Vol.2018-ARC-230 No.37
Vol.2018-SLDM-183 No.37
Vol.2018-EMB-47 No.37
2018/3/8

畳込みニューラルネットワーク向け重み量子化手法の検討

氏原 収悟[1]　密山 幸男[1]

概要：画像識別などに多く用いられる深層学習の畳込みニューラルネットワーク（CNN）のハードウェア実装において、回路規模の削減やメモリ使用量の削減を目的として、重み係数や演算データの量子化に関する研究が注目されている。ニューラルネットワークの構成やデータセットによっては、数ビット程度まで量子化するにもかかわらず、単精度浮動小数点演算による処理と同程度の認識精度が得られたという報告もある。一方で、重み係数の量子化は、学習パラメータや量子化の手順などによって認識精度が大きく変わるため、それらの最適化は容易ではない。そこで本研究では、オープンソースフレームワークのひとつである Chainer を用いて様々な量子化手法を評価できる環境を構築した。また、重み係数量子化の一手法を提案し、従来手法による認識精度と比較評価を行った。

キーワード：深層学習、ニューラルネットワーク、畳み込み、CNN、量子化、Chainer

1. はじめに

深層学習のひとつである畳込みニューラルネットワーク（CNN: Convolutional Neural Network）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、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、さまざま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が、より高い認識精度を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り、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おり、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、特に 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ため、回路規模やメモリ使用量を削減するために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]-[8]。画像識別の分野では、係数などを二値化、三値化する手法も提案されており、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]-[6]。また、分類数 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]。そこで本研究では、オープンソースフレームワークのなかでも自由度が高いと言われている 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した。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。

[1] 高知工科大学
Kochi University of Technology

2. CNN の概要

CNN は多層ニューラルネットワークの一つである。入力画像に対して畳込み処理を行うことで特徴量を抽出し、これを繰り返すことで識別を行うことができる。畳み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。

CNN の代表的な構成である LeNet の層構成を図 1 に示す。CNN は、畳込み層、プーリング層、全結合層と呼ばれる 3 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数と呼ぶ）の画像データである。入力画像がグレーケールの場合は F=1、カラーの場合は RGB で F=3 となる。入力を受け取る層以降の特徴マップ数は、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。

畳込み層では、入力に対して重み係数の畳み込み演算を施す。畳み込み層の基本構造を図 2 に示す。入力サイズが $N_x \times N_y$ の各特徴マップに対して $H \times H$ のサイズの重み係数を畳み込む。畳み込み演算結果は、活性化関数 f を経て、出力特徴マップとして出力される。同様の処理を重み

Vol.2018-ARC-230 No.37
Vol.2018-SLDM-183 No.37
Vol.2018-EMB-47 No.37
2018/3/8

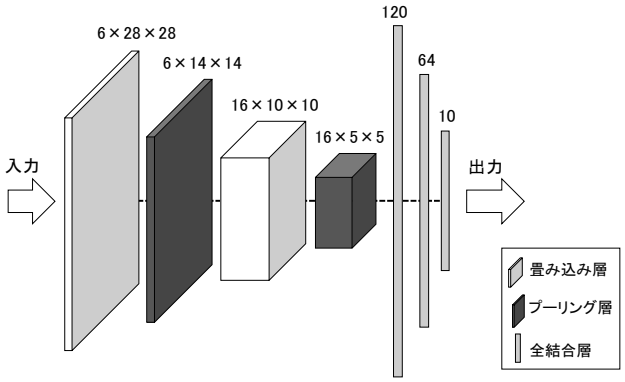

図 1 LeNet の構成

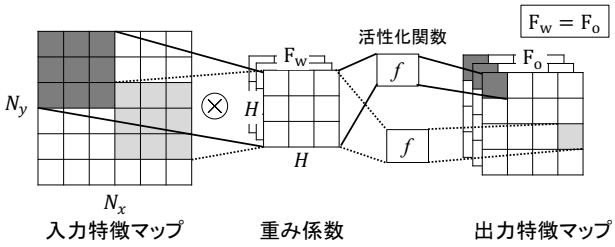

図 2 畳み込み層の構造

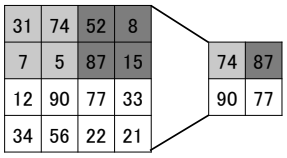

図 3 Max Pooling の処理

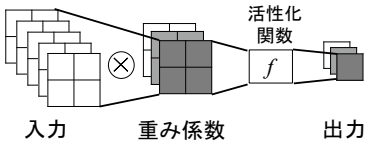

図 4 全結合層の構造

係数の枚数だけ行ことにより、出力特徴マップは重み係数と同じ枚数になる。

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、基本的には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。プーリングのひとつである Max Pooling を図 3 に示す。Max Pooling は、ある領域内の値の中で最大の値を出力する。

全結合層の基本構造を図 4 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 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。

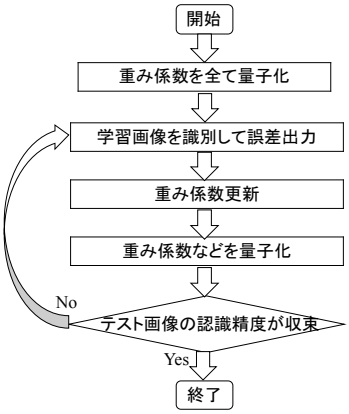

図 5 単純量子化処理の流れ

3. 既存の量子化手法

3.1 重み係数の量子化

重み係数の量子化は、まず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、本稿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。

二値化モデルでは、BNN（Binarized Newral Networks）[3] が単純量子化を採用している。BNN は、重み係数と入力値を {-1, 1} に二値化して推論を行う。BNN では活性化関数に 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。また、三値化モデルとしては、TWN（Ternary Weight Network）[5][6] が単純量子化を採用している。TWN は、重み係数を {-1, 0, 1} に三値化して推論を行うものであり、-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、本稿では 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、本稿では TWN2 と呼ぶ）では正負で異なる値を用いる。

3.2 単純量子化

単純量子化手法を図 5 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。

3.3 インクリメンタル量子化

INQ（Incremental Network Quantization）[8] に用いられている量子化手法（以下、本稿ではインクリメンタル量子化と呼ぶ）の流れを図 6 に示す。インクリメンタル量子化

Vol.2018-ARC-230 No.37
Vol.2018-SLDM-183 No.37
Vol.2018-EMB-47 No.37
2018/3/8

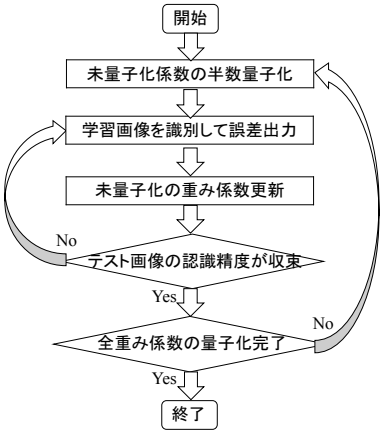

図 6 インクリメンタル量子化処理の流れ

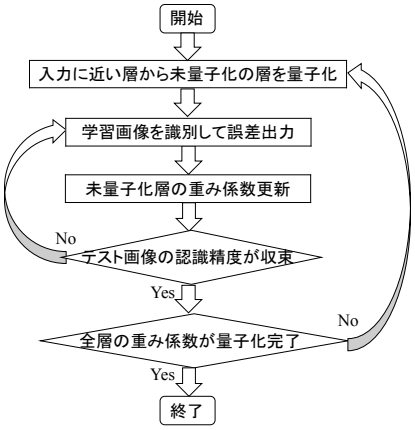

図 8 提案量子化処理の流れ

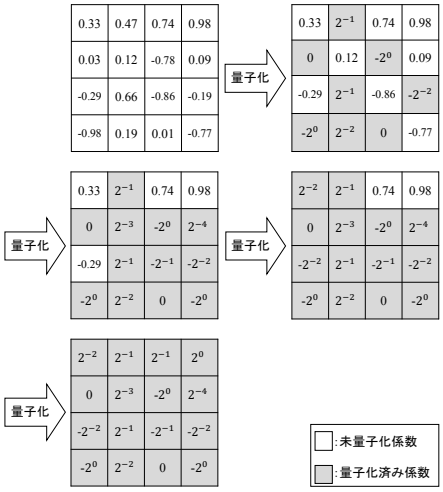

図 7 インクリメンタル量子化手法（例：5 ビットに量子化）

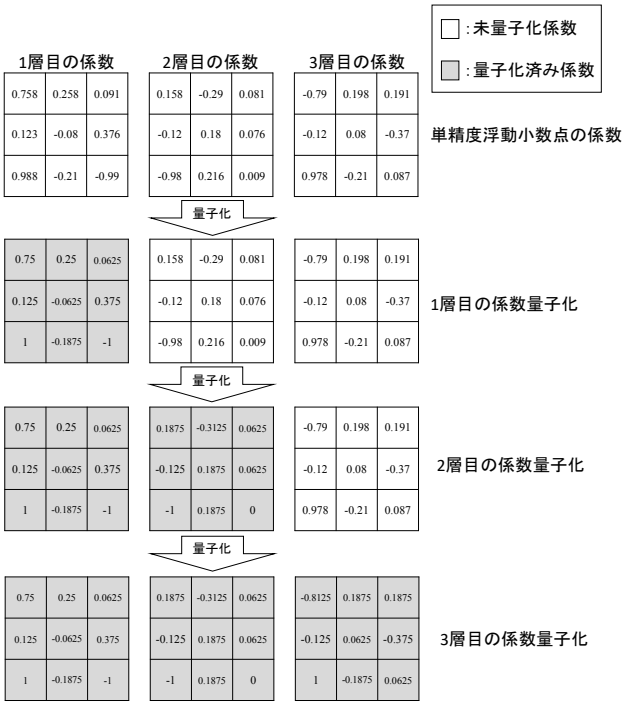

図 9 提案量子化手法（例：5 ビットに量子化）

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、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用いてテスト画像の識別を行う。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ランダムに選択、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 7 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。認識精度が収束すれば、未量子化係数からランダムに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。

4. 提案量子化手法

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ものである。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、入力の方から順に量子化を行っていく。提案手法の量子化処理を図 8 に示す。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 9 に示すように、重み係数を入力に近い方から順に量子化と学習を交互に繰り返し、全ての層を量子化するまで実行する。

5. 評価環境

5.1 オープンソースフレームワーク

本稿ではフレームワークとして Chainer ver1.24.0 を用いた。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 であったことがその選択理由である。現在では、TensorFlow[10] などさまざま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多数存在する。

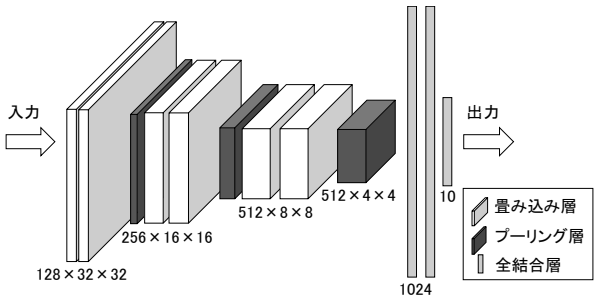

図 10　VGG-9 の層構成

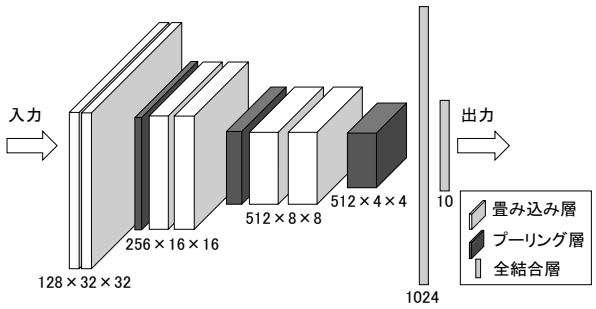

図 11　VGG-8 の層構成

表 1　VGG-9 のパラメータ構成

層	特徴マップ数		出力
	入力	出力	サイズ
Conv1	3	128	32 × 32
Conv2	128	128	32 × 32
Max Pool	128	128	16 × 16
Conv3	128	256	16 × 16
Conv4	256	256	16 × 16
Max Pool	256	256	8 × 8
Conv5	256	512	8 × 8
Conv6	512	512	8 × 8
Max Pool	512	512	4 × 4
FC1	8,192	1,024	1
FC2	1,024	1,024	1
Output	1,024	10	1

表 2　VGG-8 のパラメータ構成

層	特徴マップ数		出力
	入力	出力	サイズ
Conv1	3	128	32 × 32
Conv2	128	128	32 × 32
Max Pool	128	128	16 × 16
Conv3	128	256	16 × 16
Conv4	256	256	16 × 16
Max Pool	256	256	8 × 8
Conv5	256	512	8 × 8
Conv6	512	512	8 × 8
Max Pool	512	512	4 × 4
FC	8,192	1,024	1
Output	1,024	10	1

表 3　ResNet-18 のパラメータ構成

層	特徴マップ数		出力
	入力	出力	サイズ
Conv1	3	64	112 × 112
Max Pool	64	64	56 × 56
Residual unit1	64	64	56 × 56
Residual unit2	64	64	56 × 56
Residual unit3	64	128	56 × 56
Residual unit4	128	128	56 × 56
Residual unit5	128	256	56 × 56
Residual unit6	256	256	56 × 56
Residual unit7	256	512	56 × 56
Residual unit8	512	512	56 × 56
Global Average Pooling	512	512	1
Output	512	1,000	1

5.2　ネットワーク構成

BNN, TWN1, TWN2, INQ を提案手法の比較対象とするため、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。BNN では VGG-9、TWN-1 では VGG-8、TWN2 と INQ では ResNet-18 が用いられている。以下では、これら 3 つのネットワーク構成について概説する。

5.2.1　VGG-9

提案手法と BNN を比較するため、VGG-9 と呼ばれるネットワーク構成を対象とする。VGG-9 の層構成を図 10 に示し、パラメータ構成を表 1 に示す。VGG-9 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。BNN では活性化関数として Sign 関数を用いているが、本稿で用いた 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。

5.2.2　VGG-8

提案手法と TWN1 との比較のため、VGG-8 と呼ばれるネットワーク構成を対象とする。VGG-8 の層構成を図 11 に示し、そのパラメータ構成を表 2 に示す。VGG-8 は畳込み層 6 層、プーリング層 3 層、全結合層 2 層で構成される。

5.2.3　ResNet-18

提案手法と 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-18 の層構成を表 3 に示す。Residual unit とは、畳込み層を 2 層通った値と、畳込み層を 2 層通る値と特徴マップ数を同じにするために畳込み層を 1 層だけ通った値を足し合わせ、出力とする構成である。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。

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の 2 種類の画像で構成されている。本稿では、画像データセットとして、表 4 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

Vol.2018-ARC-230 No.37
Vol.2018-SLDM-183 No.37
Vol.2018-EMB-47 No.37
2018/3/8

表 4　データセット

データセット	学習画像	テスト画像	画像サイズ	カテゴリ数
Cipher-10	5 万枚	1 万枚	32 × 32	10
ImageNet	約 122 万枚	約 60 万枚	224 × 224	1,000

表 5　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
学習率を減衰させる epoch 数	80, 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

表 6　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
学習率を減衰させる 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

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、より高い画像識別能力が求められる。

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 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、文献 [5] における 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。

ネットワーク構成を ResNet-18 とするときのパラメータ構成を表 6 に示す。量子化前の学習時の epoch 数は TWN1 で認識精度が収束するまでの epoch 数とした。epoch 数以外の量子化前の学習パラメータ構成は、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、量子化前と同じ値に設定した。

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。構築した評価環境を用いて求めた量子化前の認識精度は、各文献に記載されている値と異なるため、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。

表 7　BNN と提案手法との認識精度の比較（VGG-9、Cifer-10）

手法	係数ビット数	認識精度
BNN	32-bit	88.60%
	1-bit	88.68%
提案手法	32-bit	85.86%
	5-bit	84.93%
	4-bit	82.78%

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

手法	係数ビット数	認識精度
TWN1	32-bit	92.88%
	2-bit	92.56%
提案手法	32-bit	88.05%
	5-bit	86.02%
	4-bit	84.42%

6.　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 7 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。これはほぼ同等の結果と考えることもできるが、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認識率低下の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考えられる。

6.1.2　TWN

TWN1 と提案手法の認識精度を表 8 に示す。TWN1 では、三値化を行った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 と比較してより大きく低下している。提案手法の量子化を行った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。

ResNet-18 のネットワーク構成を用いたときの TWN1, TWN2 と提案手法との認識精度の比較を表 9 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 に対して提案量子化手法による認識精度の低下が大きい。この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り捨てになってしまうため、ある一定以下の値が 0 になり、そのケースが多かった可能性が考えられる。一方で、

表 9　TWN1, TWN2 と提案手法との認識精度の比較（ResNet-18, ImageNet）

手法	係数ビット数	認識精度
TWN1	32-bit	65.40%
	2-bit	61.80%
TWN2	32-bit	57.20%
	2-bit	57.50%
提案手法	32-bit	61.92%
	5-bit	52.40%

表 10　INQ と提案手法との認識精度の比較（ResNet-18, ImageNet）

手法	係数ビット数	認識精度
TWN	32-bit	68.27%
	5-bit	68.98%
	4-bit	68.89%
	3-bit	68.08%
	2-bit	66.02%
提案手法	32-bit	61.92%
	5-bit	52.40%

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可能性が考えられる。

6.2　インクリメンタル量子化

INQ と提案手法との認識精度の比較を表 10 に示す。INQ では係数を 2 ビットまで量子化した場合はわずかに低下したが、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、本稿における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。INQ における量子化演算は、特別な演算式に基づいて行われている。

7.　まとめ

畳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 を用いて構築し、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 Cifer-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を同等以上の認識精度を実現できているが、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。その原因の一つとして、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。

本稿では自由度の高さを根拠に Chainer ver1.24.0 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 以外にもさまざまなオープンソース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もの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含めて今後の課題としたい。

謝辞　本研究の一部は JST, CREST の支援（JPMJCR1432）を受けたものである。

参考文献

[1] G. E. Dahl, N. Jaitly, and R. Salakhutdinov, “Multi-task Neural Networks for QSAR Predictions,” arXiv preprint arXiv:1406.1231, Jun. 2014.
[2] H. Alember, V. Leroy, A. P. Boucle, and 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, and Z. Zhang, “Accelerating Binarized Convolutional Neural Networks with Software-programmable FPGAs,” in *Proc. International Symposium on Field-Programmable Gate Arrays (ISFPGA)*, Feb. 2017.
[4] M. Courbariaux, Y. Bengio, and J. P. David, “Binaryconnect: Training Deep Neural Networks with Binary Weights During Propagations,” in *Proc. Neural Information Processing Systems (NIPS)*, Dec. 2015.
[5] F. Li, B. Zhang, and B. Liu, “Ternary Weight Networks,” in *Proc. Neural Information Processing Systems (NIPS)*, Dec. 2016.
[6] C. Zhu, S. Han, H. Mao, and W. J. Dally, “Trained Ternary Quantization,” in *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, and Y. Chen, “Incremental Networks Quantization: Towards Lossless CNNs with Low-Precision Wights,” in *Proc. International Conference on Learning Representations (ICLR)*, Apr. 2017.
[9] S. Tokui, K. Ono, S. Hido, and J. Clayton, “Chainer: A Next-Generation Open Source Framework for Deep Learning,” in Proc. Workshop on Machine Learning Systems (LearningSys) in the Conference on Neural Information Processing Systems (NIPS), Dec. 2015.
[10] 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. Mané, 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. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” arXiv:1603.04467, Mar. 2016