2.1 遺伝的アルゴリズム (GA) の Python 実装

AC-based VBQ の遺伝的アルゴリズム(GA)のコードを Python で書き直し、オリジナルの AC-based VBQ での結果と比較を行うことで、Python 化した AC-based VBQ での結果がオリジナルの AC-based VBQ での結果と相違なく、同様な手法として扱うことができるか確認をした。学習済み CNN として、第 3 章と同様の条件で学習した VGG-like モデルを使用した。オリジナルの AC-based VBQ (original VBQ) と Python で一体化した AC-based VBQ (Python VBQ)のそれぞれの手法で、各レイヤについて重み w のみを抽出して量子化した。これをモデルに差し戻して、量子化 CNN の認識精度を評価してスコアとした。このスコアに基づいて、VBQ の量子化における代表値を、遺伝的アルゴリズム (GA) に基づいて最適化した。レイヤごとに最適化した量子化形式を結合し、最終的な認識精度を算出し、双方の手法での結果を比較した。

4.2.2 Python 化した AC-based VBQ の並列処理化

AC-based VBQ では、遺伝的アルゴリズム (GA) の 1 世代分の計算を行うために、重み w を量子化する際の計算と、量子化 CNN の認識精度の計算を、1 世代分の遺伝子の数と同じ回数だけ繰り返す必要がある。そのため学習済み CNN のモデルサイズが肥大化したり、遺伝的アルゴリズムでの世代数や遺伝子数が増えたりした際に膨大な演算量となり、多くの時間を要する。そのため、Python 化した AC-based VBQ において、遺伝的アルゴリズム(GA)を並列処理で行えるコードを作成した。

同様の VGG-like モデルを使用して、Python 化した AC-based VBQ を、遺伝的アルゴリズム(GA)の並列プロセス数を変化させながら、量子化の最適化に要する時間を測定した。この際、遺伝的アルゴリズムの世代更新数とビン境界の数をそれぞれ 10, 7 に固定した。また、オリジナルの AC-based VBQ を参考に、ビン境界が取りうる値を、0 を中心に対称になるような値に限定して(Symmetry) 最適化を施した。

当実験では、計算を行うハードウェアとして産業総合研究所 (AIST) が運営するスーパーコンピュータシステム ABCI を用いた。前述したように、ABCI は 1 つの計算ノードに 4 つの GPU を持つハードウェアである。そのため、ABCI において 1 つのプロセスで CNN の演算を実行すると、1 つの GPU しか使用できず非効率的である。また Python 標準ライブラリの Multiprocessing では、単一ノードでの並列処理しか行えず、ABCI を用いた遺伝的アルゴリズムの計算においてプロセス数を 5 つ以上にすることができない。そこで MPI を用いることで、複数のノードを使用したマルチプロセス通信を可能にし、遺伝的アルゴリズム (GA)の並列プロセス化を実現した。当実験では、遺伝的アルゴリズムを 2~4 プロセスで並

列処理する際は、Python 標準ライブラリの Multiprocessing, 5 プロセス以上で並行処理する際は MPI を使用し、理想的な並列処理が行えているかどうかを確認した。

Table.5 The conditions in the multiprocessing experiment

The bit width of activations (bit)	32
The bit width of weights (bit)	3
The number of gene in one generation	50
Total Generation	10
Reproduction (%)	30
Mutation (%)	20
Crossover (%)	50

4.3 実験結果と考察

4.3.1 遺伝的アルゴリズム (GA) の Python 実装の結果と考察

遺伝的アルゴリズム(GA)の Python 実装の結果を Figure.14 に示す。Figure.14 に示すように、計測したビン境界のほとんどで、2 手法の誤差が 1%未満となった。従って双方の手法で性能差がほとんど変わらないということができ、遺伝的アルゴリズム(GA)の Python 実装がうまく機能していると推察できる。

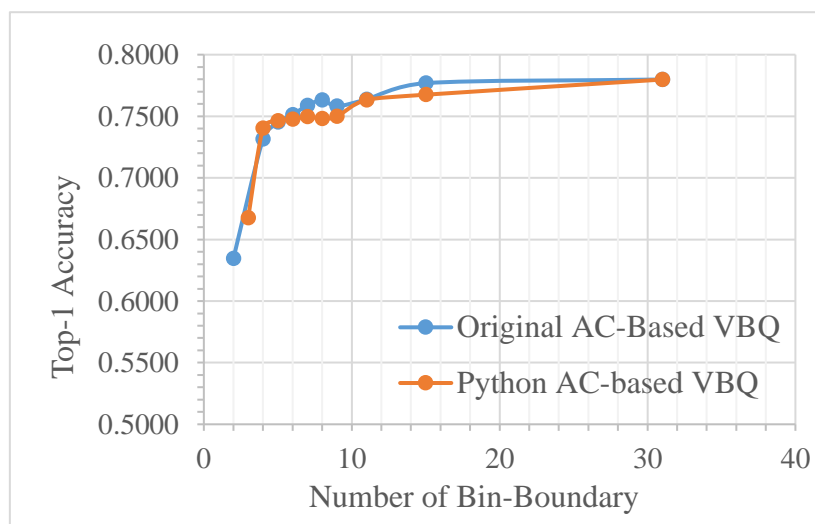


Figure.14 The comparison of original AC-based VBQ and Python AC-based VBQ

Table.6 は、当実験においてビン境界が 7 つ(3bit)の際、0 を中心に対称になるように制限を与えて最適化を施した場合(Symmetry)と、ビン境界を自由に最適化した場合 (non-Symmetry) の認識精度を比較した表である。また Figure.16 は、2 手法で最適化を行った際の代表値と重み w の分布を、モデルのレイヤごとに示した図である。Symmetry の制限を加えた最適化手法の方が、Non-symmetry の手法よりも認識精度が約 4%良くなっている。理論的には、Non-symmetry の手法の方が、Symmetry の手法よりも代表値の取りうる値の種類が多いため、より最適化された代表値で量子化が行えるはずである。しかし Figure.16 で示すように、Non-symmetry の手法では、いくつかの代表値がほぼ同じ代表値に最適化されている Layer が散見される。Symmetry の手法における 1 つの遺伝子の長さが、Non-symmetry の手法における 1 つの遺伝子の長さの半分であることを踏まえると、Non-symmetry の手法では、遺伝的アルゴリズムの最適化がまだ不完全であり、うまく収束しなかったため、Figure.16 のような結果になったと推察できる。以上より、Symmetry の手法が遺伝的アルゴリズムの収束を早めるのに効果があると推察できる。

当実験では遺伝的アルゴリズムの世代更新数を 200 としたが、Non-Symmetry の手法においては最適化が不完全であると推察できることから、世代更新数を増やしたり、遺伝的アルゴリズムの初期パラメータを変更したりすることで、Non-Symmetry において更により認識精度が得られることが期待できる。遺伝的アルゴリズム(GA)はまだまだ改善の余地があると考えられる。

Table.6 The comparison of symmetry and non-symmetry

Method	Accuracy (%)
Symmetry	74.96
Non-Symmetry	71.29

4.3.2 Python 化 AC-based VBQ の並列処理化の結果と考察

Python 化した AC-based VBQ を並列処理化した際の、プロセス数と処理時間の関係を Figure.15 に示す。当実験において、プロセス数と処理時間が反比例するのが理想的な出力である。プロセス数によって 3 つの異なる手法を使い分けているのにも関わらず、測定値が理想的な出力に近い出力となっていることから、Multiprocessing や MPI を用いた Python 化 AC-based VBQ での並列処理が機能していることが分かる。

Table.7 は Python 化した AC-based VBQ における処理の一覧と、所要時間をまとめた表である。遺伝的アルゴリズム(GA)の部分については、遺伝的アルゴリズムでの世代数や遺伝子数によって処理時間が変化することに注意したい。Table.7 より、Python 化した AC-based VBQ では、GPU を各プロセスに割り振る処理と、遺伝的アルゴリズム(GA)の処理の 2 箇所

で多くの処理時間を必要とすることが分かる。GPU を各プロセスに割り振る処理については、オリジナルの AC-based VBQ を一体化 Python で書き直したことで、遺伝的アルゴリズム(GA)を行う毎に行っていた当処理が 1 回で済むようになり、プログラム効率が向上した。また遺伝的アルゴリズム(GA)の処理においては、MPI を用いた並列処理を組み込むことで、処理時間の短縮が可能になった。従って、オリジナル AC-based VBQ を一体化 Python で書き直し、マルチプロセス化する作業は、プログラムの効率を向上させているといえる。

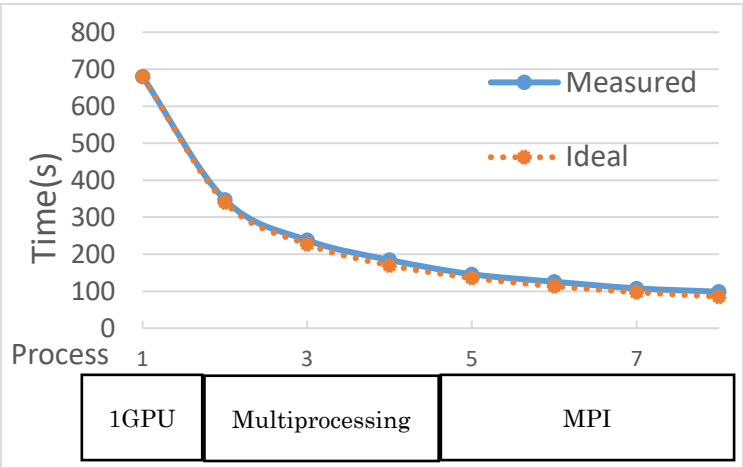


Figure.15 Relationship between the number of process and time in Python AC-based VBQ

Table.7 1 task in Python AC-based VBQ

Resources	Action		Time(s)	Etc.
Single-GPU	Initialize	GA	0.0005	
	Assign GPU to each process		0.6005	Using Tensorflow
	Make Gene		0.0025	×Generation
	Set pool		0.6546	×Generation
MPI	Load Data & model	GA	0.5977	×Gene_num×Generation
	Quantize parameters		0.0018	×Gene_num×Generation
	Evaluate all model		0.9111	×Gene_num×Generation
Single-GPU	Release GPU		0.0001	
	Finish		0.0000	

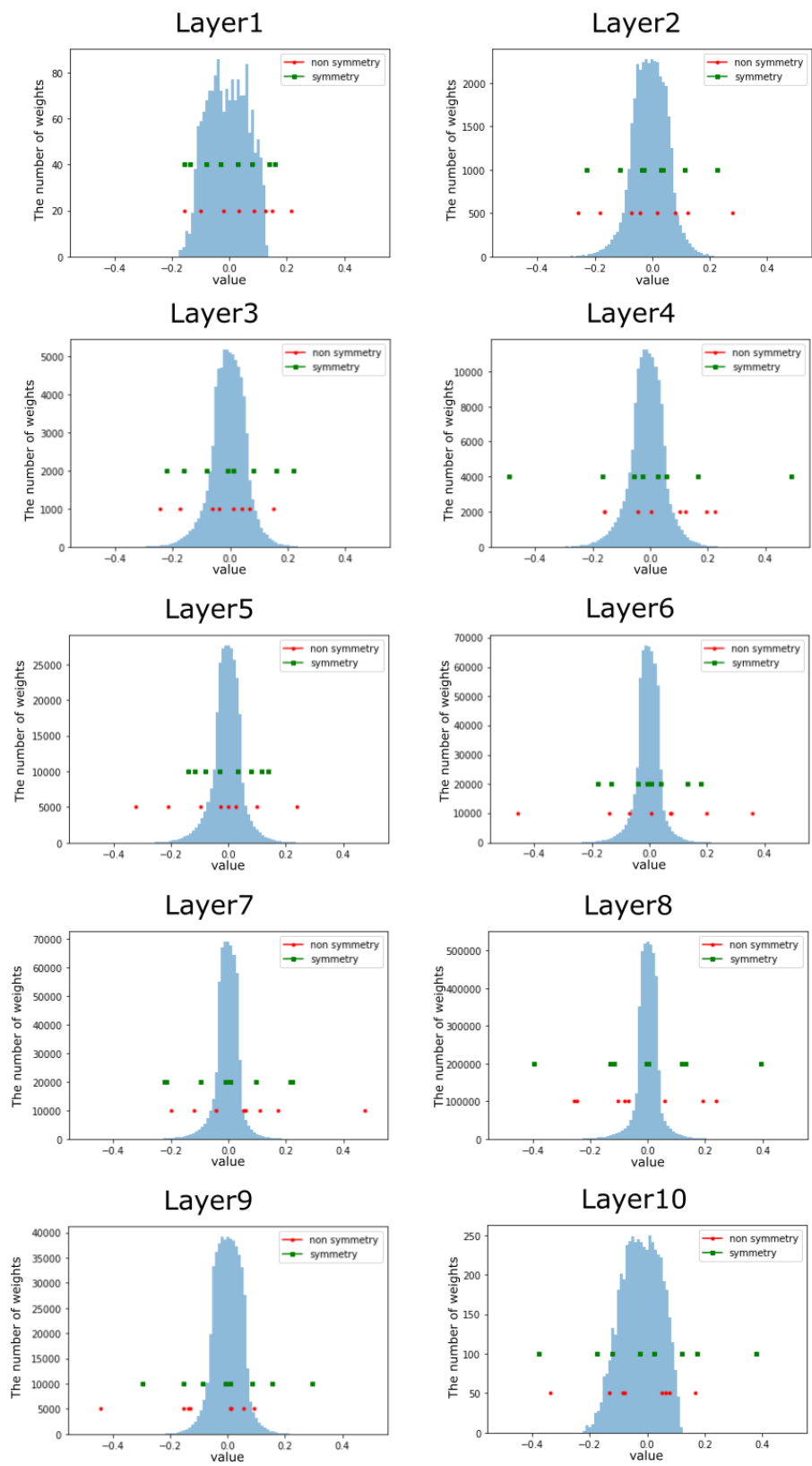


Figure.16 The distribution of weights and representative values in 2 methods

第 5 章 AC-based LQ-Nets

5.1 背景

第 2 章で述べたように、当研究の大きな目標は、VBQ の 2 つの課題 (①ビン境界の決定方法に関する問題, ②量子化後の演算方法に関する問題) に対処できる量子化の最適化手法を実現することである。本章では、第 2 章で述べた AC-based VBQ の最適化手法と、LQ-Nets の量子化手法を組み合わせた、新たな手法 (AC-based LQ-Nets) を提案する。提案手法では、AC-based での最適化が用いられている。また提案手法の実現のために、第 4 章で記述した、Python で一体化した AC-based VBQ を利用している。

5.2 提案手法

本章では、CNN の認識精度を基準として LQ-Nets 量子化の基底値を最適化する手法 (AC-based LQ-Nets) について提案する。AC-based VBQ では重み w のみを量子化していたが、LQ-Nets 量子化を用いて計算量を削減するには、重み w に加え、アクティベーション a の量子化も必要となる。今回、CNN の認識精度をスコアとして最適化を行う遺伝的アルゴリズム (GA) のフレームワークを開発し、 w と a の量子化基底の最適化に用いた。Figure.21 が AC-based LQ-Nets で用いられている遺伝的アルゴリズムのフローである。遺伝的アルゴリズムで世代を重ねることで、近似解を探索することができるようになる。当手法における選択・交叉・突然変異の操作について、詳細を述べる。

5.2.1 選択

選択は、優れた遺伝子を次世代へと引き継ぐための操作である。提案手法においては、Figure.17 のように、基底値の配列 P を 1 つの遺伝子とみなし、配列 P に存在する基底値に従って学習済み CNN に量子化を施す。複数の配列 P の候補の中から、CNN の認識性能が高くなるように数値表現を量子化できる配列を決定する。その後量子化 CNN の認識性能が良い遺伝子を、任意の個数だけ次世代に継承する。この操作によって、次世代で遺伝子が悪化しないことが保証される。

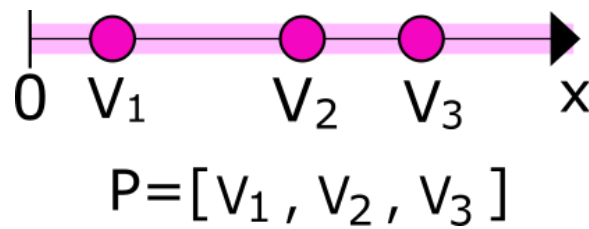


Figure.17 An array P of base-value

5.2.2 突然変異

突然変異は、遺伝子の一部情報をランダムに変更する操作である。提案手法では、Figure.18のように P 配列内の基底値をランダムに変更する操作が行われる。この際、ランダムに選択した基底値が P 配列の中で最小値・最大値をとらない場合は、隣接する 2 つの基底値の間の値いずれかに変更することで、前世代から受け継がれた P 配列の遺伝子情報が大きく変わることを防いでいる。P 配列の中で最小値・最大値の基底値が選択された際は、原点を中心とする正規分布に従って、選ばれた基底値をランダムに変更する。正規分布の標準偏差については、遺伝的アルゴリズム(GA)を施す前に予め設定しておく。この突然変異は、遺伝的アルゴリズムによる最適化が、局所最適解に陥るのを防ぐ役割を持っている。

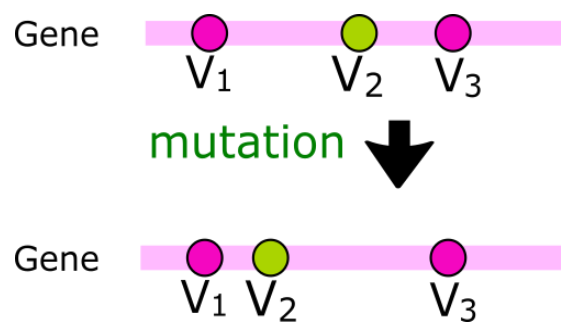


Figure.18 Mutation method in our method

5.2.3 交叉

交叉は、生物が交配によって子孫を残すことをモデル化した操作である。交叉の中にも様々な手法が存在しているが、2点交叉と呼ばれる手法が広く用いられている。2点交叉は、Figure.19のように、ランダムで選ばれた 2 つの交叉点に挟まれた区間を入れ替える手法である。提案手法においてもこの 2 点交叉を取り入れている。提案手法では、2 つの異なる遺

伝子と、交叉する区間の中心値、交換する基底値の個数をそれぞれランダムに指定し、基底値を交換する操作が行われている。具体的には、Figure.20 のように、①交叉する区間の中心値と、②交換する基底値の個数を、可能な範囲でランダムに決定した後、決定した2つの値を基に、2つの遺伝子の中で基底値の入れ替えを行うことで、2つの新たな遺伝子を作成する操作が行われる。

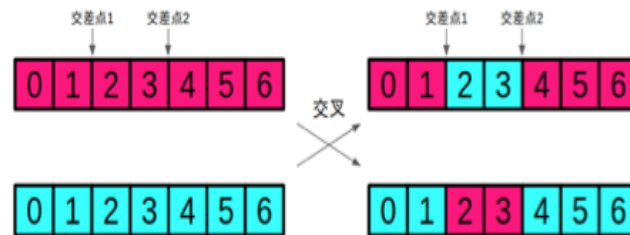


Figure.19 General 2-points crossover. [32]

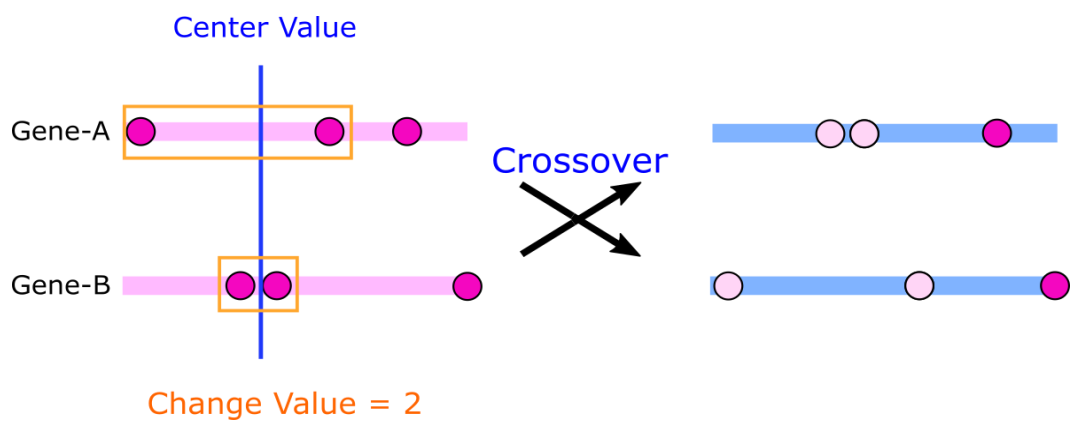


Figure.20 Crossover method in our method

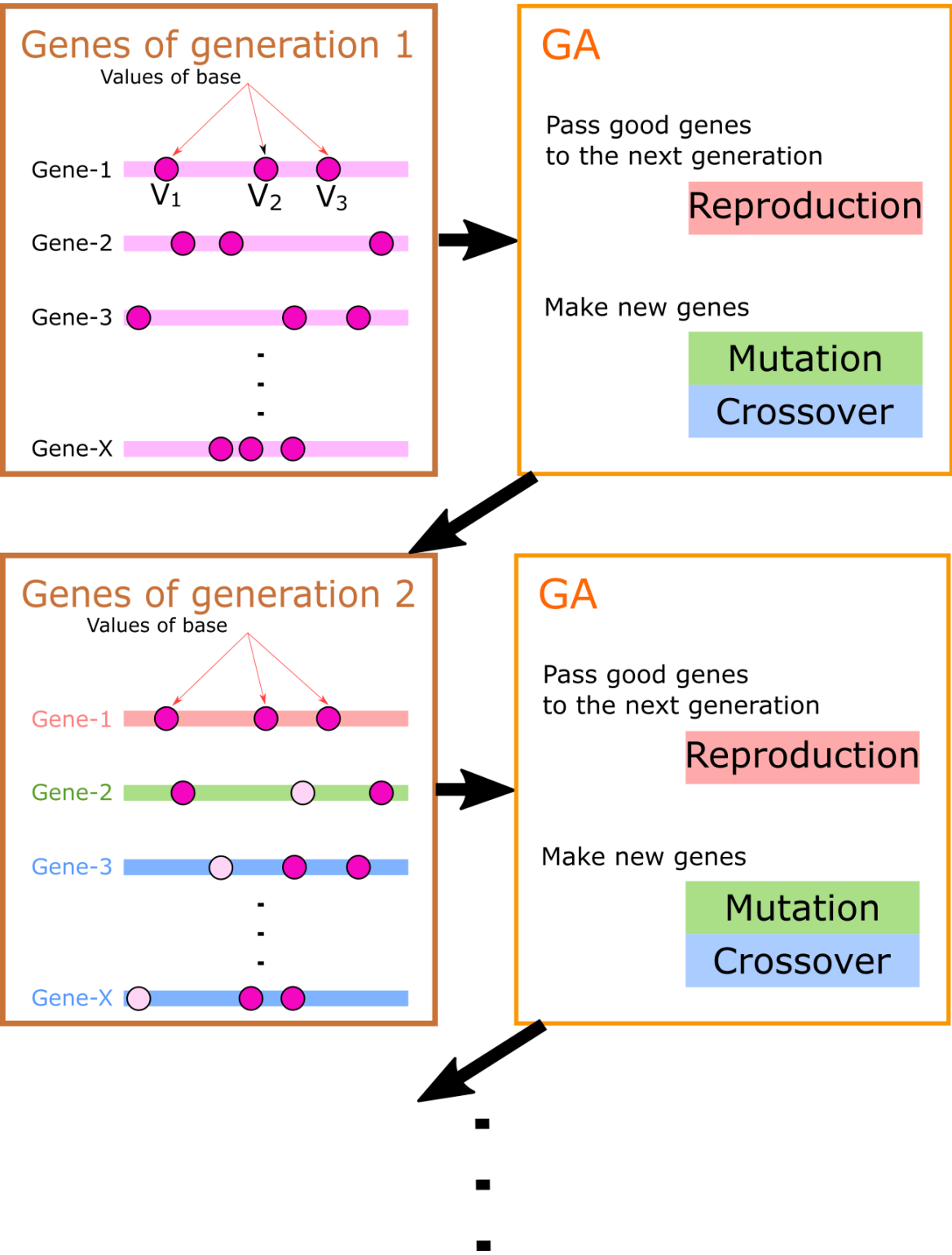


Figure.21 The structure of GA in our method.

5.3 実験内容

LQ-Nets 量子化の基底値を, CNN の認識精度をスコアとした遺伝的アルゴリズム(GA)を用いて最適化した. 当実験では学習済み CNN として VGG-like モデルを用い, 学習データセットとして CIFAR-10 を用いた.

AC-based LQ-Nets の実験で用いられている学習済みモデルは, 第 3 章・第 4 章での実験で用いられている VGG-like モデルとは違い, 学習の際に Dropout をはじめとする最適化技術を施した VGG-like モデルを使用している. 当実験で用いた学習済み VGG-like モデルの認識精度は, 32Bit 精度で 90.29%となっている. これは AC-based LQ-Nets で使用した学習済み VGG-like モデルと比べ約 12%良いものとなっている. この VGG-like モデルを FP32 のまま学習した. モデルの実装には, Keras と Tensorflow を用いた. 前述した通り, 学習した VGG-like モデルの認識精度は, CIFAR-10 のテストデータに対して 90.29%であった. この学習済みモデルが本実験の基準となる.

学習済み CNN の各レイヤについて, 重み w とアクティベーション a を抽出して量子化し, これをモデルに差し戻して, 量子化 CNN の認識精度を評価してスコアとした. このスコアに基づいて, LQ-Nets 量子化の基底値を, CNN の認識精度をスコアとした遺伝的アルゴリズム(GA)を用いて最適化した. VGG-like モデルの各レイヤの重み w とアクティベーション a について, それぞれ個別に抽出・量子化し, モデルに差し戻して認識精度を評価してスコアとした. このスコアに基づいて, LQ-Nets 量子化の基底値を遺伝的アルゴリズムで最適化した. Layer1 の重み w と, Layer10 (最終層) のアクティベーション a を除く 18 種類のレイヤのそれぞれにおいて, 最適化された基底値を用いて数値表現を量子化し, 作成した量子化 CNN の最終的な認識精度を計算した. この際 LQ-Nets 量子化は全て 3bit 精度で実行し, 遺伝的アルゴリズムに関する条件は Table.8 に示すようにした. 比較対象として, オリジナルの LQ-Nets に加え, QE-based VBQ に LQ-Nets 量子化相当の代表値制限を加えたもの (以下 Non-learning LQ-Nets と呼ぶ) を合わせて計算した. オリジナルの LQ-Nets と AC-based LQ-Nets との相違点を Table.9 に示す. オリジナルの LQ-Nets では基底値の最適化と並行してモデルの再学習を行なっているが, 本提案では再学習は実施していない. またオリジナルの LQ-Nets では重みの量子化についてチャンネルごとに個別の量子化形式を適用しているが, 本提案では単一レイヤについては同一の量子化形式を採用している. この 2 点において, AC-based LQ-Nets はオリジナルの LQ-Nets と比較して, 量子化後の数値表現の自由度がより制約されたものとなっている.

当実験では, 第 3 章・第 4 章での実験と同様, 計算を行うハードウェアとして産業総合研究所 (AIST) が運営するスーパーコンピュータシステム ABCI を用いた. Table.4 で示すように, ABCI は 1 つの計算ノードに 4 つの GPU を持つハードウェアである.

Table.8 The conditions in GA

The number of gene in one generation	50
Total Generation	200
Reproduction (%)	30
Mutation (%)	20
Crossover (%)	50

Table.9 The comparison of the 3 methods.

	AC-based LQ-Nets	Non Learning LQ-Nets	LQ-Nets
Retraining	No	No	Yes
Optimization criteria	Accuracy-based	Quantization Error-based	Quantization Error-based
Basis vector	per Layer	per Layer	per Channel in weights
Arithmetic operation	Yes	No	Yes

5.4 実験結果と考察

5.4.1 提案手法と従来手法の認識性能比較

量子化前後の認識精度の違いを Table.10 に示す. AC-based LQ-Nets と Non Learning LQ-Nets は Keras (Tensorflow v.2) ベース, オリジナルの LQ-Nets は Tensorflow v.1 ベースであるため, FP32 での結果に差異がある. モデルの再学習を伴わない AC-based LQ-Nets と Non Learning LQ-Nets の比較より, 量子化に伴う認識精度の劣化は, AC-based の最適化の方が QE-based よりも約 5%改善することが分かった. これより, 重み w だけでなく, アクティベーション a を含めた量子化の最適化においても, AC ベースでの最適化手法が優れていることが明らかになった. また, モデルの再学習を伴うオリジナルの LQ-Nets と比較しても, 提案手法が約 2%優れているという結果になった. 先に述べたように, 本提案における量子化後の数値表現の自由度はオリジナルの LQ-Nets と比べて制約されていることを考慮すると, CNN の内部数値表現の量子化においては AC ベースでの最適化が効果的であることがわかる. また量子化の際のモデル再学習を行わなくとも, 遺伝的アルゴリズム(GA)を代替手段として用いることで, 量子化 CNN の認識精度向上を行えることが分かった. 当提案手法では, 重み w の量子化はレイヤ毎に行われているが, オリジナルの LQ-Nets ではチャンネル毎に行われている. このように, 2 手法の間で重み w の量子化条件が違うという差異があるため, 遺伝的アルゴリズム(GA)が, モデルの再学習と比べてどれだけ効果的かを議論することは

困難である．しかし重み w の量子化において，チャンネルごとに遺伝的アルゴリズム(GA)の最適化を行うことで，AC-based LQ-Nets の結果が更に良くなることが推察できる．

Table.10 The result of AC-based LQ-Nets

	Accuracy (%)		Diff(%)
	FP32	3bit-Quantization	
AC-Based LQ-Nets	90.29	89.32	-0.97
Non Learning LQ-Nets	90.29	84.53	-5.76
LQ-Nets	91.14	88.54	-2.60

5.4.2 重み w とアクティベーション a の考察

Figure.22 と Figure.23 は，当実験の Non Learning LQ-Nets と AC-based LQ-Nets の 2 手法で最適化された代表値と，学習済み CNN の FP32 の重み w やアクティベーション a の分布を，CNN のレイヤ毎に図示したものである．当実験において，量子化モデルの先頭レイヤの重み w と最終レイヤのアクティベーション a は量子化していない為，それぞれ 9 層分の図示となっている．当実験では活性化関数として ReLU が用いられており，アクティベーション a は負の値を取らない．またこの影響で，アクティベーション a に LQ-Nets 量子化を行うと，全てのレイヤにおいて，基底値の最小値は 0 となる．

前半に示す重み w の量子化では，AC-based は QE-based と比較して基底値がおおむね大きな値に移動していることがわかる．重み w の分布はゼロ近傍に集中しているため，QE-based ではゼロ近傍の分解能を高めるべく，小さな値に最適化される．一方，AC-based ではより大きな値の分解能を高める方向に最適化されることから，CNN の推論においては，絶対値の大きな値の精度がより重要であることが分かる．一方アクティベーション a の量子化では，FC 層 (VGG-like モデルにおける 8 層目以降) においては，より小さな値の分解能を高める方向に最適化が行われており，絶対値の小さな値の精度がより重要であることが分かる．畳み込み層においては，AC-based と QE-based で顕著な差異は認められなかった．この結果は，畳み込み層のアクティベーション a において，どの値域の解像度が重要かはまちまちであることを示している．今後はこれらの性質を利用して，より精度の高い量子化の最適化手法について検討していく．

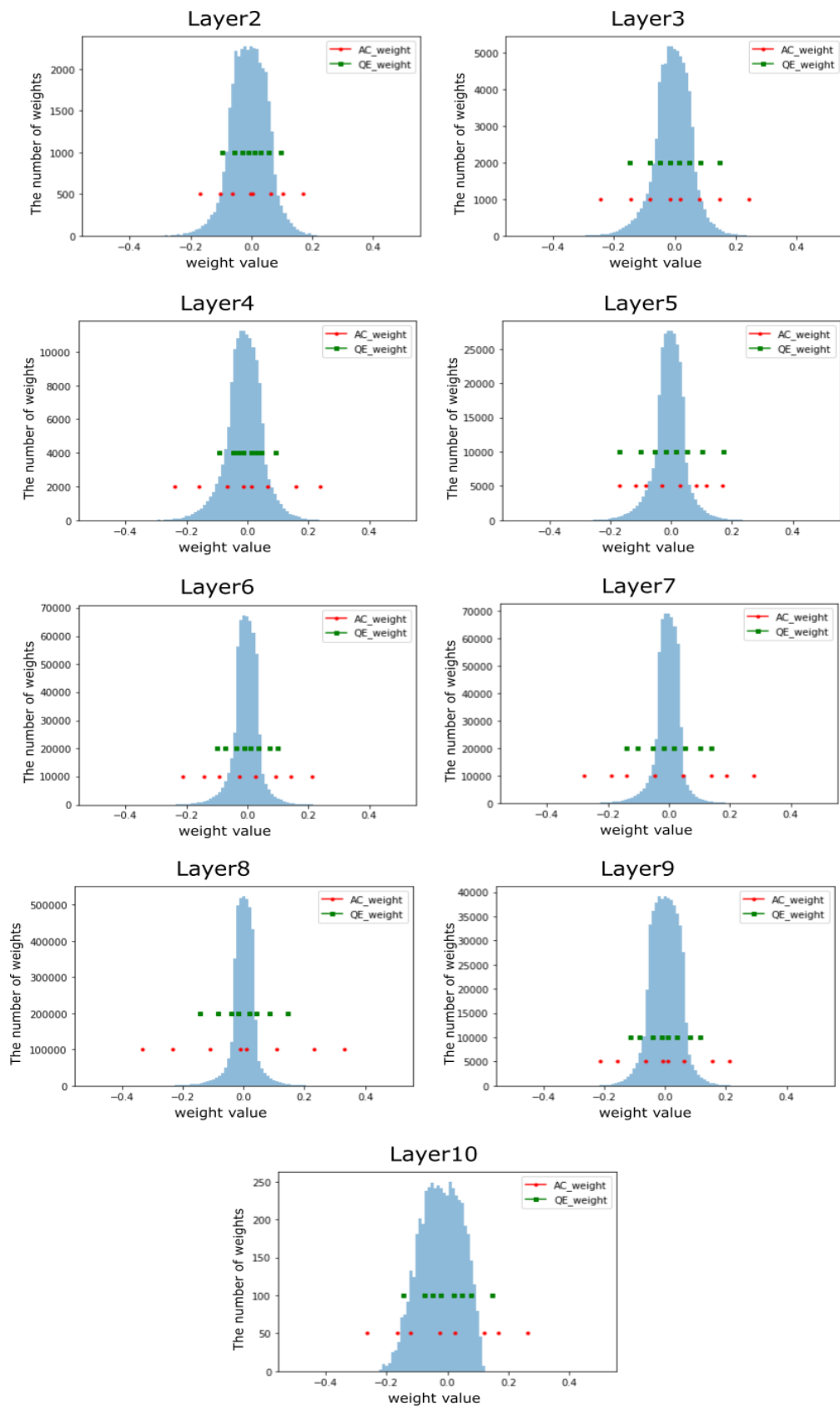


Figure.22 The weight values in AC-based LQ-Nets and Non learning LQ-Nets.

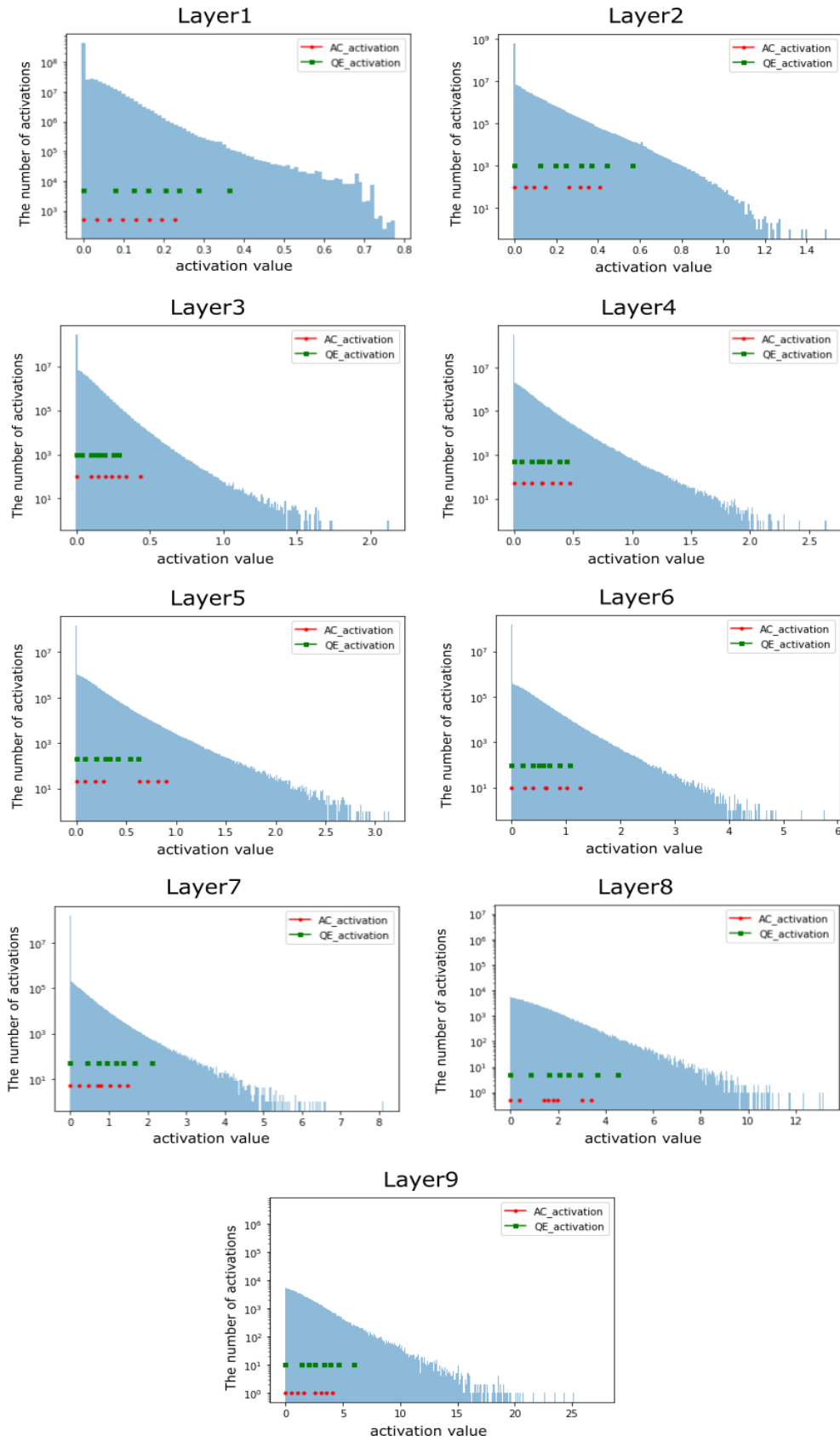


Figure.23 The activation values in AC-based LQ-Nets and Non learning LQ-Nets.

5.4.3 遺伝的アルゴリズムの世代遷移とその考察

Figure.24, Figure.25 は、当実験での AC-based LQ-Nets における遺伝的アルゴリズムで、世代ごとに最良の遺伝子がどのように推移するかについて、CNN のレイヤ毎に図示したものである。遺伝的アルゴリズムによって 100 世代程度の更新がなされた際、多くのレイヤで基底値の推移が穏やかになっている。従って、当実験で使用した VGG-like モデルの量子化の最適化は、遺伝的アルゴリズムを 100 世代程度繰り返すことでおおよそ収束したと推察できる。

また重み w 、アクティベーション a の双方とも、100 世代目以降は、1 つの基底値だけが変化しているレイヤが多く見られた。これより、遺伝的アルゴリズムが 100 世代以上繰り返された場合、最良の遺伝子は、主に突然変異によって更新されていることが分かる。遺伝的アルゴリズムの更新が 100 世代以上繰り返された際に突然変異による更新割合を増やす操作や、正規分布の標準偏差の値を大きく設定し、突然変異の際に変動する値の幅を広げる等の操作を遺伝的アルゴリズム(GA)のプログラムに加えることで、遺伝的アルゴリズムの性能がさらに向上すると推察できる。

5.4.4 量子化の条件変更

考察のため、AC-based LQ-Nets を用いて、一部の Layer のみに量子化の最適化を行った。当実験で用いた VGG-like モデルを AC-based LQ-Nets で最適化した際、Table.11 で示す計 9 つのユニットにおいて、推論時の重み w とアクティベーション a の積演算が、シンプルな加減算を用いたものとなる。当実験では、AC-based LQ-Nets による量子化の最適化を、重み w とアクティベーション a のレイヤ毎に行うのではなく、Table.11 で示したユニット毎に行った。1 つのユニットに存在する数値表現を量子化した際は、他のユニットは全て FP32 のまま推論を行った際の CNN の認識精度を Table.12 に示す。Table.12 より、1 つのユニットに限り量子化の最適化を行った際は、FP32 での推論結果よりもよい結果が得られるレイヤが存在することが分かった。またユニットによって、FP32 での結果に違いが表れることから、ユニット毎、あるいはレイヤ毎に、最適な bit 幅は異なることが推察できる。

Table.13 は、当実験の AC-based LQ-Nets において、LQ-Nets 量子化を 1~3bit 精度に変動させた際の量子化前後の認識精度の違いについてまとめたものである。LQ-Nets 量子化の精度が 2bit 以上であれば、CIFAR-10 データセットの認識精度は 8 割以上で維持できること、また 1bit の量子化精度では性能が急落することが分かった。また当実験より、重み w の bit 数の削減の方が、アクティベーション a の bit 数削減よりも量子化 CNN の認識精度の低下に影響を及ぼすことが分かる。

Table.11 The structure of multiply calculation units

parameter&layer	setting
weight_1	FP32
activation_1	Unit1
weight_2	
activation_2	Unit2
weight_3	
...	...
activation_8	Unit8
weight_9	
activation_9	Unit9
weight_10	
activation_10	FP32

Table.12 Comparison between 1 Unit 3-bit quantization and FP32 in CNN accuracy.

Unit	Accuracy(%)	
	FP32	1Unit 3bit-Quantization
Unit1	90.29	90.23
Unit2	90.29	91.00
Unit3	90.29	90.17
Unit4	90.29	90.12
Unit5	90.29	90.44
Unit6	90.29	90.40
Unit7	90.29	90.36
Unit8	90.29	90.25
Unit9	90.29	90.33

Table.13 Changing bit-width in AC-based LQ-Nets.

activation(bit)	1bit	2bit	3bit
weight(bit)			
1bit	31.30%	48.90%	53.83%
2bit	67.18%	80.73%	86.96%
3bit	69.24%	86.98%	89.32%

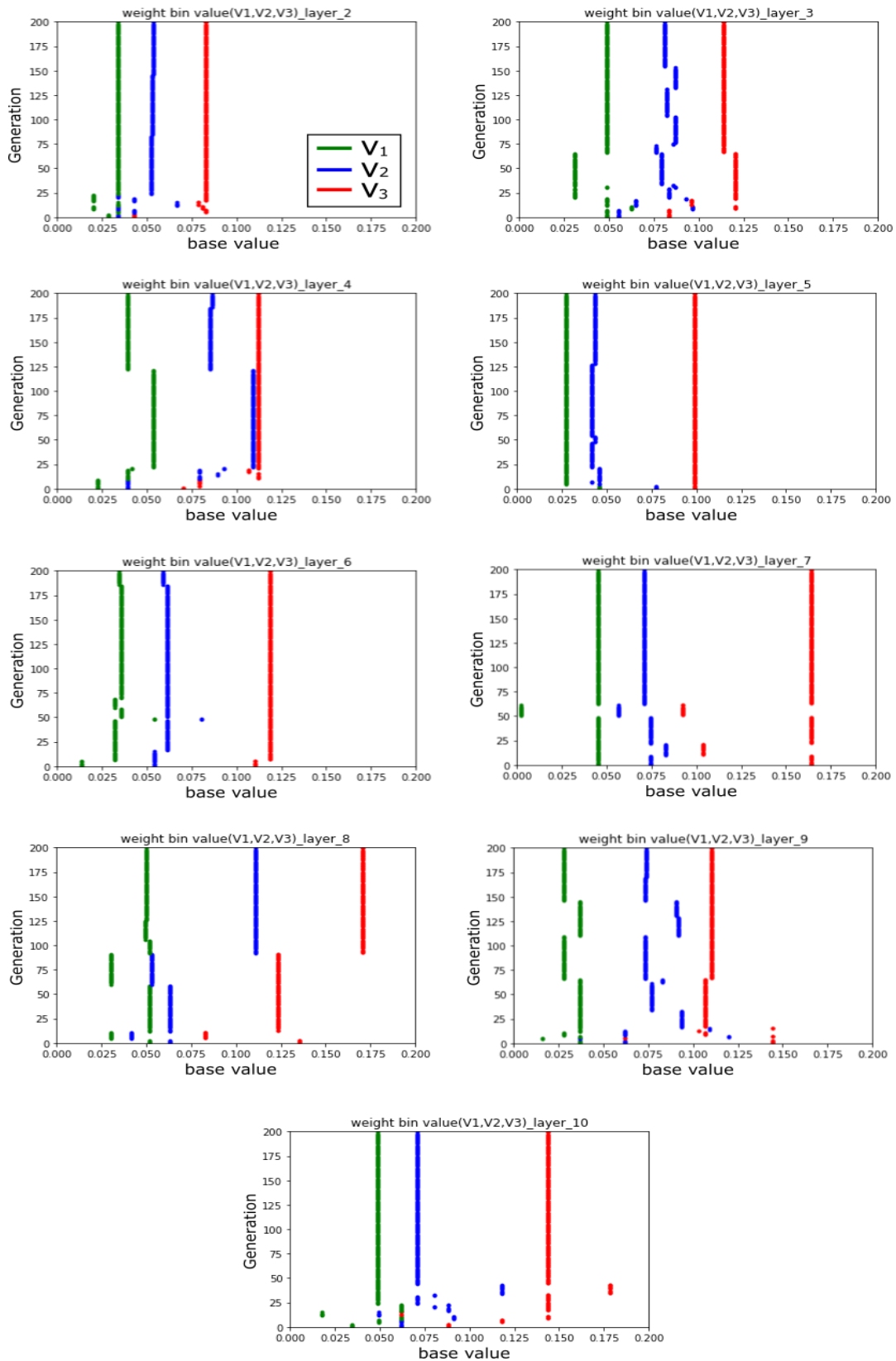


Figure.24 The transition of base values in weight GA

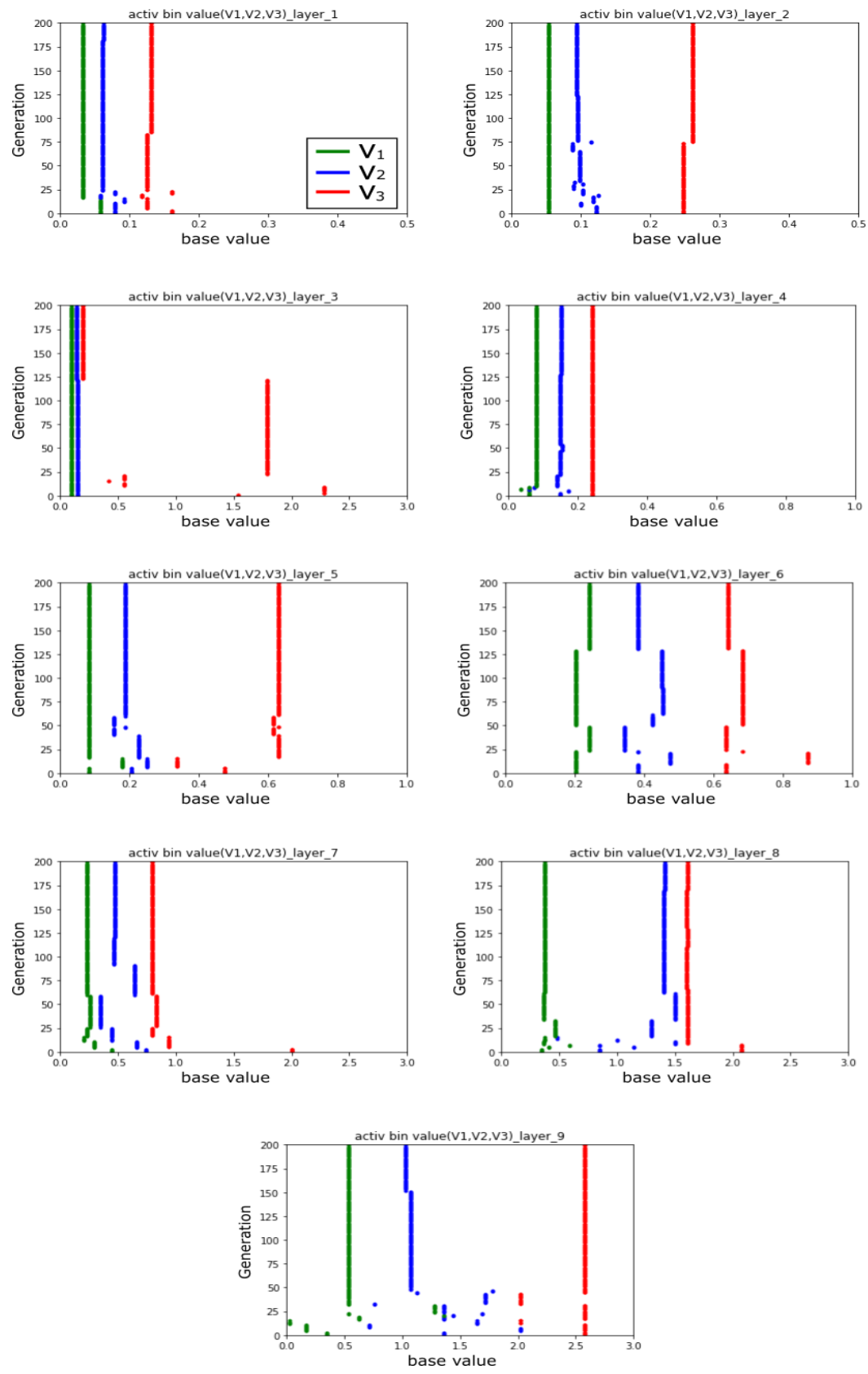


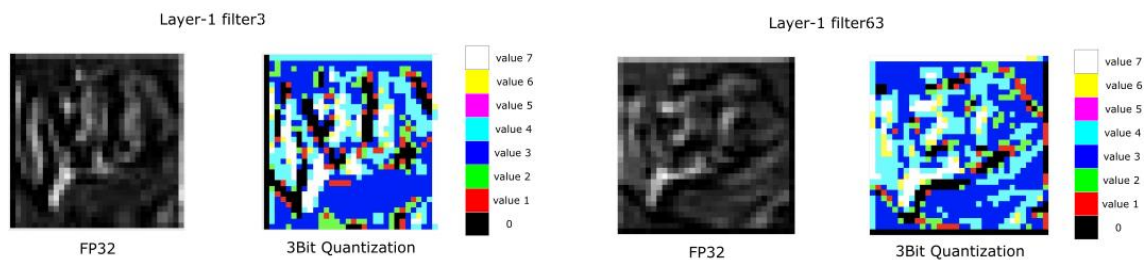
Figure.25 The transition of base values in activation GA

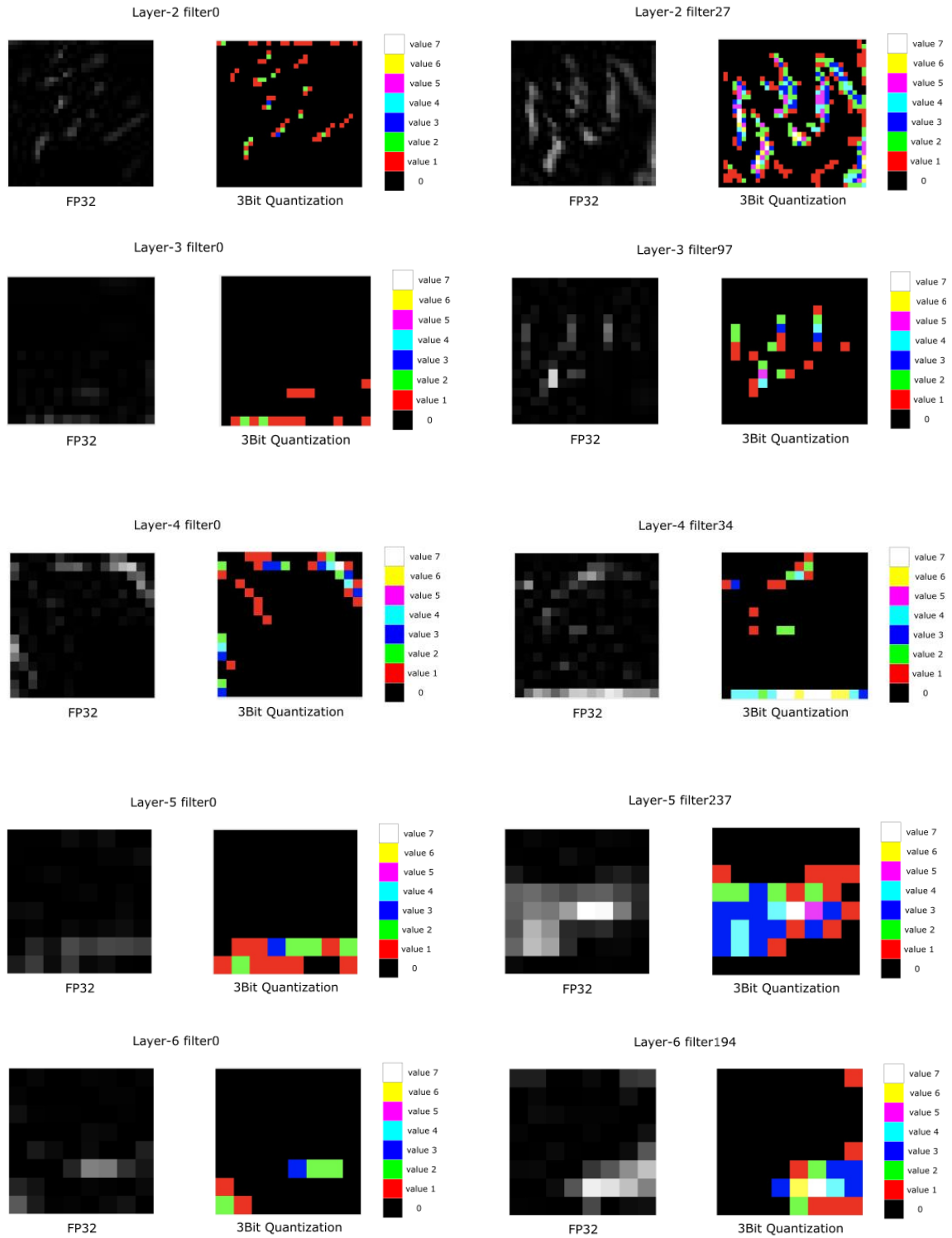
5.4.5 量子化前後の中間層の出力

Figure.27 は、FP32 精度の VGG-like と、AC-based LQ-Nets によって最適化された 3bit 精度の VGG-like の 2 種類のモデルを用いて、アクティベーション a の出力を比較した図である。この際、Figure.26 に示す、正解ラベルが Cat である CIFAR-10 のテストデータの 1 枚を入力とした。Figure.27 において、Conv2D 層に当たる Layer-1 から 7 までの出力は、チャンネル毎にアクティベーションの出力を確認した。FC 層に当たる Layer8 以降の出力は、全ての出力を 1 枚の画像に結合している。Figure.27 の FP32 での出力では、アクティベーションの出力を、画素値の取りうる範囲(0~255)にレイヤ毎に正規化し、グレースケールで結果を出力している。3bit の量子化形式では、8 つの代表値を識別するため、値の小さい順に(0~Value 7)まで、出力の種類に応じて色分けを行っている。Figure.27 より、FP32 精度のモデルの結果において、画素値が高くなっている箇所と、3bit 精度のモデルの結果において、大きい代表値が用いられている箇所が概ね同じであることが分かる。従って、AC-based LQ-Nets で使用されているアクティベーション a の量子化の最適化が、概ね機能しているといえる。また 3bit 精度のモデルの結果において、取りうる 8 種類の代表値のうち、アクティベーションが比較的小さな値に置き換わっていることから、提案手法での最適化では、アクティベーションが小さな値の分解能を高める方向に代表値が最適化されることが推察できる。



Figure.26 The input image of this experiment.





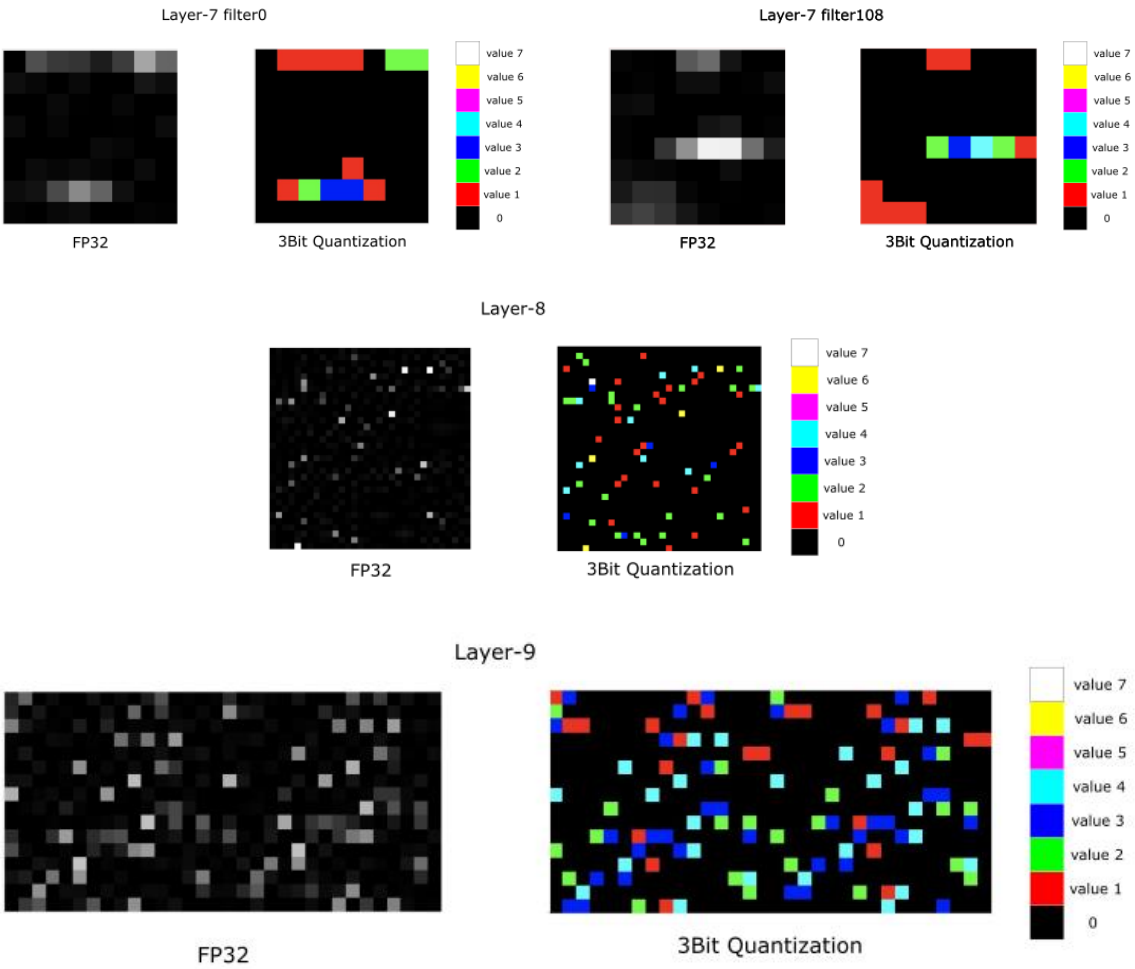


Figure.27 The deliberation of activation output of each layer.

第 6 章 結論と今後の課題

6.1 結論

近年, CNN の精度向上のため, CNN のサイズが肥大化している. この結果, CNN の演算において, 必要なメモリ量や計算コストが増大している. 巨大な CNN の推論を携帯端末などのエッジデバイス上で処理する上で, FPGA 等のハードウェアの活用が進められている. これらハードウェアを活用する上で, CNN の数値内部表現の工夫による使用メモリ量や計算コストの削減は欠かすことができない. 使用メモリ量や計算コストを低減するため, CNN の数値表現をより小さな bit 幅にする量子化についての研究が多くなされてきた. しかし低い bit 幅の表現能力は必ずしも十分ではなく, 単純な量子化を使用した多くの研究において, 量子化 CNN の認識精度は不安定であった. 量子化により計算コスト削減とメモリ使用量削減を実現しつつ, 量子化 CNN の性能を維持できる, 不規則な量子化についても研究がなされてきた. 不規則な量子化として VBQ が存在するが, VBQ には①量子化 CNN が高い推論精度を維持するために最適な手法が明らかではない点, ②量子化 CNN を推論する際の計算方法を考慮していない点, の 2 点が問題点として挙げられていた.

本研究では, 前述した 2 つの問題点に対処するため, VBQ の 1 種である LQ-Nets を対象に, その量子化形式の最適化を推論精度基準で実施する手法(AC-based LQ-Nets)を提案した. 提案手法では, 先行研究と比較し, 量子化 CNN の推論精度を約 2% 高めることができることを示した. また最適化結果の分析より, CNN 内部数値表現の量子化では量子化誤差をまんべんなく減らすよりも, 区間を絞って分解能を高める方が有効であること, LQ-Nets 量子化の精度が 2bit 以上であれば, CIFAR-10 データセットの認識精度が 8 割前後で維持できること, LQ-Nets の量子化形式を用いて量子化された, 重み w の最適化を行う際, チャンネル毎ではなく, レイヤ毎の最適化でも, 量子化 CNN の認識精度が維持できることが明らかになった.

6.2 今後の課題

当研究では、重み w とアクティベーション a を最適化する際、量子化時の数値表現の bit 幅はハイパーパラメータで与えられていた。LQ-Nets の量子化形式のユニット毎、あるいはレイヤ毎に、最適な bit 幅は異なると考えられる。そのため、量子化と bit 幅を同時に最適化できる手法についての検討が課題として残っている。

また当研究で使用された遺伝的アルゴリズム(GA)は、改良の余地が残されている。突然変異の割合を増やした実装等を行い、遺伝的アルゴリズムの改良を検討していく。

さらに当研究では、学習済み CNN として VGG-like を用いて実験しているが、当モデルでの評価だけでは、当論文で述べた結果が一般の CNN に当てはまるかどうかは定かではない。当研究での主張を強めるためには、他の CNN を用いた検討を行うことが必要だと考える。加えて RNN [43]や LSTM [44]等の時系列モデルや、GAN [45]等の生成モデルにおいても適用し、モデル毎に最適な量子化形式が異なるかどうかについて検討することも課題である。

謝辞

本研究を進めるにあたり，ご多忙の中貴重な時間を割いて，熱心な御指導，御意見を賜りました，工藤 知宏教授に深く感謝いたします。また研究において様々な御意見を頂きました，池上 努氏，大内 真一氏，高野 了成氏，Akram Ben Ahmed 氏に感謝いたします。特に，池上 努氏にはご意見だけでなく，研究態度の規範として，また同じ分野の研究者として多大な御指導，技術的なご支援，熱心な励ましを頂き，深く感謝いたします。さらに研究室での2年間を通じて，共に励ましあい，貴重な意見交換を行った研究室のメンバー全員と，2年間にわたり筆者の大学院生活を支えてくれた家族，友人に感謝の意を表します。

参考文献

- [1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 pp.1097-1105, 2012.
- [2] M. D Zeiler, and R. Fergus, "Visualizing and Understanding Convolutional Networks", arXiv:1311.2901, 2013.
- [3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions", arXiv:1409.4842v1, 2014.
- [4] J. Hu, L. Shen and G. Sun, "Squeeze-and-Excitation Networks", arXiv:1709.01507, 2017.
- [5] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes", arXiv:1711.04325, 2017.
- [6] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet Training in Minutes," in Proceedings of the 47th International Conference on Parallel Processing - ICPP 2018. New York, New York, USA: ACM Press, 2018, pp. 1-10.
- [7] H. Mikami, H. Suganuma, P. Uchupala, Y. Tanaka, and Y. Kageyama, "ImageNet/ResNet-50 Training in 224 Seconds", arXiv:1811.05233, 2018.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016, pp. 770–778.
- [9] K. Guo, S. Zeng, J. Yu, Y. Wang, H. Yang, "A Survey of FPGA based Neural Network Accelerator", arXiv:1712.08934v3, 2017.
- [10] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '15. New York, New York, USA: ACM Press, 2015, pp. 161-170.
- [11] R. Zhou Ding, Z. Liu, R.D(Shawn) Blanton, D. Marculescu, "Quantized Deep Neural Networks for Energy Efficient Hardware-based Inference", ASPDAC '18: Proceedings of the 23rd Asia and South Pacific Design Automation Conference, 2018, pp. 1-8.
- [12] K. He, X. Zhang, S. Ren and J. Sun: "Deep Residual Learning for Image Recognition", arXiv:1512.03385, 2015.
- [13] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited

- numerical precision”, Proceedings of the 32nd International Conference on Machine Learning - Volume 37, pp. 1737-1746, 2015.
- [14] M. Courbariaux, Y. Bengio, and J.-P. David, “BinaryConnect: Training Deep Neural Networks with binary weights during propagations”, arXiv: 1511.00363, 2015.
- [15] R. Mohammad, V. Ordonez, R. Joseph, and F. Ali, “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”, Master Thesis Reference in Computer Vision ECCV 2016, L. Bastian, J. Matas, and S. Nicu, and W. Max, Eds. Cham: Springer International Publishing, 2016, pp. 525-542.
- [16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector”, arXiv:1512.02325, 2015.
- [17] A. Zhou, A. Yao, “Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights”, in International Conference on Learning Representations, ICLR2017, 2017.
- [18] S.-i. O’uchi, H. Fuketa, T. Ikegami, W. Nogami, T. Matsukawa, T. Kudoh, and R. Takano, “Image-Classifer Deep Convolutional Neural Network Training by 9-bit Dedicated Hardware to Realize Validation Accuracy and Energy Efficiency Superior to the Half Precision Floating Point Format”, in 2018 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 5 2018, pp. 1-5.
- [19] D. Zhang, J. Yang, D. Ye, and G. Hua, “LQ-Nets Learned Quantization for Highly Accurate and Compact Deep Neural Networks”, arXiv: 1807.10029v1, 2018.
- [20] R. Ding, Z. Liu, R. Shi, D. M. Culescu, and R. D. Blanton, “LightNN: Filling the Gap between Conventional Deep Neural Networks and Binarized Networks”, arXiv: 1802.02178v1, 2017.
- [21] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T. Yang, and E. Choi, “MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks”, arXiv:1711.06798, 2018.
- [22] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, arXiv: 1510.00149, 2016.
- [23] D. Miyashita, E. H. Lee, and B. Murmann, “Convolutional Neural Networks using Logarithmic Data Representation”, arXiv:1603.01025, 2016.
- [24] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, “Fixed Point Quantization of Deep Convolutional Networks”, in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ser. ICML’16. JMLR.org, 2016, pp. 2849-2858.
- [25] R. Kamiya, T. Yamashita, M. Ambai, I. Sato, Y. Yamauchi, and H. Fujiyoshi, “Binary-decomposed DCNN for accelerating computation and compressing model without retraining”, in Workshop on International Conference on Computer Vision, 2017.
- [26] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, “Object Recognition with Gradient-based Learning”, in 1999 Shape, Contour and Grouping in Computer Vision pp. 319-345.
- [27] F. Chen, N. Chen, H. Mao, and H. Hu, “Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST)”, arXiv:1811.08278, 2019.

- [28] W. Nogami, T. Ikegami, S. O'uchi, R. Takano, Y. Kishi, and T. Kudoh, "Optimization of Numerical Expression in CNN using Genetic Algorithm", Institute of Electronics, Information and Communication Engineers Technical Report, 2018, pp. 193-198.
- [29] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Master Thesis Reference Networks", in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4107-4115.
- [30] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Network: Training Neural Networks with Weights and Activations Constrained to +1 or -1", 2016.
- [31] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training Low bitwidth Convolutional Neural Networks with Low bitwidth Gradients", arXiv:1606.06160, 2018.
- [32] W. Nogami, T. Ikegami, S. O'uchi, R. Takano and T. Kudoh, "Optimizing Weight Value Quantization for CNN Inference," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852331.
- [33] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv: 1409.1556, 2015.
- [34] J. McCall, "Genetic algorithms for modelling and optimisation". Journal of Computational and Applied Mathematics, pp. 205-222. 2005.
- [35] D. Anderson, J. R. Vliment, and M. Spiropulu, "An MPI-based Python Framework for Distributed Training with Keras", arXiv: 1712.05878, 2017.
- [36] M. P. Whelan, and D. F. Mackey, "schwimmbad: A uniform interface to parallel processing pools in Python", The Journal of Open Source Software papers, pp. 1-2, 2017.
- [37] S. O'uchi, H. Fuketa, T. Ikegami, W. Nogami, T. Matsukawa, T. Kudoh, and R. Takano, "Image-Classifer Deep Convolutional Neural Network Training by 9-bit Dedicated Hardware to Realize Validation Accuracy and Energy Efficiency Superior to the Half Precision Floating Point Format", in 2018 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 5 2018, pp. 1-5.
- [38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", The Journal of Machine Learning Research, Volume 15 Issue 1, pp. 1929-1958, 2014.
- [39] C. Shorten, and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning", Journal of Big Data, 2019.
- [40] L. Rere, M. Fanany, and A. Arymurthy, "Simulated Annealing Algorithm for Deep Learning", Procedia Computer Science 72, pp. 137-144, 2015.
- [41] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", Science (New York, N.Y.), vol. 220, no. 4598, pp. 671-80, 5 1983.

- [42] V.Cerny, “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm”, *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.
- [43] J. S. Bridle, “Alpha-nets: A recurrent neural network architecture with a hidden Markov model interpretation”, *Speech Communication*, vol. 9, no. 1, pp. 83–92, 2 1990.
- [44] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets”, in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.
- [46] S. Ioffe, and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, *arXiv: 1502.03167*, 2015.
- [47] J. Bjorck, C. Gomes, B. Sekman, and K. Q. Weinberger, “Understanding Batch Normalization”, *arXiv:1806.02375*, 2018.
- [48] A. Maurya, “Not Just Introduction To Convolutional Neural Networks”, <https://becominghuman.ai/not-just-introduction-to-convolutional-neural-networks-part-1-56a36b938592>, 最終更新日: 2018.12/14, 最終閲覧日: 2021.1/27.
- [49] S. Manna, “Converting MNIST dataset for Handwritten digit recognition in IDX Format to Python Numpy Array”, <https://medium.com/the-owl/converting-mnist-data-in-idx-format-to-python-numpy-array-5cb9126f99f1>, 最終更新日: 2018.2/13, 最終閲覧日: 2021.1/27.
- [50] “CIFAR-10 and CIFAR-100 datasets”, <http://www.cs.toronto.edu/~kriz/cifar.html>, 最終閲覧日: 2021.1/27.

学会発表

- [1] 冨永 一輝, 池上 努, 潘 虹芸, 工藤 知宏, “遺伝的アルゴリズムを用いた LQ-Nets 量子化の最適化”, 信学技報, vol. 120, no. 331, NC2020-33, pp. 7-12, 2021 年 1 月.
- [2] 潘 虹芸, A.B.Ahmed, 池上 努, 冨永 一輝, 工藤 知宏, “量子化機械学習の FPGA 実装”, 信学技報, vol. 120, no. 339, RECONF2020-69, pp. 63-68, 2021 年 1 月.