Vol.2019-SLDM-189 No.24
Vol.2019-EMB52 No.24
2019/11/14

手書き文字認識 CNN に対する近似乗算器の設計探索

白根健太[1]　山元貴普[1]　冨山宏之[1]

概要：本論文では，近似乗算器を手書き文字認識 CNN へ適用した事例について述べる．複数の異なるビット幅の近似乗算器を適用した手書き文字認識 CNN を用いて画像認識率を評価し，近似乗算器の回路面積，遅延時間，画像識別率のトレードオフを解析する．解析結果を基に近似乗算器を改良し，近似乗算器の回路面積の削減を行った．

キーワード：Approximate Computing，近似乗算器, CNN, MNIST

1. はじめに

畳込みニューラルネットワークは画像認識や自然言語処理などの分野で広く知られているアプリケーションである．近年，CNN (Convolutional Neural Network) に向けた計算機設計に関する研究がますます活発になっている．一般的に, CNN は行列の乗算を多く繰り返すことがボトルネックとなり，計算コストが高くなる傾向がある．行列乗算のコストを削減する手法の一つとして，Approximate computing が注目されている．Approximate computing はある程度の計算誤差に耐性のあるアプリケーションに向けた回路面積，実行時間，電力において効率の良い回路を実現させる設計手法である．中でも Approximate computing による近似乗算器の設計は広く研究されている[1][2]．文献[3]や文献[4]では上位ビットの計算を正確に乗算し，下位ビットの計算に関する論理ゲートを削減し，近似乗算する手法が提案されている．文献[5]では 8 ビット配列型乗算器の計算に用いる加算器を削除する体系的な設計手法を提案し，設計した 57 個の近似乗算器の回路面積，遅延，電力，精度のトレードオフを解析している．文献[6]ではいくつかの近似圧縮器によって部分積を削減した乗算器の設計探索を行っている．また，この研究ではユーザーが，許容する誤差を指定してそれに対応する面積，遅延時間，または電力において最適な近似乗算器を設計するツールも提案している．文献[7]では高効率な部分積圧縮と誤差低減を実現し，近似乗算器を設計している．この近似乗算器は正確な計算と同様の精度にもかかわらず高パフォーマンスであり低電力で動作することができる．

CNN に向けた近似乗算器に関しても, 多数研究されている．文献[8]では CNN の誤差耐性分析から電力効率の良い近似乗算器を設計している．この研究では MNIST と Street-View House Number データセットの画像認識を行い，設計した近似乗算器の有用性を示している．文献[9]では VGGNet に近似乗算器を適用した場合のシミュレーションを行い，画像認識により高い画像識別率を維持できることを示した．文献[10]では Stochastic Computing に基づく乗算アルゴリズムとそのベクトル拡張を CNN に適用させることを提案し, 実験により通常の 40 倍から 490 倍のエネルギー効率での画像認識が実現できることを示した．文献[11]ではニューラルネットのパラメータをバイナリで学習させる手法が提案されている．このネットワークは BinaryNet と呼ばれており，メモリ資源を大きく減少させることや XNOR 回路を乗算器として扱うことができることから，様々なアクセラレータや改善手法が提案されている[12][13]．文献[14]や文献[15]では 8 ビットの固定小数点演算近似乗算器を LeNet に適用し, FPGA に実装させている．どちらの文献も MNIST データセットでの画像識別率を維持しつつ遅延時間，回路リソースを削減することを可能としている．

文献[16]では乗算器のビット幅の削減を行い CNN に適用し，MNIST データセットの画像識別率，回路面積，遅延時間のトレードオフを解析している．また，解析結果から効率の良いビット幅の組み合わせの近似乗算器を一部の出力を変更しカルノー図を用いてさらに最適化を行った．この手法ではゲート数の削減の面では一定の効果が得られたが，遅延時間の面ではより良い回路が存在していた．そこで，本論文では画像識別率を維持しつつ，回路面積(ゲート数)と遅延時間の両面で効率的な回路を目指す．

本論文の構成は以下の通りである．2 章では一般的な CNN と，MNIST データセットを識別する CNN について述べる．3 章では乗算器のビット幅削減手法とその実験について述べる．4 章ではより効率的な回路の設計手法を述べる．5 章ではまとめと今後の課題について述べる．

2. 手書き文字認識 CNN

本章では一般的な CNN とその畳込み層に関して述べる．CNN は畳込み層を含むニューラルネットである．CNN は高精度な画像認識モデルとして広く知られている．Alexnet は最も有名な画像認識モデルの一つであり，2012 年に画像認識競技会 ILSVRC において大差で優勝したことで CNN が世界に広まるきっかけとなった．

1 立命館大学
Ritsumeikan University

Vol.2019-SLDM-189 No.24
Vol.2019-EMB52 No.24
2019/11/14

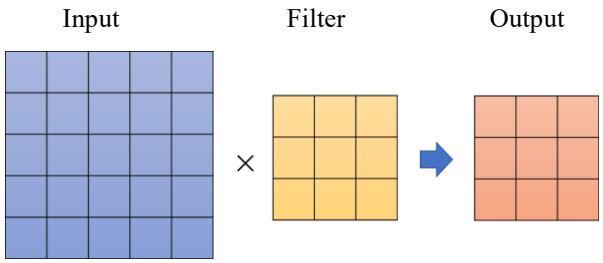


図 1　畳込み層

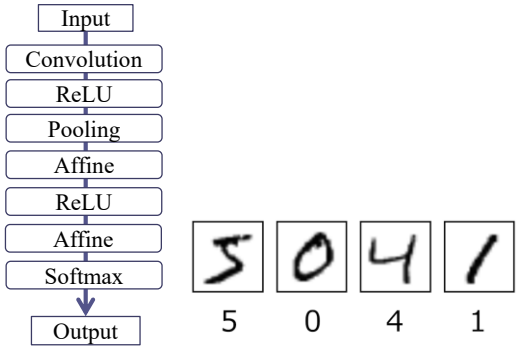


図 2　手書き文字認識 CNN と MNIST データセット

2.1　畳込み層

畳込み層は画像データとフィルタデータに対して畳込み処理を行う層である．その結果を次の層や活性化関数に受け渡す．畳込み処理では画像データとフィルタデータの積和を取り対応した場所に結果を格納している．この処理は stride という値に従って一定の間隔で行われる．図 1 に畳込み処理の例を示す．5×5 ピクセルの画像データに対し 3×3 のフィルタを stride = 1 の間隔で畳み込むとその出力は 3×3 となる．この場合，畳込み処理は 1 つのフィルタにつき 81 回の乗算処理を行う．一般的に CNN は多くのフィルタを持つ．本研究で用いる CNN は 1 画像につき 432,000 回の乗算処理を行う．

2.2　手書き文字認識 CNN の例

図 2 は手書き文字データセット MNIST を識別する CNN モデルと MNIST データセットの例である．この CNN は畳込み層 (Convolution)，プーリング層 (Pooling)，全結合層 (Affine)，ReLU 関数，Softmax 関数から構成されている．本節では畳込み層以外のそれぞれの層や活性化関数について簡潔に述べる．

プーリング層はある範囲の入力から最大値を抽出して対応する出力に格納する．この層はパラメータが持つ空間的な情報を維持したまま近似する役割を担っている．

全結合層は全ての入力に重みの値を掛け合わせ，その合計を対応する出力に格納する層である．この処理は判定する各クラスの数だけ行われ，最終段の全結合層では後の Softmax と組み合わされ，そのクラスである確率が算出される．

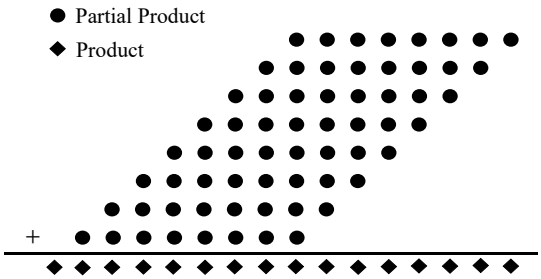


図 3　8 ビット乗算器

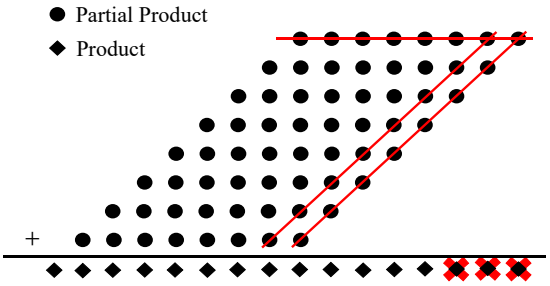


図 4　7×6 ビット乗算器

ReLU 関数と Softmax 関数は各層の出力に対してある処理を行い，後の層に値を渡す活性化関数である．ReLU 関数は 0 より小さい値を 0 として，それ以外の値をそのまま出力する関数である．この機能はランプ関数として知られている．Softmax 関数は最終段で全結合層の後に用いられ，入力を確率の値に直し，最も高い確率のクラスを CNN の画像識別結果として出力する．

この CNN を用いて手書き文字のデータセットである MNIST を学習させた．MNIST は 0~9 までの手書き文字で構成されており，28×28 の疑似バイナリ画像がグレースケール画像として 0~1 の値で格納されている．

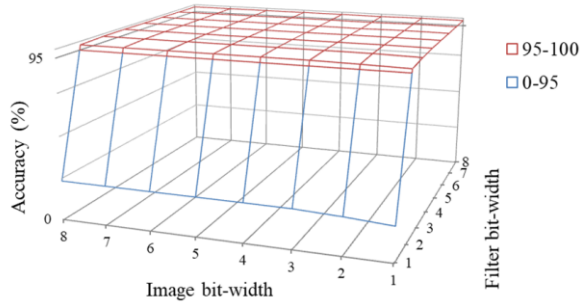
3.　過去の研究

本章では本研究の先行研究である文献[16]の概要を述べる．まず，ビット幅を削減する手法で様々な精度の乗算器を設計し，CNN に適用させた実験について述べる．次に，実験結果に基づいた 2 ビット乗算器の解析に関して説明する．最後にカルノー図を用いた近似乗算器の設計について述べる．

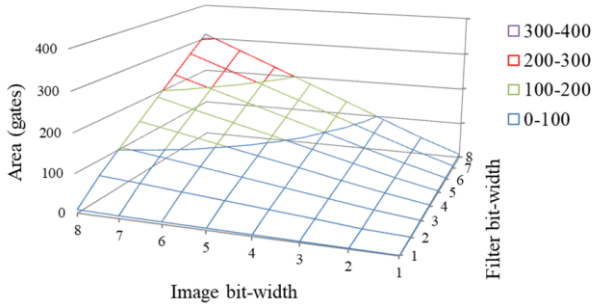
3.1　乗算器のビット幅削減

図 3 は正確な 8 ビット乗算器の例である．乗算演算では被乗数ビットと乗数ビットの部分積の集合から構成されており，それらの和が積である．部分積の和は右上から左下のビットにかけて順次行われる．8 ビット乗算の積は 16 ビットとなる．

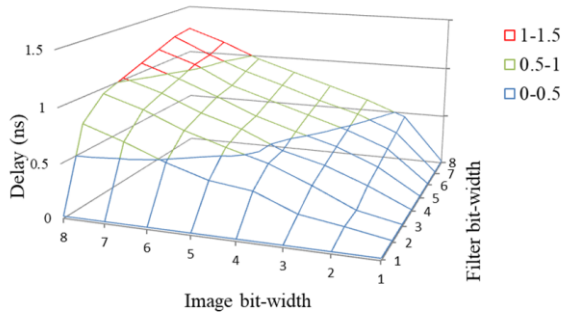
Vol.2019-SLDM-189 No.24
Vol.2019-EMB52 No.24
2019/11/14



(a) 画像識別率



(b) 面積(ゲート数)



(c) 遅延時間

図 5　ビット幅の削減実験結果

図 3 の乗算器を基にしてにビット幅を削減する．被乗数か乗数の桁を下の桁から順番に 0 に置き換えることで乗算器のビット幅を削減する．0 に置き換えた被乗数または乗数に関する部分積が 0 となるため，一度の削減で多くの部分積が消え，積の桁数も削減される．図 4 は 7×6 ビット乗算器の例である．被乗数の最下位 1 ビットと乗数の下位 2 ビットを 0 に置き換えるため，22 個の部分積が削除され，積の下位 3 ビットが削除される．

このビット幅の削減により設計した乗算器を MNIST 認識 CNN に適用する実験を行った．様々なビット幅の乗算器を CNN に適用することで CNN の画像識別率，乗算器の回路面積，遅延時間とのトレードオフを評価する．8 ビット乗算器のビット幅は被乗数と乗数で異なるビット幅を削減することができるため 64 個の乗算器を生成することができる．実験で用いる CNN は 2000 枚の教師画像を学習させたパラメータを使用し，1000 枚のテスト画像を識別させて画像識別率を測定する．乗算器の面積，遅延時間は Verilog で記述し設計されたものを測定する．乗算器を LeonardoSpectrum で合成し，ゲート数と最大遅延時間を測定する．合成は Nangate45 ライブラリを使用して行い，合成は面積が小さくなることを優先して行う．

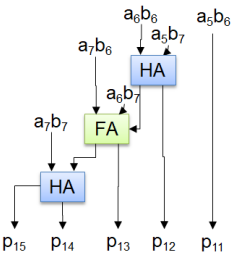


図 6　3×2 ビット乗算器

実験結果を図 5 (a)-(c) に示す．フィルタデータを被乗数とし，画像データを乗数とする．図 5 (a) は画像識別率，図 5 (b) は回路面積 (ゲート数)，図 5 (c)は遅延時間の測定結果である．正確な乗算器を用いて測定した画像識別率は 98.4%である．図 5 (a) より，8×8 ビット乗算器の画像識別率は 98.4%であり，4×1 ビット乗算器も同等の結果が得られたことが確認できた．また，2×1 ビット乗算器は 97.1%の識別率を維持し，それ以降は著しく低下することが分かった．また，図 5 (b), 図 5 (c) より各乗算器の回路面積，遅延時間を確認することができた．

3.2 カルノー図を用いた近似

本節ではカルノー図を用いた近似手法について述べる．3×2 ビット乗算器の一部の出力を近似することで回路を単純化する．図 6 に 3×2 ビット乗算器を示す．'a'はフィルタデータ，'b'は画像データ，'p'は積を表し'HA'は半加算器，'FA'は全加算器である．カルノー図に改良した乗算器の出力を図 7 に示す．0 から 1 に変更した出力を 0→1 と示し，1 から 0 に変更した出力を 1→0 と示す．この変更により，部分積の加算に繰上げが発生した場合に 2 分の 1 の値を出力する回路とみなせる．p15 は p14 の繰上げの出力のみのため削除し，p11 は最下位ビットの部分積の出力のため削除する．

Vol.2019-SLDM-189 No.24
Vol.2019-EMB52 No.24
2019/11/14

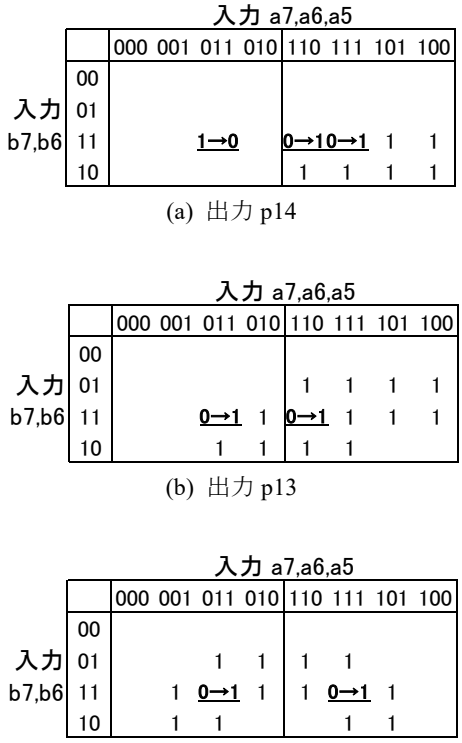


図 7　近似した 3×2 ビット乗算器のカルノー図

表 1　3×2 ビット乗算器の近似，実験結果

	3×1	4×1	3×2	近似 3×2
画像識別率(%)	98.0	98.5	98.5	98.4
回路面積(ゲート数)	3	4	15	5
遅延時間(ns)	0.02	0.02	0.10	0.05

この近似を行った乗算器を近似 3×2 ビット乗算器として 3.1 節と同様の実験を行った．3.1 節における 3×1 ビット乗算器，4×1 ビット乗算器，3×2 ビット乗算器をそれぞれ 3×1, 4×1, 3×2 と示す．実験結果を表 3 に示す．4×1, 3×2 では画像識別率が正しい CNN よりも高い値となっているが，これは稀有なケースであると考えられる．また，近似 3×2 は，3×2 より 10 ゲート減少し，0.05 ナノ秒遅延時間を短縮した．しかしながら，4×1 ビット乗算器と比べるとゲート数，遅延時間の両面で削減することができなかった．

このように，事前研究ではカルノー図を用いた近似により画像識別率を正確な乗算を行った場合と同様の値まで維持することができた．しかし，遅延時間においては十分に最適化できていなかった．表 2 に 3.1 節の実験における画像データのビット幅が 1~3 ビットまでの結果を示す．表 2 より，画像ビット幅が 1 ビットの場合はいずれも遅延時間が 0.02 ナノ秒である．これは，いずれの出力も 1 段の AND ゲートのみで構成されているからである．そこで本研究ではいずれの出力も 1 段の AND ゲートで構成されている乗算器を用いて CNN の画像識別率を維持することを目指す．

表 2　3.1 節実験における画像データ 1~3 ビットの詳細

フィルタビット幅	画像ビット幅	識別率 (%)	面積 (gate)	遅延時間 (ns)
1	1	20.4	1	0.02
2	1	97.1	2	0.02
3	1	98.0	3	0.02
4	1	98.5	4	0.02
5	1	98.5	5	0.02
6	1	98.4	6	0.02
7	1	98.4	7	0.02
8	1	98.4	9	0.02
1	2	22.1	2	0.02
2	2	97.6	7	0.09
3	2	98.5	15	0.10
4	2	98.7	22	0.20
5	2	98.6	28	0.26
6	2	98.6	35	0.33
7	2	98.6	42	0.39
8	2	98.6	48	0.46
1	3	23.2	3	0.02
2	3	97.9	17	0.15
3	3	98.5	30	0.28
4	3	98.6	42	0.35
5	3	98.6	55	0.42
6	3	98.3	67	0.48
7	3	98.3	80	0.55
8	3	98.3	92	0.61

4. 近似乗算器の設計探索

本章では 3.1 節で設計した 64 個の乗算器および 3.2 節で設計した近似 3×2 乗算器より回路規模が小さく，遅延時間が短い乗算器の実現を目指す．前段階として 2 ビット乗算器の解析を行い，高い画像識別率を保ちながら遅延時間が AND ゲート 1 段分である近似乗算器の設計を目指す．

4.1 2 ビット乗算器の解析

本節では 2×2 ビット乗算器に対する解析について述べる．図 6 は正確な 2 ビット乗算器である．'a'はフィルタデータ，'b'は画像データ，'p'は積を表し'HA'は半加算器である．この乗算器に含まれる 4 つの部分積に対して部分積が 1 つ，2 つ，3 つ，4 つの場合について網羅的な解析を行う．設計した近似乗算器を MNIST 認識 CNN に適用し，画像識別率を算出する．また，回路面積と遅延時間から乗算器の評価を行う．実験環境や条件は 3.1 節と同様である．

表 3 に実験結果を示す．表より部分積 1 つの場合はどれも画像識別率が著しく低いことがわかる．部分積 2 個の場

Vol.2019-SLDM-189 No.24
Vol.2019-EMB52 No.24
2019/11/14

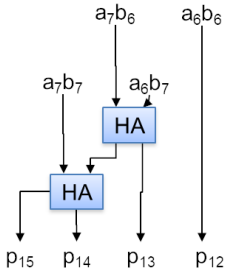


図 8　2 ビット乗算器

表 3　2 ビット乗算器の解析実験結果

部分積数	選択した部分積	識別率 (%)	面積 (gate)	遅延時間 (ns)
1	a_6b_6	43.7	1	0.02
	a_6b_7	17.8	1	0.02
	a_7b_6	63.6	1	0.02
	a_7b_7	20.4	1	0.02
2	a_6b_6, a_6b_7	69.5	2	0.02
	a_6b_6, a_7b_6	86.6	2	0.02
	a_6b_6, a_7b_7	94.9	2	0.02
	a_6b_7, a_7b_6	91.0	4	0.06
	a_6b_7, a_7b_7	97.1	2	0.02
	a_7b_6, a_7b_7	22.1	2	0.02
3	a_6b_6, a_6b_7, a_7b_6	90.5	5	0.08
	a_6b_6, a_6b_7, a_7b_7	90.0	3	0.02
	a_6b_6, a_7b_6, a_7b_7	96.9	3	0.02
	a_6b_7, a_7b_6, a_7b_7	97.5	6	0.08
4	a_6b_6, a_6b_7, a_7b_6, a_7b_7	97.6	7	0.09

合は a_6b_7, a_7b_7 を残した場合が最適である．これは 3.1 節の実験における 2×1 ビット乗算器と同じものである．部分積 3 個の場合は a_6b_7, a_7b_6, a_7b_7 を残したものが最適であるが回路面積と遅延時間が大きく増加している．部分積 4 個の場合は 3.1 節の実験における 2×2 ビット乗算器と同じものである．

実験結果から a_6b_7 と a_7b_7 で構成される近似乗算器が最も効率的であると考えられる．図 9 は a_6b_7 と a_7b_7 から構成される近似乗算器である．p_{14}, p_{13} は積の最上位から 2 ビット目と 3 ビット目である．結果からもう 1 ビット部分積を追加した近似乗算器が高い識別率を保てば回路面積 3 ゲートで遅延時間 0.02 ナノ秒の近似乗算器が設計できると考えられる．表 2 における 3×1 ビットが回路面積 3 ゲートで遅延時間 0.02 ナノ秒の近似乗算器だが，画像識別率は 98%である．したがって，p_{12} に出力される部分積の選択を行い，より良い画像識別率を目指す．

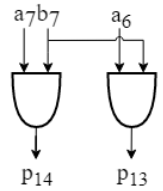


図 9　a_7b_7, a_6b_7 の回路図

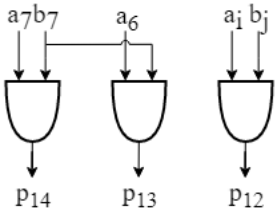


図 10　a_7b_7, a_6b_7, a_ib_j の回路図

表 4　3 つの部分積を用いた近似乗算器の設計探索実験結果

p_{12}	識別率 (%)	面積 (gate)	遅延時間 (ns)
a_4b_7	98.2	3	0.02
a_5b_6	98.4	3	0.02
a_5b_7	98.0	3	0.02
a_6b_5	96.6	3	0.02
a_6b_6	96.9	3	0.02
a_6b_7	96.2	3	0.02
a_7b_4	97.3	3	0.02
a_7b_5	97.4	3	0.02
a_7b_6	97.3	3	0.02
a_7b_7	97.3	3	0.02

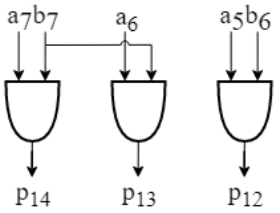


図 11　a_7b_7, a_6b_7, a_5b_6 の回路図

4.2　3 つの部分積を用いた近似乗算器の設計探索

図 8 に設計する近似乗算器の回路図を示す．a_ib_j に任意の部分積を挿入することで画像識別率の向上を目指す．実験では p_{11} より上位に関係する 10 個の部分積を p_{12} に出力し CNN の MNIST 画像識別率から最も良い近似乗算器の探索を行う．同時に回路面積と遅延時間も確認し，3 ゲート，0.02 ナノ秒の近似乗算器を目指す．実験環境や条件は 3.1 節と同様である．

実験結果を表 4 に示す．実験結果から a_5b_6 を選択した場

表 5　実験結果まとめ

	3×1	4×1	3×2	近似 3×2	改良近似 3×2
画像識別率(%)	98.0	98.5	98.5	98.4	98.4
回路面積(gate)	3	4	15	5	3
遅延時間(ns)	0.02	0.02	0.10	0.05	0.02

合が最も画像識別率が高く，正常な計算を行って画像識別した場合と同様の識別率が実現できることが分かった．また，いずれの近似乗算器も回路面積 3 ゲート，遅延時間 2 ナノ秒であることが確認できた．図 11 に a_7b_7, a_6b_7, a_5b_6 の回路図を示す.

表 5 に文献[16]の近似 3×2 ビット乗算器，3.1 節での各乗算器および本研究で設計した近似乗算器の比較を示す. 3.1 節における 3×1 ビット乗算器，4×1 ビット乗算器，3×2 ビット乗算器をそれぞれ 3×1，4×1，3×2 と示し，3.2 節の近似乗算器を近似 3×2 と示す. 本研究で設計した乗算器を改良近似 3×2 と示す. 本研究で設計した乗算器は正確な計算を行い識別した場合と同様の画像識別率を維持しつつ，過去の研究の乗算器より回路面積，遅延時間の両面において優れているか，同等であることが分かった.

5. おわりに

本稿では，文献[16]の簡潔な説明およびその改良により近似乗算器の設計探索を行った．手書き文字認識 CNN に近似乗算器を適用しその画像識別率，乗算器の回路面積，遅延時間のトレードオフからより良い近似乗算器の設計探索を行った．実験の結果，過去の研究で設計したどの乗算器よりも効率の良い近似乗算器を実現させることができた. 今後の課題としては FPGA などのハードウェアに実装することが挙げられる．また，より大規模な CNN への近似回路の適用が挙げられる.

謝辞　本研究は一部，科研費（15H02679・19H04081）の支援による.

参考文献

[1] J. Han and M. Orshansky, “Approximate Computing: An Emerging Paradigm for Energy-Efficient Design,” *IEEE European Test Symposium*, 2013.
[2] Q. Xu, N. Kim and T. Mytkowicz, “Approximate Computing:A Survey,” *IEEE Design & Test, Volume 33, Issue1, pp8-22,* 2015.
[3] B. Shao and P. Li, “Array-Based Approximate Arithmetic Computing: A General Model and Applications to Multiplier and Squarer Design,” *International Symposium on Low Power Electronics and Design (ISLPED),* 2014
[4] T. A. Drane, T. M. Rose, and G. A. Constantinides, “On the Systematic Creation of Faithfully Rounded Truncated Multipliers and Arrays,” *IEEE Transactions on Computers, vol. 63, no. 10, pp2513-2525,* Oct. 2014.
[5] T. Yamamoto, I. Taniguchi, H. Tomiyama, S. Yamashita and Y. Hara-Azumi, "A Systematic Methodology for Design and Worst-Case Error Analysis of Approximate Array Multipliers," *IEICE Trans. on Fundamentals, vol. E100-A, no. 7, pp. 1496-1499*, 2017.
[6] S. Boroumand, H. P. Afshar, P. Brisk and S. Mohammadi, “Exploration of Approximate Multipliers Design Space Using Carry Propagation Free Compressors,” *Design Automation Conference (ASP-DAC)*, *pp.611-616*, 2018.
[7] C. Liu, J. Han and Fabrizio Lombardi, “A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery,” *Design, Automation & Test in Europe Conference & Exhibition (DATE),* 2014.
[8] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek and K. Roy, “Design of Power-efficient Approximate Multipliers for Approximate Artificial Neural Networks,” *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016.
[9] I. Hammad and K. El-Sankary, “Impact of Approximate Multipliers on VGG Deep Learning Network,” *IEEE Access, Vol. 6*, 2017.
[10] H. Sim and J. Lee, “A New Stochastic Computing Multiplier with Application to Deep Convolutional Neural Networks,” *IEEE Design Automation Conference (DAC),* 2017.
[11] M. Courbariaux and Y. Bengio “BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1,” *arXiv*, 2016.
[12] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. Leong, M. Jahre and K. Vissers, “Scaling Binarized Neural Networks on Reconfigurable Logic, ” *ACM International Conference Proceeding Series (ACM-ICPS)*, 2017.
[13] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh and D. Marr, “Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC,” *International Conference on Field-Programmable Technology (FPT)*, 2016.
[14] Z. Li, L. Wang, S. Guo, Y. Deng, Q. Dou, H.Zhou and W. Lu, “Laius: An 8-bit Fixed-Point CNN Hardware Inference Engine,” *IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 2017.
[15] A. Balaji, S. Ullah, A. Das, and A. Kumar, ”Design Methodology for Embedded Approximate Artificial Neural Networks,” *Great Lakes Symposium on VLSI (GLSVLSI),* 2019.
[16] 白根健太, 山元貴普, 谷口一徹, 冨山宏之, "近似乗算器の手書き文字認識 CNN への適用事例,"信学技報, vol. 118, no. 457, VLD2018-95, 2019 年.
[17] A. Krizhevsky, I. Sutskever and E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *International Conference on Neural Information Processing Systems,* 2012.