# 修 士 論 文

畳込みニューラルネットワーク向け重み量子化に関する研究

Training Convolutional Neural Networks
with Weights Quantization


**報 告 者**

学籍番号: 1205065
氏名: 氏原 収悟


**指 導 教 員**

密山 幸男 准教授

平成 **30** 年 **2** 月 **12** 日


高知工科大学 大学院工学研究科

基盤工学専攻 電子・光システム工学コース

# 目次

# 第 1 章　序論

深層学習の手法の一つである畳込みニューラルネットワーク（CNN：Convolutional Neural Networks)とは，人間などの視神経系を模倣したニューロンモデルを多層にしたものである．CNN は現在，医療分野では新薬の発見や画像診断の自動化[1]，自動車分野では自動運転など様々な分野で実用化が進められている．CPU (Central Processing Unit) や GPU (Graphics Processing Unit)の性能向上によって, CNN の膨大な演算量に起因する問題は軽減されつつある. しかし，より高い認識精度得るために学習処理に求められる演算量は増加する一方である. また, ソフトウェアによる実装には処理速度や消費電力の面で限界がある. そこで高速化や低消費電力化を目指したハードウェア実装に関する研究が盛んに行われている．特に FPGA（Field Programmable Gate Array）を用いた実装が注目されている. ハードウェア化において，FPGA などでは演算器ブロックやメモリブロックなどの回路資源が限られている．このため，回路規模やメモリ使用量の削減を目的として，深層学習に用いられる重み係数などのビット数を削減することが有効であり，さまざまな量子化手法が報告されている[2][3][4][5][6][7][8]．画像識別の分野では，重み係数などを二値化，三値化する手法も報告されており, 分類数が少ない Cifar-10 などのデータセットを用いた際, 単精度浮動小数点を用いるよりも認識精度が向上したという報告もある[3][4][5][6]．また，分類数が 1,000 のデータセットである ImageNet を用いた場合でも，重み係数を 4 ビットの固定小数点に量子化して単精度浮動小数点より認識精度が向上したという報告もある[8]．そこで今後用いるネットワークなどにおいて最適な量子化を行うための環境構築を目的に，本研究では，フレームワークの中でも自由度が高いと思われる Chainer [9]を用いて, さまざまな量子化手法を評価できる環境を構築する．さらに，重み係数量子化の一手法を提案し，既存の量子化手法による認識精度と比較評価を行った.

本論文は，以下 7 章で構成する．第 2 章では，機械学習の概要とその手法の 1 つである深層学習について述べる．第 3 章では，既存の量子化手法について概説し，第 4 章では，提案手法について述べる．第 5 章では本実験の評価環境として用いたデータセットやネットワーク構成について述べる．第 6 章では，提案手法と比較対象の既存手法を比較して，評価について述べ，第 7 章では，結論と今後の課題について述べる.

# 第 2 章　深層学習

## 2.1　機械学習

機械学習とは, 人間が自然と行っている認識の仕方や経験則を機械が得るように，機械自身にデータから学習をさせ，データに対するパターンなどを発見させる技術や理論のことである.

機械学習において重要なことは，機械自身にどのようなデータを与えて，学習をさせるかということである. その学習の手法は図 2.1 に示すように大きく分類され，以下に後述する教師あり学習，教師なし学習と中間的手法の 3 つに分かれる[10].

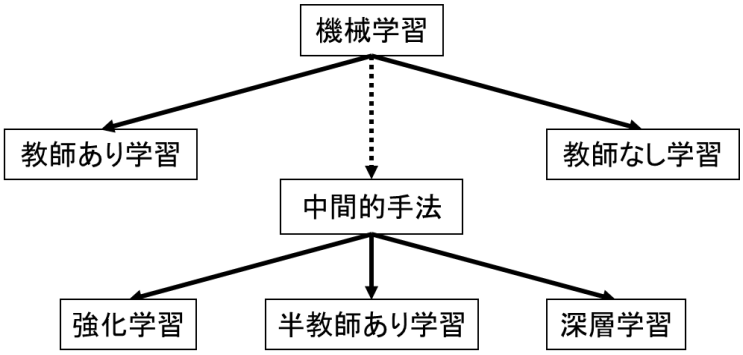



図 2.1　機械学習の分類[13]

教師あり学習では, 学習に用いるデータは図 2.2 に示すように入力データとその入力データが何を表すかを示す正解データからなる．教師あり学習は，既知であるデータを用いて学習し，未知のデータを推測するために使われる．つまり，教師あり学習は主に識別と回帰に用いられる.

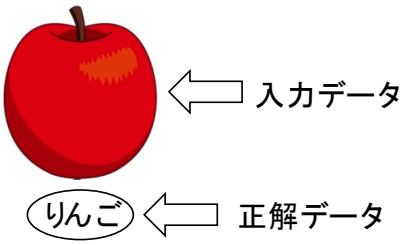



図 2.2　教師あり学習の学習データ

教師なし学習では，学習に用いるデータは入力データのみである．教師なし

学習は，未知のデータから学習し，データの類似性を学習するために使われる．つまり，教師なし学習はクラスタリングや密度推定などに使われる．

中間的手法とは，教師あり学習や教師なし学習に当てはまらない学習の手法である．中間手法としては，半教師あり学習，強化学習や深層学習などがある．半教師あり学習は，学習に用いる入力データが，正解データを持つデータと正解データを持たないデータの両方のデータを含む．強化学習は，ある与えられた状況下に応じて，取った行動に点をつけ，なるべく高い点数を目指す．深層学習は，教師あり学習と教師なし学習のどちらも用いる．

## 2.2 深層学習

深層学習とは，人間や動物の脳神経系を模倣して作られた数学モデルである人工ニューラルネットワークを多層にして行う機械学習のことである．機械学習では，学習に用いるデータから特徴量を人の手で決定する手間が必要だったが，深層学習では，その特徴量の決定すらも学習によって機械が行う．

深層学習では，教師あり学習と教師なし学習の両方を扱う．教師あり学習を用いて行われるものとしては，画像識別などが挙げられる．教師なし学習を用いて行われるものとしては，音声識別などが挙げられる．

以下に深層学習モデルにおけるニューラルネットワークの構造とノードと呼ばれるニューロンを模したモデルについて述べる．

### 2.2.1 ニューラルネットワーク

ニューラルネットワークは，ノードを階層状に並べ，また層毎にノードを多数配置して構成される．図 2.3 に最も基本的な構成として，3 層の階層型ニューラルネットワークを示す．ニューラルネットワークは，入力層，中間層と出力層から構成される．図 2.3 では入力層のノードは 3 つ，中間層のノードは 4 つ，出力層のノードが 3 つで構成される．入力層では入力を受け取り，入力に重みを掛けて，結合している中間層のノードに値を渡す．そして，中間層から出力を行う出力層にも同様に，中間層から出力された値に重みが掛けられ，出力層のノードに値を渡す．ニューラルネットワークの層をさらに多層にするには，中間層の層数を増やしていく．階層型ニューラルネットワークは，このように入力層から出力層へと一方向に伝播される[14].

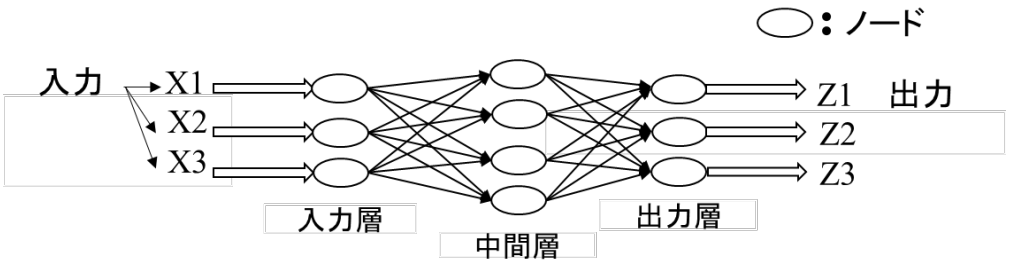



図 2.3　3 層の階層型ニューラルネットワーク

### 2.2.2　ノード

2 入力 1 出力のノードの構造を図 2.4 に示す．入力 X1,X2 に重み係数 W1,W2 を掛け合わせ，総和を取る．そして活性化関数$f(\cdot)$を通して，出力 Z を出す．ノードが行っている処理を式 (2.1) に示す．

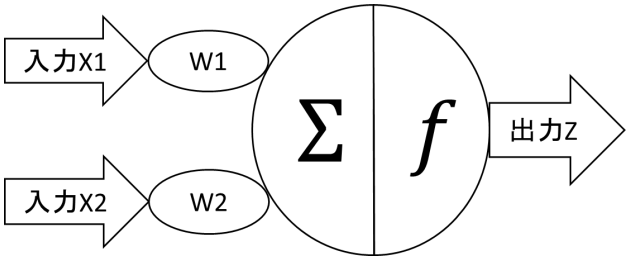



図 2.4　2 入力 1 出力のノードの構造

$$z = f\left(\sum_{i=1}^{L} w_i x_i\right) \qquad \cdots (2.1)$$

ここで，$z$ は出力，$x$ は入力，$w$ は重み係数，$L$ は前層のノード数，$f(\,)$は活性化関数である．活性化関数としては，Sigmoid 関数，Rectified Linear Unit (ReLU) 関数，Sign 関数等がある．

## 2.3　深層学習による画像識別

深層学習は画像識別や音声識別など用途の違いによってノード間の結合の仕方が大きく異なる．そのため，本節では，深層学習の中でも画像識別を行う場

合に多く用いられる Convolutional Neural Network (畳み込みニューラルネットワーク：CNN) と呼ばれるニューラルネットワークの構成を，代表的な構成である LeNet[11]を参考に述べる．

### 2.3.1 CNN の構成

図 2.5 に CNN の代表的な構成である LeNet の層構成を示し，CNN の構成について述べる．CNN は畳込み層，プーリング層と全結合層と呼ばれる三種類の層で構成される．入力はサイズが$N_x \times N_y$の$F$枚（以下，この枚数を特徴マップと呼ぶ）の画像データである．入力画像がグレースケールなら$F = 1$，カラーの場合 RGB で$F = 3$となる．入力を受け取る層以降の入力特徴マップ数は直前の畳込み層の出力特徴マップ数となる．すなわち，入力サイズは$N_x \times N_y \times F$となる．

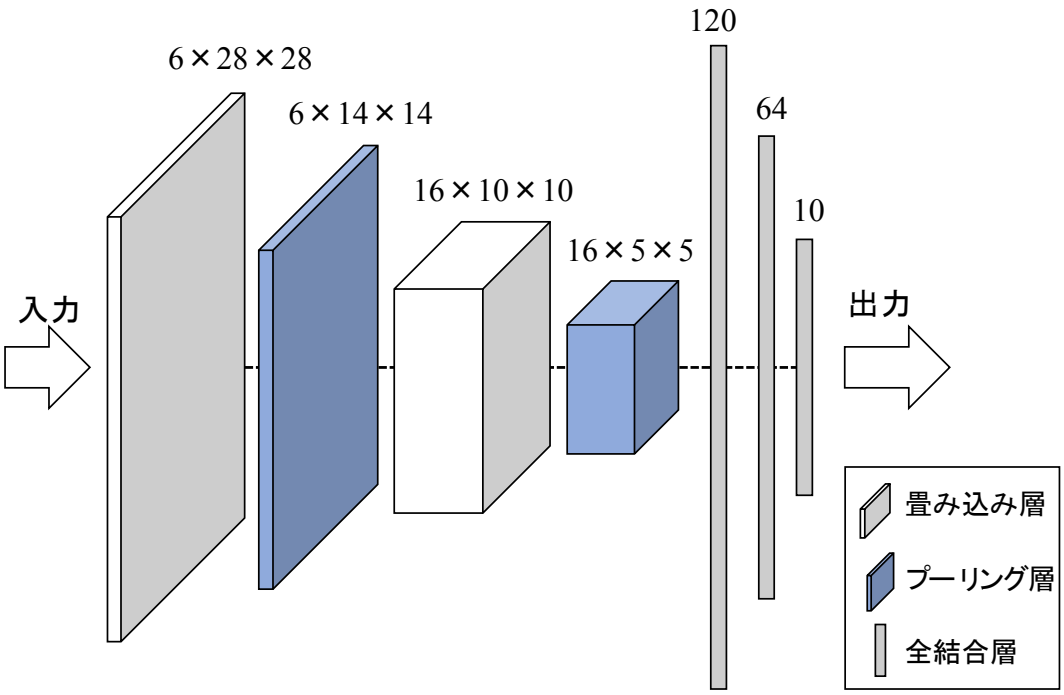



図 2.5　LeNet の構成

### 2.3.2 畳込み層

畳込み層では，入力に対して重み係数の畳込み演算を施す．畳込み処理で用いられる重み係数は学習によって得られる．

畳込み層の基本構造を図 2.6 に示す．畳込み層ではこの入力に重み係数を畳込む．入力サイズが$N_I \times N_I$の各特徴マップに対して$H \times H$のサイズの重み係数$F_w$を

畳込む．畳み込み演算結果は活性化関数$f(\cdot)$を経て，出力マップとして出力される．同様の処理を重み係数の枚数分だけ行うことにより，出力特徴マップ数が重み係数と同じ枚数になる．重み係数の枚数は任意に変更できる．

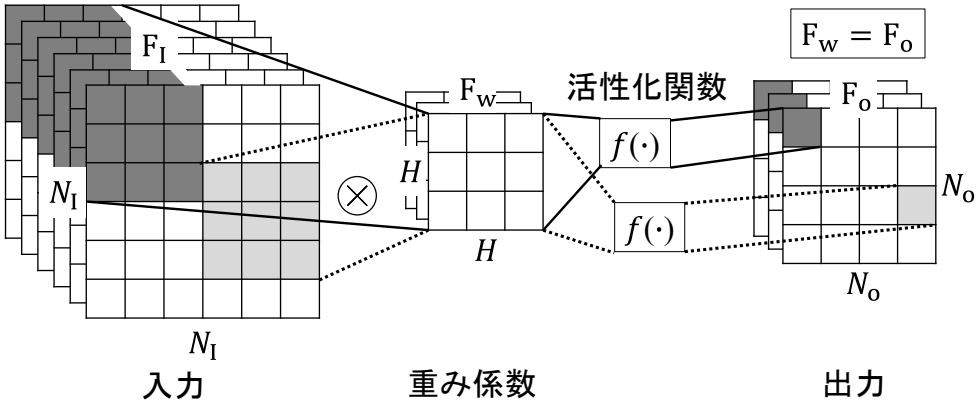



図 2.6　畳込み層の構造

### 2.3.4　プーリング層

プーリング層は，特徴の位置感度を低下させることにより，特徴の微小な位置変化に対する不変性を実現する処理であり，基本的に畳込み層と対で使われる．プーリングは，特徴マップ毎に独立して行われるため，特徴マップ数は変化しない．プーリングの一例として，マックスプーリングとグローバルアベレージプーリングについて述べる．

図 2.7 にマックスプーリングについて示す．マックスプーリングとは，領域内にある値の中で最大の値を出力するプーリングのことである．

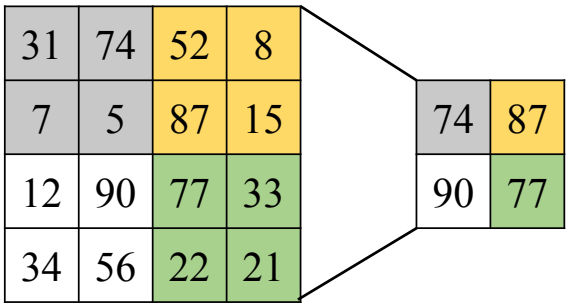



図 2.7　マックスプーリング

図 2.8 にグローバルアベレージプーリングを示す．グローバルアベレージプーリングとは，特徴マップ毎の平均値を出力するプーリングのことである．

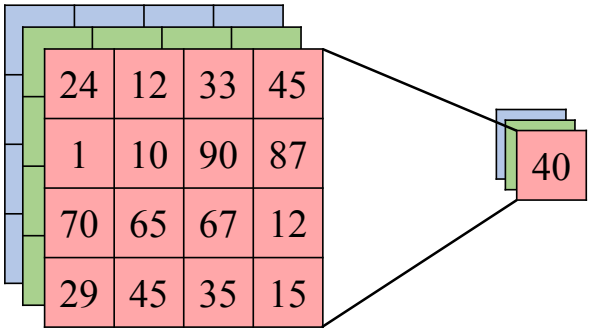



図 2.8　グローバルアベレージプーリング

### 2.3.5　全結合層

全結合層の基本構造を図 2.9 に示す．全ての入力に重み係数を掛け，総和を取り，活性化関数$f(\cdot)$を経た値が出力の 1 要素となる．同様の処理を重み係数の枚数分行うことにより，出力数は重み係数の枚数と等しくなる．

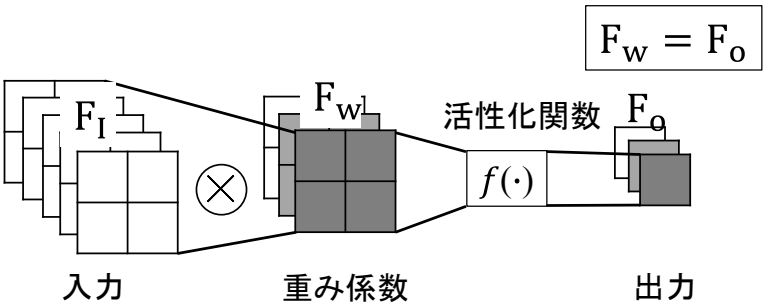



図 2.9　全結合層の構造

### 2.3.6　出力

出力として，多分類の画像認識を行うために，しばしば Softmax 関数を出力直前の全結合層の活性化関数として用いる．Softmax 関数とは，出力を確率にする関数である．Softmax 関数を式 (2.2) に示す．

$$Z_k = \frac{exp\,(u_k)}{\sum_{j=1}^{L} exp\,(u_j)} \qquad \cdots (2.2)$$

ここで，$Z$ は出力，$u$ は入力，L は出力数である．Softmax 関数の出力 Z の総和は常に 1 となる．

# 第 3 章　量子化の既存手法

## 3.1　重み係数の量子化

重み係数の量子化は，単精度浮動小数点による学習処理によって得られた重み係数について，量子化を施すことである．量子化を施す手順については，全ての重み係数を一斉に量子化する方法（以後，本論分では単純量子化と呼ぶ）や，段階的に量子化を進める方法などがある．また，量子化を行う際の計算方法には Sign 関数を用いて量子化するや閾値を用いて量子化するなどの方法がある．

## 3.2　量子化方法

### 3.2.1　単純量子化

単純量子化手法の処理を図 3.1 に示す．単純量子化では，学習処理によって得られた重み係数を一度に量子化する．量子化した重み係数を用いて学習画像の識別を行い，誤差を出力する．その誤差を用いて重み係数を更新する際は，量子化した値ではなく単精度浮動小数点で行う．更新した重み係数を再び量子化し，テスト画像の識別を行う．この一連の処理を認識精度が収束するまで繰り返す．

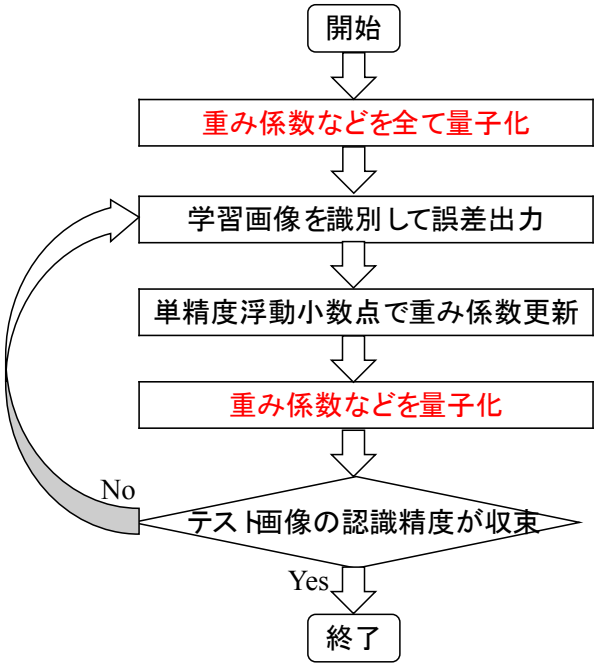



図 3.1　単純量子化処理の流れ

### 3.2.1　インクリメンタル量子化

INQ（Incremental Network Quantization）[8]に用いられている段階的に量子化を行う量子化手法（以後，本論分ではインクリメンタル量子化と呼ぶ）の処理を図 3.2 に示す．インクリメンタル量子化では，学習処理によって得られた重み係数を部分的，段階的に量子化する．まず，学習で得られた重み係数の半数を量子化し，学習画像の識別を行った結果である予測誤差を出力する．この誤差を用いた重み係数の更新は，単精度浮動小数点で行う．これらの処理を認識精度が収束するまで繰り返したあと，未量子化係数の半数を選択，量子化し，再び認識精度が収束するまで係数更新の処理を繰り返す．図 3.3 に示すように，重み係数を量子化した後の値は 2 の乗数または 0 の値を取る．再度認識精度が収束するまで学習を行い，未量子化係数を更新する．認識精度が収束するまで学習を行い，未量子化係数を更新する．認識精度が収束すれば，未量子化係数から半数選択して量子化し，認識精度が収束するまで学習を行う．この一連の処理を INQ では計 4 回繰り返して重み係数を全て量子化している．

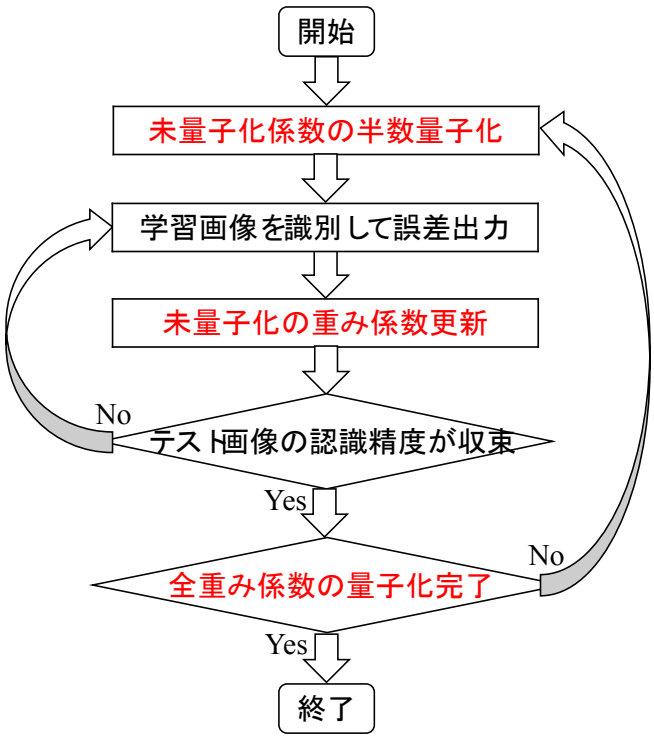



図 3.2　インクリメンタル量子化処理の流れ

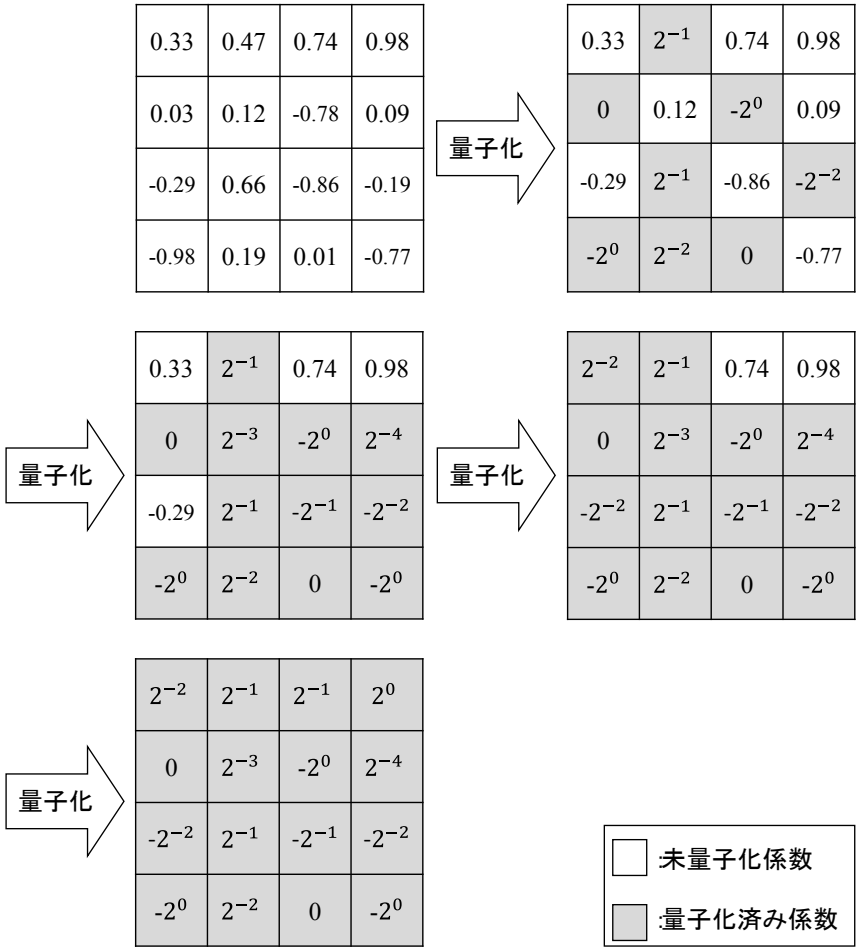



図 3.3　インクリメンタル量子化手法（例：5 ビットに量子化）

## 3.3　量子化計算方法

### 3.3.1　二値化

二値化モデルでは，BNN（Binarized Neural Networks）[3]が単純量子化を用いて量子化している．BNN では，重み係数と入力値を{-1,1}に二値化して識別を行う．BNN では量子化計算方法として Sign 関数を用いており，正の値ならば 1，負の値ならば-1 というように二値化を行う．

### 3.3.2　三値化

三値化モデルとしては，TWN（Ternary Weight Network）[5][6]が単純量子化を用いて量子化している．TWN は，重み係数を{-1,0,1}に三値化して識別を行う．

TWN では量子化計算方法として，-1,0,1 の各値の割合を決めるスケーリング係数と閾値を用いて三値化を行う．TWN[5]（以後，本論分では TWN1 と呼ぶ）ではスケーリング係数は正負で同じ値を用いる．TWN1 を改良した TWN[6] （以後，本論分では TWN2 と呼ぶ）では正負で異なる値を用いる．

### 3.3.3 量子化

様々なビット数に量子化を行っている一例として，INQ の量子化計算方法について述べる．INQ では特別な演算式(3.1)を用いて行っている．ある層の重み係数の絶対値を取り，その絶対値が$2^n$より大きいか確認し，大きければ$2^{n+1}$に量子化をする．そしてその量子化した値を元の重み係数の符号をつける．

$$\widehat{W_l} = \begin{cases} \beta sgn(W_l) & (\alpha+\beta)/2 \le abs(W_l) < 3\beta/2 \\ 0 & \text{otherwise} \end{cases} \quad \cdots (3.1)$$

ここで，$\widehat{W}$は量子化した重み係数，W は重み係数，l は何層目かを表し，$\beta$と$\alpha$は 2 の乗数の値である．

# 第 4 章　提案量子化手法

## 4.1　量子化方法

提案手法の量子化処理を図 4.1 に示す．提案手法では，出力層に近い層の重み係数が入力層に近い重み係数より重要であるという考えに基づいた．入力層を量子化しても出力層に近い層で学習による調整が利くように，入力に近い層から順に量子化を行っていく．INQ における量子化と同様に，まず単精度浮動小数点で学習を行い，重み係数を得る．図 4.2 に示すように，重み係数を入力に近い層から順に量子化と学習を繰り返し，全ての層を量子化するまで実行する．

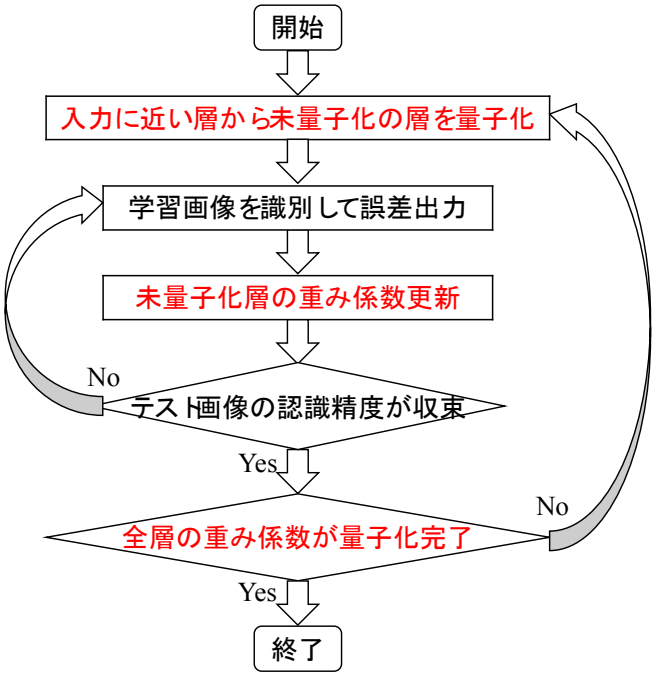



図 4.1　提案量子化手法の処理の流れ

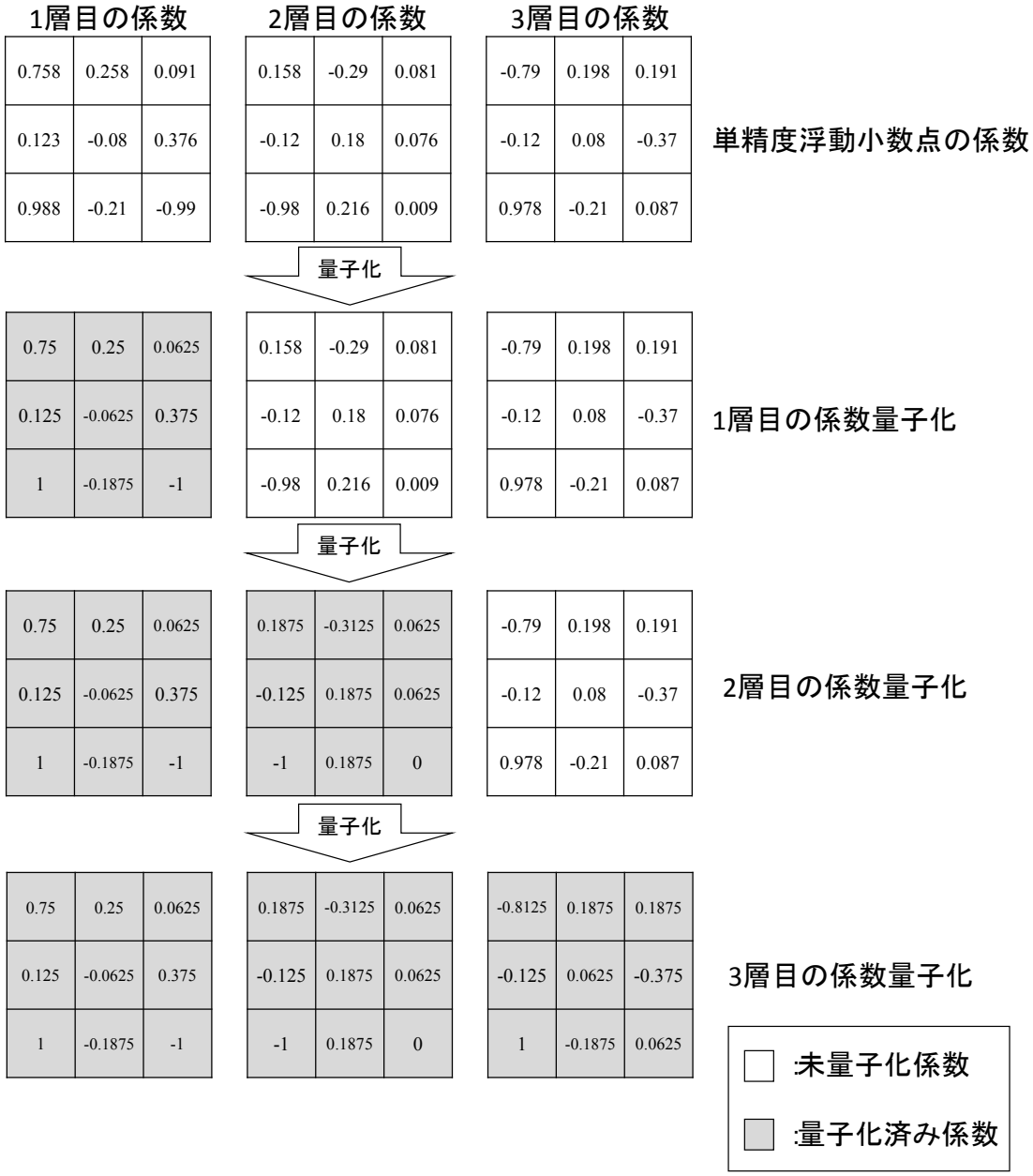



図 4.2　提案手法での量子化手法（例：5 ビットに量子化）

## 4.2　量子化計算方法

提案手法の 5 ビットに量子化するときの量子化計算方法を図 4.3 に示す. 重み係数を固定小数点の指定するビット数の次のビットを確認し，その値が 1 であれば，切り上げを行い，固定小数点の指定するビット数とする.

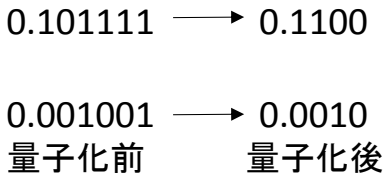



図 4.3　提案手法の量子化計算方法（例：5 ビットに量子化）

# 第 5 章　評価環境

## 5.1　深層学習のオープンソースフレームワーク

深層学習の代表的なオープンソースフレームワークを表 5.1 に示す. 本研究を開始した当時, 自由度が最も高いフレームワークのひとつが Chainer であったため，Chainer v1.24.0 を用いた．現在では，TensorFlow[14]など様々なオープンソースフレームワークが公開されており，Chainer の他にも自由度が高いフレームワークが存在する.

表 5.1　深層学習の代表的なフレームワーク

| フレームワーク名 | 開発元 | 公開月 |
|---|---|---|
| Caffe[15] | UC Berkeley | 2013.1 |
| Torch7[16] | Facebook | 2015.1 |
| Chainer[9] | PFN | 2015.6 |
| TensorFlow[14] | Google | 2015.11 |
| Pytorch[17] | Facebook | 2016.1 |
| Caffe2[18] | Facebook | 2017.4 |

## 5.2　ネットワーク構成

BNN，TWN1，TWN2，INQ を提案手法の量子化手順と比較対象にするためそれぞれで用いられているネットワーク構成について述べる．この節では，単純量子化を用いて, 二値化を行っている BNN から順に, 三値化を行っている TWN1, TWN2，インクリメンタル量子化を行っている INQ に用いたネットワーク構成を述べていく.

### 5.2.1　VGG-9

提案手法と BNN を比較するため, 図 5.1 に層構成を示す VGG-9 と呼ばれるネットワーク構成を対象とする．表 5.2 にパラメータ構成を示す．VGG-9 では畳込み層 6 層，プーリング層 3 層，全結合層 3 層で構成される．また VGG[12]と呼ばれるネットワーク構成では, 畳込み層で用いる重み係数のサイズは全て3×3と設定している．BNN では活性化関数として Sign 関数を用いているが，本研究で用いた Chainer v1.24.0 は Sign 関数が実装されていなかったため，代わりに ReLU 関数を用いた.

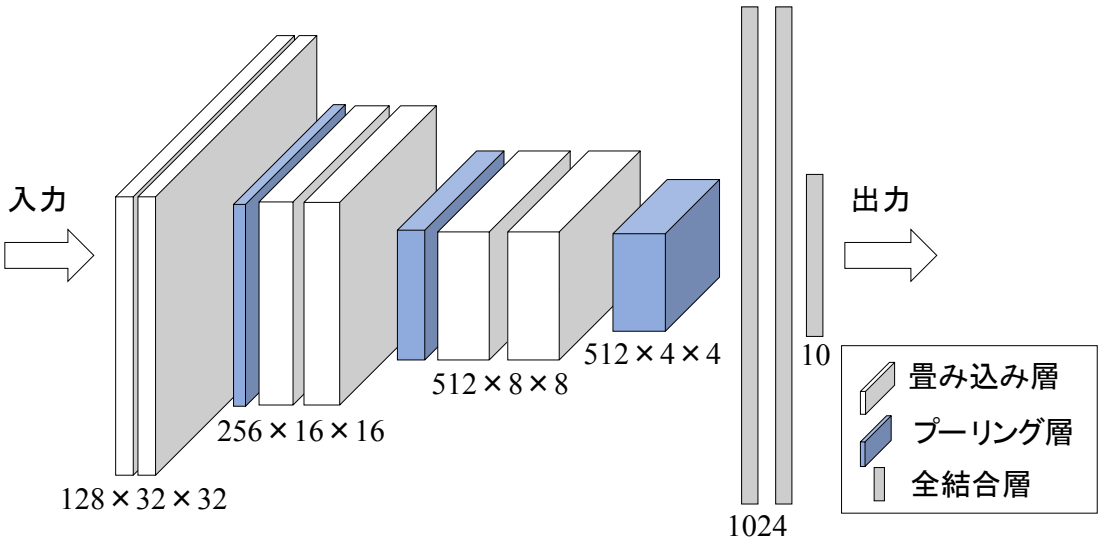



図 5.1　VGG-9 の層構成

表 5.2　VGG-9 の層のパラメータ構成

| 層 | 特徴マップ数 | | 出力サイズ |
|---|---|---|---|
| | 入力 | 出力 | |
| 畳込み1 | 3 | 128 | 32×32 |
| 畳込み2 | 128 | 128 | 32×32 |
| マックスプーリング | 128 | 128 | 16×16 |
| 畳込み3 | 128 | 256 | 16×16 |
| 畳込み4 | 256 | 256 | 16×16 |
| マックスプーリング | 256 | 256 | 8×8 |
| 畳込み5 | 256 | 512 | 8×8 |
| 畳込み6 | 512 | 512 | 8×8 |
| マックスプーリング | 512 | 512 | 4×4 |
| 全結合1 | 8192 | 1024 | 1 |
| 全結合2 | 1024 | 1024 | 1 |
| 全結合3 | 1024 | 10 | 1 |

### 5.2.2　VGG-8

提案手法と TWN1 との比較のため，図 5.2 に層構成を示す VGG-8 呼ばれるネットワーク構成を対象とする. 表 5.3 に VGG-8 のパラメータ構成を示す. VGG-8 では畳込み層 6 層，全結合層 2 層から構成される．活性化関数としては ReLU 関数を用いている.

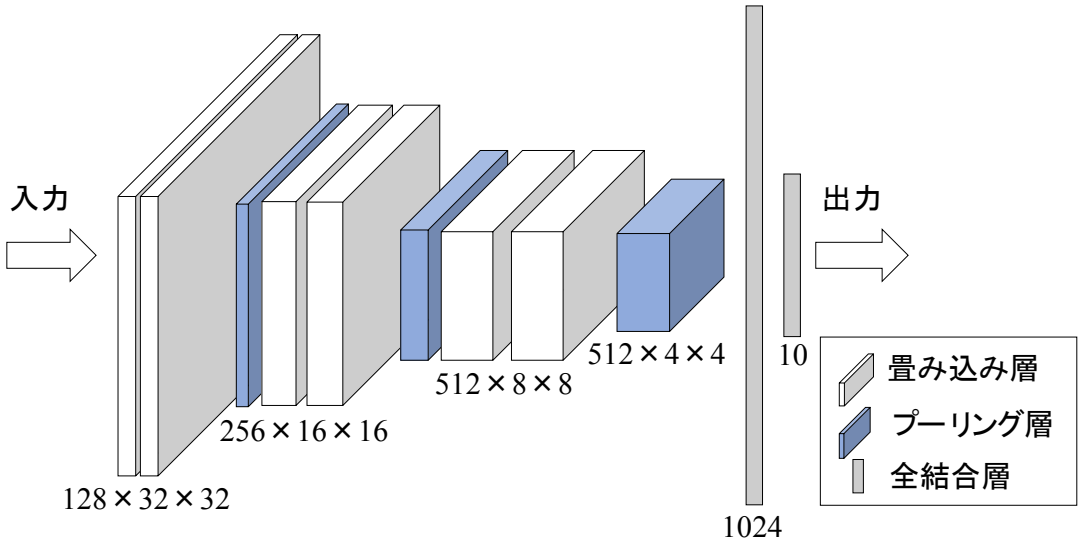



図 5.2　VGG-8 の層構成

表 5.3　VGG-8 の層のパラメータ構成

| 層 | 特徴マップ数 | | 出力サイズ |
|---|---|---|---|
| | 入力 | 出力 | |
| 畳込み1 | 3 | 128 | 32×32 |
| 畳込み2 | 128 | 128 | 32×32 |
| マックスプーリング | 128 | 128 | 16×16 |
| 畳込み3 | 128 | 256 | 16×16 |
| 畳込み4 | 256 | 256 | 16×16 |
| マックスプーリング | 256 | 256 | 8×8 |
| 畳込み5 | 256 | 512 | 8×8 |
| 畳込み6 | 512 | 512 | 8×8 |
| マックスプーリング | 512 | 512 | 4×4 |
| 全結合1 | 8192 | 1024 | 1 |
| 全結合2 | 1024 | 10 | 1 |

### 5.2.3　ResNet-18

提案手法と TWN1，TWN2 および INQ との比較のため，ResNet-18 と呼ばれるネットワーク構成を対象とする．

ResNet[13]は，ニューラルネットワークを多層にしたときに発生する勾配消失などの学習が行えなくなる問題を解決するために開発されたネットワーク構成である．ResNet では図 5.3 に示す Residual unit という特徴的な構成がある．Residual unit では，畳込み層を 2 層通る値と，畳込み層 2 層を通らずにショートカットを行う値を足し合わせ，出力とする．しかし畳込み層を通らずにショー

トカットを行う値と畳込み層を通る値では，特徴マップ数が合わない場合がある．そのために用いる方法は図 5.4 に示す(a)Zero-padding，(b)Projection，(c)Bottleneck の三種類ある．Zero-padding では足りない特徴マップ数の値を全て 0 にして出力に付け加える方法である．Projection ではショートカットを行う部分に, 畳込み層を 1 層入れることにより特徴マップ数を同数にする方法である．Bottleneck では畳込み層を 2 層のところを 3 層にして，三層目の畳み込み層で特徴マップ数を調整する方法である．本研究では Projection を用いる．

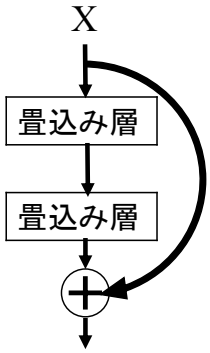
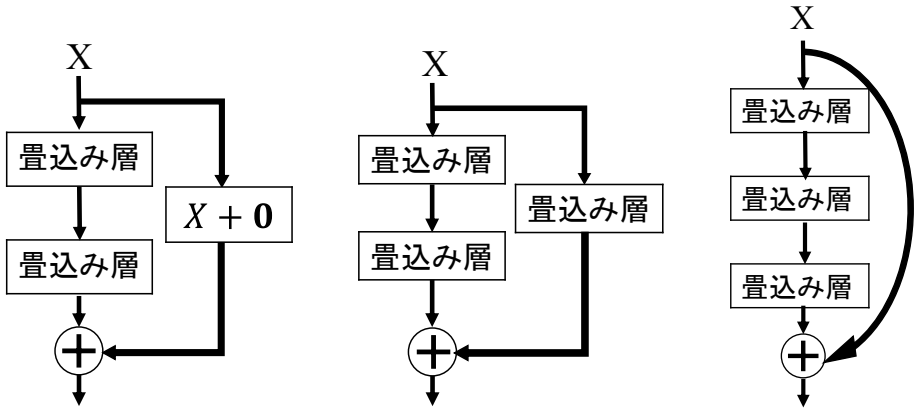



図 5.3　Residual unit の構成




(a)Zero-padding　　(b)Projection　　(c)Bottleneck

図 5.4　Residual unit の特徴マップに対する三種類の調整方法

本研究で用いた ResNet-18 は活性化関数としては ReLU 関数を用いている．表 5.4 に ResNet-18 のパラメータ構成を示す．カーネルサイズは ResNet-18 の 1 層目の畳込み層のみ7×7とし，他のカーネルサイズは全て3×3とした．

表 5.4　ResNet-18 の層構成

| 層 | 特徴マップ数 | | 出力サイズ |
|---|---|---|---|
| | 入力 | 出力 | |
| 畳込み | 3 | 64 | 112×112 |
| マックスプーリング | 64 | 64 | 56×56 |
| Residual unit1 | 64 | 64 | 56×56 |
| Residual unit2 | 64 | 64 | 56×56 |
| Residual unit3 | 64 | 128 | 28×28 |
| Residual unit4 | 128 | 128 | 28×28 |
| Residual unit5 | 128 | 256 | 14×14 |
| Residual unit6 | 256 | 256 | 14×14 |
| Residual unit7 | 256 | 512 | 7×7 |
| Residual unit8 | 512 | 512 | 7×7 |
| グローバル<br>アベレージプーリング | 512 | 512 | 1 |
| 全結合 | 512 | 1000 | 1 |

## 5.3　画像データセット

画像データセットとは，学習に用いる学習画像と認識精度の測定に用いるテスト画像で構成され，本研究では，表 5.5 に示す 2 種類の画像データセットを用いた．Cifar-10 は一般画像認識用のデータセットであり，カテゴリ数が少なく，比較的小規模なデータセットである．一方，ImageNet は Cifar-10 と同じく一般画像認識用のデータセットであるが，Cifar-10 と比較してカテゴリ数が多いため，より高い識別能力が求められる．

表 5.5　画像データセット

| 画像データセット名 | 学習画像 | テスト画像 | 画像サイズ | カテゴリ数 |
|---|---|---|---|---|
| Cifar-10 | 5 万枚 | 1 万枚 | 32×32 | 10 |
| ImageNet | 約 122 万枚 | 約 6 万枚 | 224×224 | 1,000 |

画像データセットの Cifar-10 の画像例を図 5.5 に示す．Cifar-10 は飛行機，自動車，鳥，猫，鹿，犬，蛙，馬，船，トラックのカテゴリ数 10 からなる．各カテゴリに学習画像として 5,000 枚，テスト画像として 1,000 枚ある．


図 5.5　Cifar-10 の画像例

ImageNet の画像例を図 5.6 に示す．ImageNet はテンチ，ホホジロザメ，金魚，スティングレー，おんどり，めんどり，ダチョウ，ユキヒメドリ，シロソウメンタケ，トイレットペーパーなどのカテゴリ数が 1,000 からなる．ImageNet の画像サイズは画像毎に異なるため，画像を拡大縮小し，画像サイズを224×224ピクセルに統一した．


図 5.6　ImageNet の画像例

## 5.4　パラメータ構成

評価実験において，ネットワーク構成を VGG-8，VGG-9 とするときのパラメ

ータ構成は表 5.6 に示すようにした．量子化前の単精度浮動小数点での学習時のパラメータ構成は，TWN1 の VGG-8 に関するパラメータ構成と同様にした．量子化時のバッチサイズ，重み減衰，モーメンタムの値は量子化前の学習時と同じ値を設定した．

表 5.6　VGG-8 と VGG-9 におけるパラメータ構成

| パラメータ | 学習時 | 量子化時 |
|---|---|---|
| epoch数 | 160 | 20 |
| 初期学習率 | 0.1 | 0.5 |
| 学習率減衰のepoch数 | 80及び120 | 5毎 |
| 学習率の減衰率 | 10% | 50% |
| バッチサイズ | 100 | 100 |
| 重み減衰 | 0.0001 | 0.0001 |
| モーメンタム | 0.9 | 0.9 |

ネットワーク構成を ResNet-18 とするときのパラメータ構成は表 5.7 に示すようにした．量子化前の単精度浮動小数点で学習を行うときの epoch 数は，TWN1 で認識精度が収束するまでの epoch 数とした．epoch 数以外の量子化を行う前の学習時のパラメータ構成は，TWN1 と同様にした．量子化時のバッチサイズ，重み減衰，モーメンタムの値は量子化前の値に設定した．

表 5.7　ResNet-18 におけるパラメータ構成

| パラメータ | 学習時 | 量子化時 |
|---|---|---|
| epoch数 | 35 | 2 |
| 初期学習率 | 0.1 | 0.25 |
| 学習率減衰のepoch数 | 30 | 1毎 |
| 学習率の減衰率 | 10% | 25% |
| バッチサイズ | 64 | 64 |
| 重み減衰 | 0.0001 | 0.0001 |
| モーメンタム | 0.9 | 0.9 |

## 5.5　評価項目

提案量子化手法と既存手法について認識精度を用いて比較評価する．既存の量子化手法による認識精度は，文献で報告されている値を採用する．量子化前

の認識精度は，使用プラットフォームによって異なるため，評価指標として量子化前の認識精度と量子化後の認識精度の差を用いる．

# 第 6 章　評価結果

## 6.1　単純量子化

### 6.1.1　BNN

BNN と提案手法の認識精度を表 6.1 に示す．BNN では二値化後の認識精度は二値化前よりもわずかに良くなっている．しかし，提案手法では量子化後の認識精度は量子化前と比べ，係数のビット数が 5 ビットでは約 1%，4 ビットでは約 3%低下している．BNN と比べて，提案手法の結果が悪くなっている．

表 6.1　BNN と提案手法との認識精度の比較（VGG-9，Cifar-10）

| 手法 | 重み係数 | 認識精度 | 量子化前との差 |
|---|---|---|---|
| BNN | 32bit | 88.60% | |
| | 1bit | 88.68% | 0.08% |
| 提案手法 | 32bit | 85.86% | |
| | 5bit | 84.93% | -0.93% |
| | 4bit | 82.78% | -3.08% |

提案手法では活性化関数として Sign 関数ではなく ReLU 関数を用いたことが原因の一つとして考えられる．ReLU 関数を用いることにより，負の値が消されるため，取りうる値が少なくなってしまう可能性がある．

### 6.1.2　TWN

VGG-8 のネットワーク構成を用いたときの TWN1 と提案手法の認識精度を表 6.2 に示す．TWN1 では三値化後の認識精度は三値化前の認識精度よりもわずかに低下している．提案手法では，量子化前後の認識精度の差は，TWN1 よりも低下している．提案手法の量子化後の認識精度は量子化前と比べ，係数のビット数を 5 ビットにしたとき約 2%，4 ビットにしたとき約 4%低下した．

表 6.2　TWN1 と提案手法との認識精度の比較（VGG-8，Cifar-10）

| 手法 | 重み係数 | 認識精度 | 量子化前との差 |
|---|---|---|---|
| TWN1 | 32bit | 92.88% | |
| | 2bit | 92.56% | -0.32% |
| 提案手法 | 32bit | 88.05% | |
| | 5bit | 86.02% | -2.03% |
| | 4bit | 84.42% | -3.63% |

ResNet-18 のネットワーク構成を用いたときの TWN1，TWN2 と提案手法の認識精度を表 6.3 に示す. 三値化後の認識精度は, TWN1 では量子化前より低下し，TWN2 では量子化前よりわずかに向上している．一方，提案手法による量子化後の認識精度は, 量子化前と比べて係数のビット数を 5 ビットにしたとき約 10%も低下しており，TWN1 および TWN2 に対して提案手法による量子化では認識精度の低下が大きい.

TWN1 および TWN2 と比較して提案手法の認識精度が低い原因のひとつとして，提案量子化手法では，量子化ビット数に対して小さい値の場合は単純な切捨てになってしまうため，ある一定以下の値が 0 になるケースが多かった可能性がある．一方で，TWN1 および TWN2 ではスケーリング係数を用いて 0 の数を適当な割合に設定するため，良い結果につながったと考えられる.

表 6.3　TWN1，TWN2 と提案手法との認識精度の比較（ResNet-18, ImageNet）

| 手法 | 重み係数 | 認識精度 | 量子化前との差 |
|---|---|---|---|
| TWN1 | 32bit | 65.40% | |
| | 2bit | 61.80% | -3.60% |
| TWN2 | 32bit | 57.20% | |
| | 2bit | 57.50% | 0.30% |
| 提案手法 | 32bit | 61.92% | |
| | 5bit | 52.40% | -9.52% |

## 6.2　インクリメンタル量子化

INQ と提案手法の認識精度を表 6.4 に示す. INQ では係数を 2 ビットまで量子化した場合は認識精度が低下しているが, 3 ビット以上であれば量子化前と同等以上の認識精度となった．一方で，提案手法により 5 ビットまで量子化した場合の認識精度は，量子化前と比べて約 10%も低下した．この原因のひとつとして，提案手法では量子化は単純な切り上げで行っていることが考えられる．一方で，INQ における量子化演算は，特別な演算式(6.1)で行われている.

$$\widehat{W_l} = \begin{cases} \beta sgn(W_l) & (\alpha+\beta)/2 \le abs(W_l) < 3\beta/2 \\ 0 & \text{otherwise} \end{cases} \quad \cdots (6.1)$$

表 6.4　INQ と提案手法との認識精度の比較（ResNet-18，ImageNet）

| 手法 | 重み係数 | 認識精度 | 量子化前との差 |
|---|---|---|---|
| INQ | 32bit | 68.27% | |
| | 5bit | 68.98% | 0.72% |
| | 4bit | 68.89% | 0.63% |
| | 3bit | 68.08% | -0.19% |
| | 2bit | 66.02% | -2.25% |
| 提案手法 | 32bit | 61.92% | |
| | 5bit | 52.40% | -9.52% |

# 第 7 章　まとめ

畳み込みニューラルネットワーク向け重み量子化のための評価環境を Chainer で構築し，量子化手法について既存手法との比較評価を行った．ネットワーク構成として VGG-8, VGG-9, ResNet-18, データセットとして Cifar-10 と ImageNet を用い，単純量子化としては二値化と三値化，さらにインクリメンタル量子化との比較評価を行った．実験結果より，従来手法では数ビットまで量子化しても単精度浮動小数点と比較して，同等以上の認識精度を実現できている．しかし，提案量子化手法では同等以下もしくは大きく低下させる結果となった．その原因として，量子化を行う手順よりも量子化の演算方法や，活性化関数も含めた学習パラメータ設定に大きく依存することが考えられる．

本研究では自由度の高さを根拠に Chainer v1.24.0 を元に評価環境を構築したが，バージョンは最新のものではない．また現在は Chainer 以外にも様々なフレームワークが存在しており，より自由度が高く扱いやすいフレームワークもあると考えられるので，フレームワーク選択の見直しも行う必要がある．

# 謝辞

本研究を進めるにあたり，ご指導を頂きました高知工科大学システム工学群電子・光システム工学専攻　密山幸男准教授に心より感謝致します．また，ご助言を頂くとともに日頃からお世話になりました．高知工科大学システム工学群電子・光システム工学専攻　橘昌良教授に深く感謝致します．副査をしていただきました高知工科大学システム工学群電子・光システム工学専攻　星野孝総准教授に深く感謝致します．

橘・密山研究室の皆様には，日頃から様々な意見を頂き，精神的にも支えられました．心から感謝致します．

# 参考文献


[1] G.E. Dahl, N. Jaitly, R. Salakhutdinov, “Multi-task Neural Networks for QSAR Predictions”, arXiv preprint arXiv:1406.1231, Jun. 2014.

[2] H. Alember, V. Leroy, A.P. Boucle, F. Pétrot, “Ternary Neural Networks for Resource-Efficient AI Applications”, arXiv preprint arXiv:1609.00222, Feb. 2017.

[3] R.Zhao, W.Song, W.Zhang ,T.Xing, J.Lin, M.Srivastava, R.Gupta, Z.Zhang “Accelerating Binarized Convolutional Neural Networks with Software-programmable FPGAs”, Proc. International Symposium on Field-Programmable Gate Arrays (ISFPGA), Feb. 2017.

[4] M. Courbariaux, Y. Bengio, J.P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations”, Proc. Neural Information Processing Systems (NIPS), Dec. 2015.

[5] F. Li, B. Zhang, B. Liu, “Ternary weight networks”, Proc. Neural Information Processing Systems (NIPS), Dec. 2016.

[6] C. Zhu, S. Han, H. Mao, W.J. Dally, “Trained Ternary Quantization”, Proc. International Conference on Learning Representations (ICLR), Apr. 2017.

[7] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, “DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients”, arXiv preprint arXiv: 1606.06160, Jun. 2016.

[8] A. Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen, “Incremental Networks Quantization:Towards Lossless CNNs with Low-Precision Wights”, Proc. International Conference on Learning Representations (ICLR), Apr. 2017.

[9] S. Tokui, K. Ono, S. Hido, J. Clayton, “Chainer: a next-generation open source framework for deep learning”, Proc. Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), Dec. 2015.

[10] 荒木雅弘，“フリーソフトではじめる機械学習入門”森北出版社，2015 年

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, Proc. The Institute of Electrical and Electronics Engineers (IEEE), Nov. 1998.

[12] K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, Proc. International Conference on Learning Representations (ICLR), May. 2015.

[13] K. He, X. Zhang, S. Ren, J. Sun. “Deep residual learning for image recognition”, Proc. Computer Vision and Pattern Recognition (CVPR), Jun. 2016.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man´e, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vi´egas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems", arXiv:1603.04467, Mar. 2016.

[15] Y. Jia, "Caffe: An open source convolutional architecture for fast feature embedding", Berkeley Artificial Intelligence Research, http://caffe.berkeleyvision.org, Jan. 2018.

[16] R. Collobert, K. Kavukcuoglu, C. Farabet, "Torch7: A matlab-like environment for machine learning", Proc. Neural Information Processing Systems (NIPS), Dec. 2011.

[17] "PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration.", Pytorch core team, http://pytorch.org, Feb. 2018.

[18] "Caffe2: A New Lightweight, Modular, and Scalabele Deep Learning Framework", Facebook, https://caffe2.ai, Feb. 2018.