Vol.2022-ARC-247 No.22
Vol.2022-SLDM-197 No.22
2022/1/25

CNNの組み合わせ回路実装に向けた重み調整によるLUT数削減手法の検討

根尾 優一郎[1]　橋本 昌宜[2]

概要：近年のFPGA技術の進歩により積載可能な回路規模が増大している．低レイテンシで高度な画像処理が求められる場合，CNNを組み合わせ回路に実装することが一つの解決策になる可能性がある．演算器を共有し繰り返し利用する一般的なCNNのFPGA実装に比べ，組み合わせ回路実装は演算を並列に実行するため遅延時間が小さく，重み係数を組み合わせ回路に埋め込むため係数メモリが不要となる利点があるが，論理回路の規模が膨大となる問題がある．そこで係数を埋め込んだ組み合わせ回路の規模が係数に依存することに注目し，推論精度の劣化を抑制しつつ，各係数ごとに量子化ビット数を定めて量子化を行うことで回路規模を削減する手法を検討した．LeNetを用いた評価実験では，畳み込み層において，1.4 %の精度のペナルティで回路規模を85 %削減できるという結果が得られた．

1. はじめに

畳み込みニューラルネットワーク（CNN:Convolutional Neural Network）は，自動運転や画像での不良品検知など，様々な分野において実用化が進められている．計算機技術の進歩によりCPUやGPUなどのプロセッサの性能は向上しているものの，リアルタイム処理に求められる低遅延時間の要求に対し，ソフトウェア実装の改良だけでは限界がある．そのため，ハードウェア実装に関する研究に注目が集まり，特に回路を書き換え可能なFPGA（Field Programmable Gate Array）を用いたCNNアクセラレータの実装が盛んに行われている．近年のFPGA技術の進歩により積載可能な演算回路が増加し，従来より大規模な回路の実装が可能になっている．

そこで本研究では，CNNを組み合わせ回路に全展開して実装することで，低遅延時間での画像認識の実現を目指した．メモリに格納された係数を読み出して繰り返し同じ演算器を利用する一般的なFPGA実装に比べると，組み合わせ回路へ全展開での実装は演算を並列で実行できるため遅延時間が小さく，重み係数を組み合わせ回路に埋め込むため係数メモリが不要であるという利点を持つ．一方，論理回路の規模が膨大となる問題がある．汎用プロセッサでは入力ビット幅が半精度浮動小数点型や倍精度浮動小数点型，32ビット型などに固定されるため重み係数のビット幅をそれらの中から選択する必要がある．一方，組み合わせ回路へ展開する場合は重み係数を組み合わせ回路に埋め込むことにより重み係数のビット幅に指定がなく，重み係数ごとに任意のビット幅にすることが可能である．個の自由度を活用し，重み係数ごとに異なる精度への影響を考慮してによって量子化ビット数を変化させることで精度の劣化を抑えた回路規模の削減手法を検討した．

本稿の構成は次のとおりである．2章ではCNNアクセラレータのFPGA実装と重み量子化について，既存手法を紹介する．3章ではCNNを組み合わせ回路に実装する利点欠点を挙げ，予備実験により課題を明らかにする．4章では3章で課題として明らかになった膨大な回路規模を削減すべく回路規模の削減を直接的に目指した重み係数の量子化手法を提案する．5章では4章の手法を用いてCIFAR-10を対象としたLeNetで実験を行い，提案手法による改善を確認する．

2. 既存手法

2.1 CNNアクセラレータのFPGA実装

CNNは画像認識において高い精度を示すが，学習，推論に膨大な時間を要する．そのため，FPGA上に専用データパスを実装することで効率的にパイプライン処理や並列処理を行い高速化を実現する．[1]に挙げられているように，CNNをFPGAに実装する場合，一般的には全体で1つのシストリックアレイを用いて1層ごとに順に計算する方法[2]，各層ごとに1つの演算器を用意しストリーミング処理を行う方法[3]などがとられる．これらはFPGA内の限られた資源を有効に活用し，スループットを向上させるこ

[1] 大阪大学情報科学研究科情報システム工学専攻
[2] 京都大学情報学研究科通信情報システム専攻

とに成功して以降，次節の重み量子化や多様なネットワークに対応した実装が多く報告されている [4]．基本的に，演算器をニューロンや層等で共有し，メモリから重みを読み出して繰り返し演算を行うアーキテクチャとなっている．

2.2 CNN の重み量子化

CNN の重み係数の量子化は，浮動小数点での学習で得られた重み係数に対し，固定小数点での値や二値などに量子化することである．量子化方法には，線形量子化，非線形量子化がある．量子化の手順としては，全ての重み係数を一斉に量子化する方法や，段階的に量子化を進める方法などがある．

二値に量子化するものとして BNN（Binarized Newral Network）[5] があり，重み係数を-1,1 の二値にして推論を行う．活性化関数は Sign 関数を用いて，正なら 1，負なら—1 を返すことで二値化を行っている．固定小数点では，GPU でも INT8 がサポートされるなど，少ないビット数への量子化例が多く報告されている [6]．

線形量子化は量子化前の値を均等な範囲で量子化後の値に割り当てるのに対し，非線形量子化は割当てられる量子化前の値の範囲が量子化後の値によって異なるものとなる．非線形量子化の例には対数量子化 [7] が挙げられる．

段階的に量子化の手法として，INQ（Incremental Network Quantization）[8] がある．これは，学習で得られた重みの半数を量子化し固定したうえで再度学習を進め，精度が終息したのち未量子化係数のうちから半数を量子化する，という作業を指定回数行うことで量子化を行う．この量子化では，重み係数は 2 の乗数または 0 の値に量子化する．これらの値に限定することで，重み係数の乗算がシフト演算のみで実現できるため，計算時間の削減が可能である．

また，量子化ビット数を層ごとに設定する手法 [9] もあり，こちらでは層ごとに重み係数の値の範囲を制限して量子化している．重み係数のビット幅が可変長であるハードウェアアクセラレータは存在するものの，それらの多くは層ごとに重みのビット幅が決まっている．

3. CNN の組み合わせ回路実装とその課題

3.1 CNN の組み合わせ回路実装

既存の CNN アクセラレータの FPGA 実装では，演算器を共有し，繰り返し利用することで推論計算を行っている．一般には，層ごとに分割して計算処理が実行され，層内でも共有された演算器が繰り返し利用されて計算が進むため，遅延時間が大きくなる．そこでネットワーク全体を組み合わせ回路に全展開し，遅延時間の削減を行うことを考える．ネットワークを全てを組み合わせ回路に展開する場合には，演算器を共有しないため全ての計算を並列に実行できるようになり遅延時間の削減が期待される．

3.2 期待される利点，欠点

CNN の組み合わせ回路への全展開は，ネットワーク内の演算 1 つにつき 1 つの専用回路を用意し完全に並列化して演算を行う．ネットワークの係数を用いる演算は係数を埋め込んだ回路で実装する．一般的な乗算は 2 入力の乗算器で実現されるが，これをネットワークの重み係数 w を用いて 1 入力の w 倍器で実装する．重み係数は固定小数点で表現されているため，この w 倍器は整数倍器と同等である．これらによって，1 つの入力に対して反復して回路を使用せずに並列実行できるため遅延時間が小さくなる．また RAM から係数メモリを読み出す必要がなくなることでメモリアクセス時間と保存に必要な RAM が不要になる．例えば，畳み込み層 2 層，全結合層 3 層の CNN である LeNet（表 1）では，すべての重み係数が 8 ビットの場合，375,280 ビット分の RAM が不要になる．畳み込み層は $F \times F \times C_{in}$（F：フィルタサイズ，C_{in}：入力チャンネル）のフィルタ演算であるため，$F \times F$ 個の入力に対し各々に重み係数を掛けたものの総和を求める積和演算を行う．つまり，1 つの出力につき，$F \times F \times C_{in}$ 入力の積和演算器が必要となる．重み係数は回路合成前に設定される定数であるため，重みに合わせた N 倍器と加算器で構成される．畳み込み回路はこの積和演算器と全く同じものが出力の数だけ必要となる．

全結合層は，層への入力全てを入力とする積和演算を行う．こちらも同様に重み係数は回路合成前に設定される定数であるため，重みに合わせた N 倍器と加算器で構成される．畳み込み回路の場合は係数の同じ積和演算器を必要とするが，全結合層の場合は各ノードの重み係数に合わせて乗数が異なる積和演算器が出力の数だけ必要となる．

表 1: MNIST 向け LeNet の構成

画像 28x28x1	
conv1	入力チャンネル：1 出力チャンネル：6 フィルタサイズ：5x5
maxpool1	2x2
conv2	入力チャンネル：6 出力チャンネル：16 フィルタサイズ：5x5
maxpool2	2x2
fc1	入力：256 出力：128
fc2	入力：128 出力：84
fc3	入力：84 出力：10

表 2: LeNet の構成要素と各々の回路規模

回路	遅延時間	LUT 数	必要数
conv 5x5	4.42ns	998	24x24x6
maxpool 2x2	2.52ns	34	12x12x6
conv 5x5	4.42ns	998	8x8x16
maxpool 2x2	2.52ns	34	4x4x16
FC 256 → 1	7.45ns	8640	128
FC 128 → 1	6.67ns	4383	84
FC 84 → 1	6.22ns	2776	10

3.3 予備実験

畳み込み層 2 層，全結合層 3 層の CNN である LeNet[10] を対象として遅延時間と回路規模の概算見積りを行った．LeNet の構成を表 1 に示す．入力画像は MNIST に合わせて 28x28x1 を想定し，重み係数を 8bit 量子化したモデルを用意した．回路規模が大きく合成に膨大な時間がかかるため，各層で別々に回路合成した結果を表 2 に示す．この結果から，1 枚の画像認識が遅延時間 34.21ns で実行可能であるが，600 万個の LUT が必要となることがわかる．計算時間は，3x3 ピクセルの n と v を判別するイメージセンサ—内画像処理回路 [11] で 50ns 程度の計算時間のため，この遅延時間は十分小さいといえる．しかし，一般的な FPGA の LUT 積載数が 100 万個程度であることを考えると，回路規模がかなり大きく大幅な削減が求められる．そこで本研究では，学習済みのネットワークに対して，演算回路の改善と組み合わせ回路化に適した量子化手法を考え，精度劣化を抑えて回路規模の削減を図る手法を次章で考える．

4. 回路規模の削減

4.1 回路規模を削減する重み係数の量子化手法

4.1.1 方針

重み係数のビット幅はハードウェアの規模に影響を与える．重み係数を保存する RAM や演算器の入力・出力ビット数の変化により規模が変化するためである．推論精度の劣化を抑制しつつ，量子化を進めることで回路規模の縮小が期待できる．一方，この重みの量子化による回路規模削減問題では，膨大な数の重み係数に対してビット幅を定め，そのビット幅に量子化した値を割り当てる必要がある．大規模な組み合わせ最適化問題であり，また回路規模を正確に見積もるためには長時間の回路合成が必要となるため，厳密な解を求めることは不可能である．そのため，回路合成を必要とせず，大量の重み係数に対して実用的な時間で解を求めることができる発見的手法の構築が求められる．

多数の重み係数を効率的に量子化するため，[12] にならって入力側の層から逐次的に重み係数の量子化を行うことにする．上流の層で行った量子化により推論精度は劣化するものの，後段の層の再学習により精度劣化を抑制することが期待できるためである．この逐次的量子化では，先に量子化する層をどの程度まで量子化しておくと，ネットワーク全体として回路規模と推論精度のトレードオフの中で良い解が得られるかを正確に予測することは難しい．そのため，後述の目的関数を複数設定し，各層ごとに目的関数と推論精度のトレードオフの中でのパレート解を選択し，後段の量子化への入力とすることにする．

上述のパレート解を回路合成なく求めるためには，回路規模と密接に関係する評価指標を定め，回路規模と推論精度の関係を，新たな評価指標と推論精度の関係に置き換える必要がある．ここで，重み係数中の 1 の総数と重み係数のビット幅の総和と 2 つの指標を考える．正の整数の乗算では，1 の総数は部分積の数に相当するため，回路規模に強い影響を与える．一方で，負の数を含む重み係数を考えると，必ずしも良い指標とはならない．一方で重み係数のビット幅の総和は負の係数の場合でも，回路規模と強い相関を持つ指標となる．そのため，重み係数のビット幅の総和を回路規模に変わる指標として採用する．

4.1.2 概要

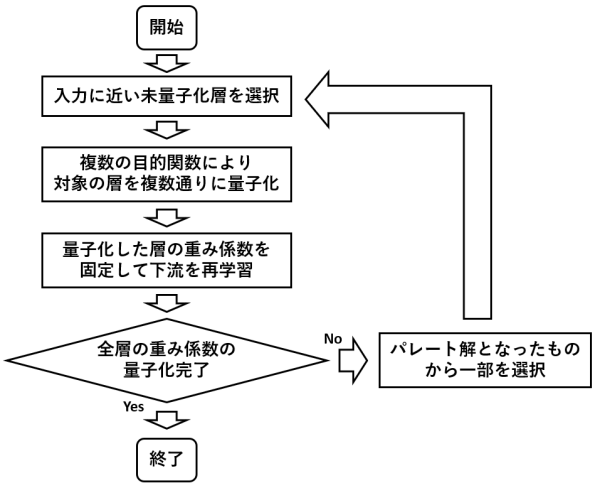
量子化の流れを図 1(a) に示す．まず入力層の重み係数に対して目的関数を複数適用して量子化を行う．次に，複数できた上流のみ量子化済みのモデルに対して量子化済みの層の重み係数を固定し，下流の層に対して再学習を行う．これにより，同じ層まで量子化済みの複数のモデルが生成される．最後に，未量子化層が残っている場合には，生成された複数のモデルのうち，パレート解となっているもののみを残し，次に入力に近い層の量子化を行う．これを全ての層が量子化できるまで行う．

例えば，図 1(b) では，conv1 層の重み係数を，(b)-1 は重み係数のビット幅を大幅に減らすように，(b)-2 は元の重み係数の値に近くなるように量子化したものとなる．これらの重み係数の値でモデルの値を更新し，図 1(c) のように固定したうえで下流の未量子化層の重みを再学習する．次の層の量子化を行う場合には，(b)-1，(b)-2 それぞれに対し複数の目的関数によって量子化されたモデルが生成されることとなる．

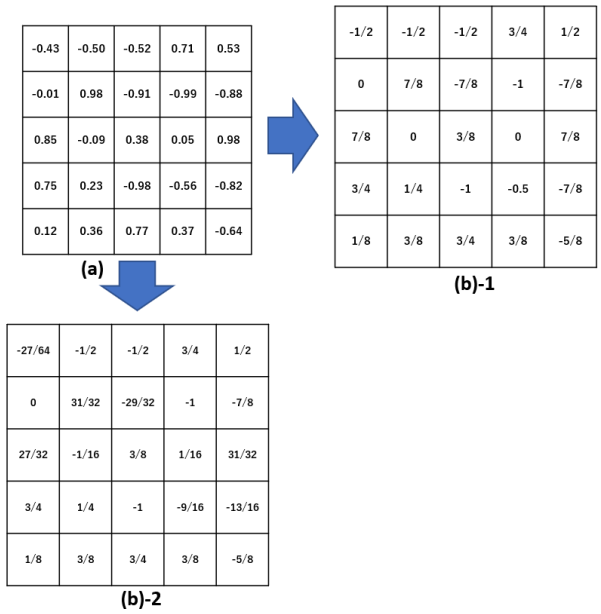
4.1.3 層内のビット幅の総和最小化

層内の重み係数の量子化では，推論精度と重み係数のビット幅の総和のトレードオフの中で，パレート解を求める必要がある．ここで，層内に限っても重み係数の数は膨大にあるため，各重み係数のビット幅と量子化値の最適解を求めることは難しく，また単純に貪欲手法に頼ると膨大な数の推論回数が必要となる．そこで，推論を必要としない目的関数を設定し，解候補を求めることとする．目的関数中に含まれるパラメータを変化させることで，複数の解候補を生成し，解候補ごとに最後に推論精度評価を 1 度だけ行うことで計算時間を削減する．

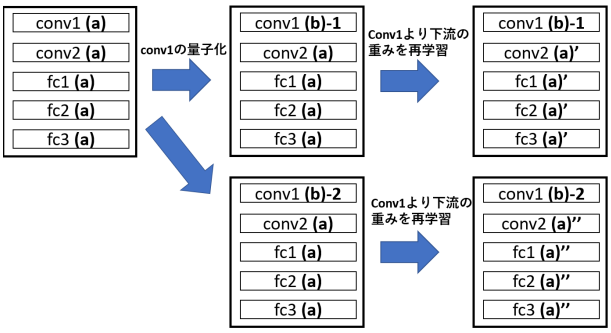
上記の実現に適した目的関数を考える．重み係数のビット幅の総和を最小化したいため，まず重み係数のビット幅



(a) 量子化の流れ



(b) conv1 層の量子化



(c) 量子化済みの層より下流の再学習

図 1: 量子化の概要

の総和は目的関数に含まれるべきである．また，量子化によって重み係数の値が量子化の前から変化する．変化幅が大きいと精度劣化が大きくなると予想されるため，変化幅の総和は小さくあるべきである．一方で許容できる変化幅は重み係数ごとに異なることが予想される．つまり推論結果に対する感度が高い係数は，推論精度に与える影響が大きく，逆に感度が低い係数は大胆な量子化を許容する可能性がある．しかし，感度の低い係数であっても勾配の正負と逆方向に大胆に量子化した場合には，推論精度に与える影響は無視できない大きさになると考えられる．

以上の議論から，層内のビット幅の総和最小化問題を以下のように定式化する．

入力： 1 層の重み係数 $\boldsymbol{w_{before}}$
正規化した勾配 $\boldsymbol{g}$

変数： 量子化後の重み係数 $\boldsymbol{w_{after}}$

目的関数（最小化）：

$$\sum_{i}^{n} \mathrm{bit\_width}(\boldsymbol{w_{after}}) + \boldsymbol{\alpha} \sum_{i}^{n} |\boldsymbol{w_{after}} - \boldsymbol{w_{before}}||\boldsymbol{g}^{(i)}| - \boldsymbol{\beta} \sum_{i}^{n} \mathrm{sign}(\boldsymbol{w_{after}} - \boldsymbol{w_{before}})\boldsymbol{g}^{(i)} \quad (1)$$

制約： $|\boldsymbol{w_{after}} - \boldsymbol{w_{before}}| \leq \boldsymbol{R}$

n は重み係数の数であり，畳み込み層では $n = F^2 \times C_{in} \times C_{out}$（$F$：フィルタサイズ，$C_{in}$：入力チャンネル数，$C_{out}$：出力チャンネル数），全結合層では $n = p \times q$（p：入力数，q：出力数）となる．bit_width(x) はビット幅を返す関数，sign(x) は符号関数であり，x>0 なら 1，x=0 なら 0，x<0 なら-1 を返す関数である．ここで，α，β は第二項，第三項の重みを表す係数である．また，R は量子化による重みの値の最大変化値である．α，β，R を変化させることで目的関数をさまざまに変化させることができ，異なる解候補を生成することができる．

4.2 シフトによる定数乗算器の実現

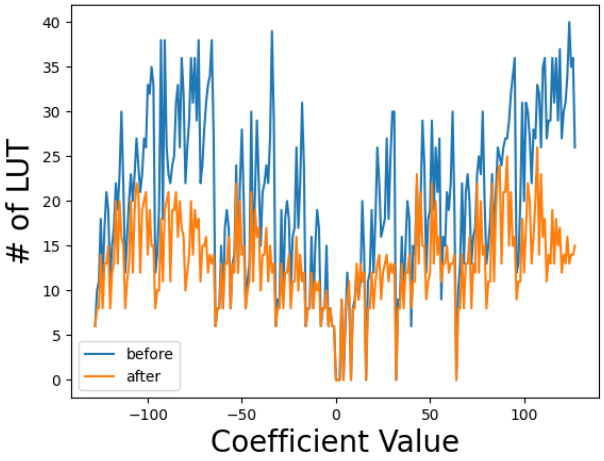
一般的な乗算器は部分積の加算によって乗算結果を求める．それに対し，本研究では定数倍器で計算を実現できるため，[13] で示されているような，シフトと加減算を用いた構造にすることで乗算回路の規模を縮小可能である．シフト演算は配線により実現できるため LUT が不要であることから特に有効である．

5. 実験

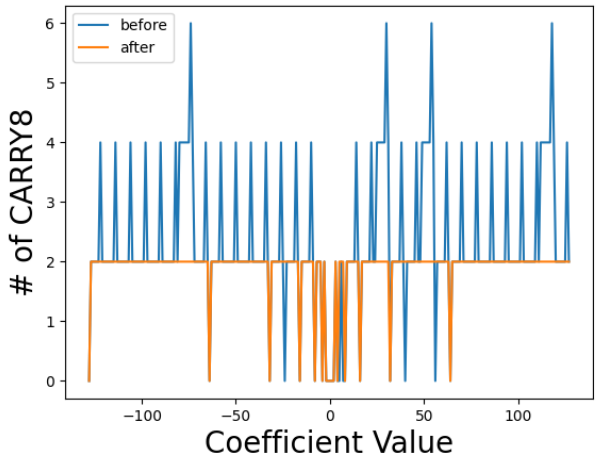
以下の実験では，回路合成ツールには Xilinx 社の vivado(v2020.1) を使い，FPGA は Virtex UltraScale+ FPGA VCU118 を対象として回路合成を行った．

5.1 シフトを用いた定数乗算器による回路規模削減

4.2 節の手法を用いて定数乗算器を合成した．これによる乗算部分での回路規模の削減を図 2(a) に示す．横軸は乗数を示す．削減前は HDL 内で乗算記号 $*$ を用いて回路を記述している．削減前と比較して LUT 数を平均で 45 %

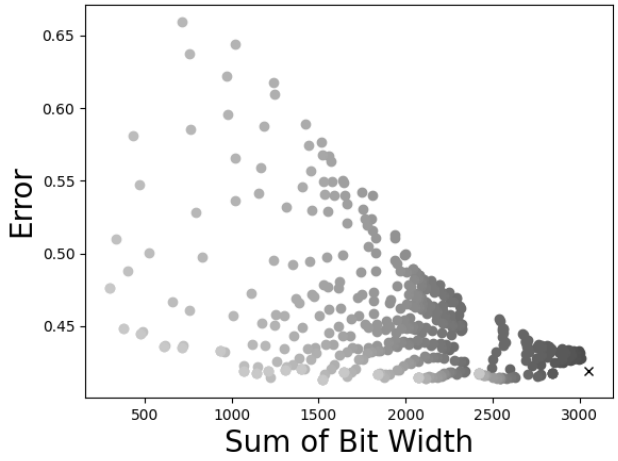


(a) 乗算器 1 つに必要な LUT 数

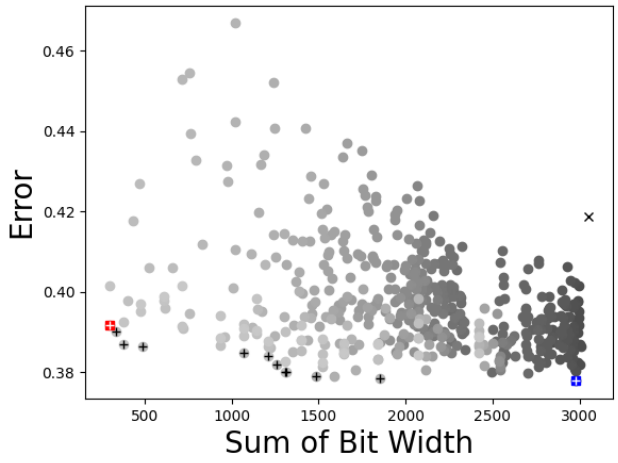


(b) 乗算器 1 つに必要な CARRY8 数

図 2: 乗算器に必要な論理素子数



(a) 量子化直後の精度



(b) 再学習後の精度

図 3: 畳み込み 1 層目の量子化結果
縦軸は推論誤差，横軸は conv1 層の重みのビット幅の総和である．x 印は重みを 8 ビット量子化した状態のモデルを示し，＋印のついたものはパレート解を示す．図 3(a)，図 3(b) ともに色の濃さで α の大きさを表しており，濃いほど α が大きい．

削減している．また，FPGA に存在する回路素子として高速キャリーロジックである CARRY8 がある．乗算器の回路規模削減後は，こちらも平均で 18 %の削減された．一部の値では元の手法で回路を構成したほうが回路規模が小さくなる場合があるが，それらの低数値では元の手法で構成した回路が改良後の他の値と比較しても十分小さいため元の手法で構成した回路を用いればよい．5.3 節の実験では，シフトを用いた定数乗算器を用いる．

5.2 重み係数の量子化による回路規模削減

5.2.1 実験目的・条件

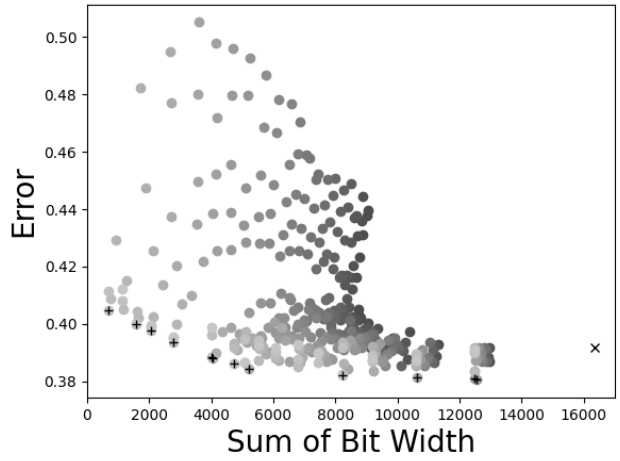
CIFAR-10 を学習した LeNet を例として，学習済みモデルを提案手法で量子化し重みのビット幅と精度の関係を確認する．量子化前のモデルは学習時に重みを 8 ビットにしており，このモデルを上流の層から各重みのビット幅が小さくなるようになるように量子化した．

量子化は，学習済みモデルに対し，4.1 で示した手法でを α は [0,10]，R は [1,32] の範囲を変化させて最適化を行った．

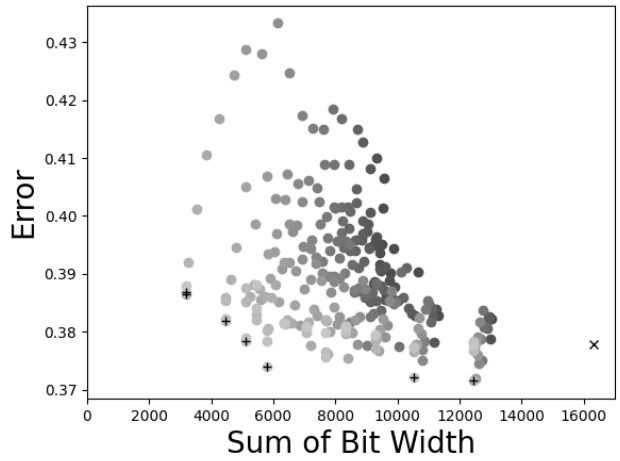
5.2.2 実験結果

図 3(a) は conv1 層を量子化直後の精度であり，図 3(b) は量子化後，量子化済みの重み以外を再学習したものである．図 3 の点の色は α の値の大小を示しており，α が小さいものはビット幅が大きく削減されており，量子化直後は精度劣化が目立つ．一方，図 3(a) と図 3(b) を比較すると，再学習後には精度の向上している．conv1 層の重みの総ビット幅を 1/10 以下にしたにも関わらず，精度が 2.7 % 上がっているものもある．精度が上がった理由としては，量子化に伴い局所解から値が移動し最適解に近づいたことが考えられる．

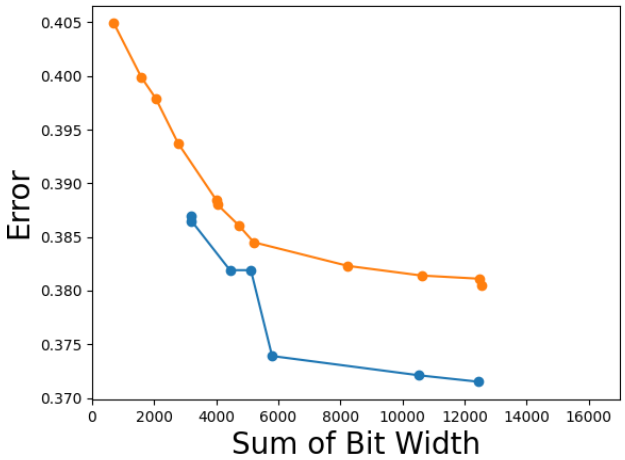
図 4 は，conv1 層を量子化したモデルに対し，conv2 層



(a)

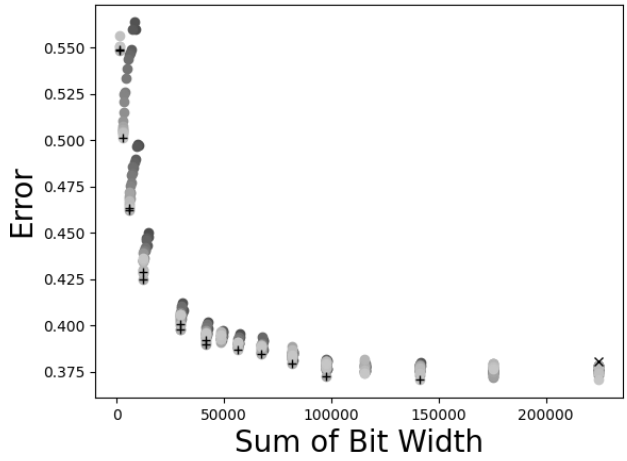


(b)

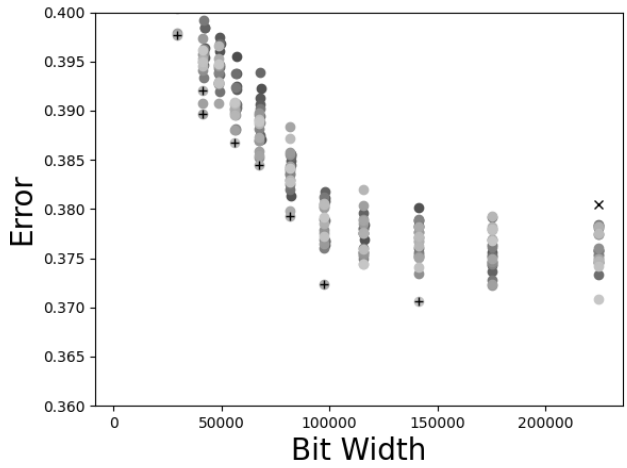


(c)

図 4: 畳み込み 2 層目の量子化結果
図 4(a) は図 3(b) の左端のパレート解（赤の□印）の conv2 層を量子化したもの，図 4(b) は図 3(b) の右端のパレート解（青の□印）の conv2 層を量子化したものである．図 4(c) は図 4(a) と図 4(b) で示したパレート解の比較である．橙が図 4(a)，青が図 4(b) に対応している．



(a)



(b)

図 5: 全結合層 1 層まで量子化した結果
図 5(a) は，図 4(a) の右端のパレート解の fc1 層を量子化したものである．図 5(b) は図 5(a) の拡大図である．

を量子化した結果である．conv1 層の重みのビット幅を大きく減らした場合を図 4(a) に，conv1 層のビット幅を保った場合を図 4(b) に示す．これらのパレート解の比較を図 4(c) に示す．conv1 層の重み係数のビット幅を大きく削った場合，conv2 層の重みのビット幅を同程度削減した場合に精度が劣化する結果となった．特に，最も誤差を小さくした場合に 1 %程度の差が見られた．

図 5(a) は，conv1 層，conv2 層の双方が量子化済みのモデルに対して fc1 層を量子化した結果である．図 3(b) や図 4 と比較すると，ビット幅による精度への影響が顕著に表れている．このことから，畳み込み層ではビット幅を削減し，全結合層ではビット幅を残すようにすることで精度を維持したビット幅の削減が可能であると考えられる．

表 3: 量子化後の conv1 層演算回路規模の比較

総ビット幅	LUT 数	CARRY8 数	精度
296	1,109,170	178,352	0.608
487	1,562,750	264,204	0.613
1262	3,317,515	582,600	0.618
1314	3,516,231	613,696	0.620
2979	7,212,999	1,230,208	0.622

5.3 量子化した conv1 層の回路規模比較

図 3(b) のパレート解のうち 5 つの conv1 層を回路合成した結果を表 3 に示す．この表から，総ビット幅に合わせて回路規模も大きくなっていることが分かる．また，1.4 %の精度のペナルティで LUT 数，CARRY8 数をともに 15 %程度まで削減できている．ネットワーク全体での回路規模削減効果の評価は今後の課題である．

6. まとめ

本稿では，組み合わせ回路に CNN を全展開する上で有効となるシフトを用いた定数乗算器の評価と，回路規模を削減する量子化手法を検討し，それらの実験的評価を行った．シフトを用いた乗算器回路により LUT 数を平均 45 %，CARRY8 を平均 18 %削減することが可能であった．また，学習済みモデルの勾配を用いた重み係数ごとの量子化ビット数可変の量子化手法を提案し，精度を維持しつつ重み係数のビット幅削減を行った．conv1 層では，1.4 %の精度ペナルティで 85 %の回路規模削減が可能であった．

今後の課題としては，ネットワーク全体の回路規模の評価を実施し，パレート解を複数評価することで回路規模の最小化ができていることを確認することが挙げられる．

謝辞

本研究の一部は科研費基盤研究 (B)(課題番号: 19H04079) ならびに JST CREST (課題番号: JPMJCR19K5) による．

参考文献

[1] Janardan Misra and Indranil Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, Vol. 74, No. 1-3, pp. 239–255, 2010.

[2] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pp. 161–170, 2015.

[3] Stylianos I Venieris and Christos-Savvas Bouganis. fpgaconvnet: A framework for mapping convolutional neural networks on fpgas. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 40–47. IEEE, 2016.

[4] Neural Network Accelerator Comparison. `https://nicsefc.ee.tsinghua.edu.cn/project.html`.

[5] Eriko Nurvitadhi, David Sheffield, Jaewoong Sim, Asit Mishra, Ganesh Venkatesh, and Debbie Marr. Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic. In *2016 International Conference on Field-Programmable Technology (FPT)*, pp. 77–84. IEEE, 2016.

[6] Jian Cheng, Jiaxiang Wu, Cong Leng, Yuhang Wang, and Qinghao Hu. Quantized cnn: A unified approach to accelerate and compress convolutional networks. *IEEE transactions on neural networks and learning systems*, Vol. 29, No. 10, pp. 4730–4743, 2017.

[7] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[8] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[9] Bert Moons, Bert De Brabandere, Luc Van Gool, and Marian Verhelst. Energy-efficient convnets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–8. IEEE, 2016.

[10] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pp. 319–345. Springer, 1999.

[11] Lukas Mennel, Joanna Symonowicz, Stefan Wachter, Dmitry K Polyushkin, Aday J Molina-Mendoza, and Thomas Mueller. Ultrafast machine vision with 2d material neural network image sensors. *Nature*, Vol. 579, No. 7797, pp. 62–66, 2020.

[12] 氏原収悟, 密山幸男ほか. 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. 研究報告システムと LSI の設計技術 (SLDM), Vol. 2018, No. 37, pp. 1–6, 2018.

[13] Michael A Soderstrand. Csd multipliers for fpga dsp applications. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, Vol. 5, pp. V–V. IEEE, 2003.