C-007

エッジ AI による ResNet の高速化手法の提案
A Proposal for Accelerating ResNet Using Edge

杼森 雄己‡　中西 知嘉子‡
Yuki Tochimori　Chikako Nakanishi

1. はじめに

近年, AI 技術の発展に伴い, 特にエッジ AI への関心が高まっている. エッジ AI は小型端末であるエッジデバイスを用いることで, クラウドとの通信を必要とせず, 通信遅延が回避できる利点を有している. 一方で, エッジデバイス上で複雑な処理を高速に実行することは容易ではない.

そこで本研究では, エッジデバイスとして SoC FPGA ボードを使用し, 精度を維持しつつ, 処理の高速化を行った. SoC FPGA ボードとは CPU と FPGA が同一チップ上に統合されたデバイスであり, CPU で複雑な処理を行いつつ, FPGA で単純で膨大な演算処理を行い, 協調動作させることで処理の高速化を図る.

先行研究においては, 畳み込み層の入力データを 1 行ずつ処理する設計が採用されており, これにより回路の起動回数が増加し, 処理時間の増加を招いていた. また, データパス幅が 32bit に設定されており, FPGA 側で待機時間が発生し, 処理効率が低下していた. 本稿では, これらの課題に対して, 1 回あたりの回路のデータ処理単位の拡大とデータパス幅の拡張による処理時間短縮の効果について述べる.

2. 開発環境・使用モデル

本研究では, AI モデルとして ResNet50[4]を選定し, エッジデバイスとして SoC FPGA である Ultra96-V2[1]を使用した. また, 回路生成ツールとして Vitis HLS 2022.2[2]と Vivado 2022.2[3]を使用し, 推論ツールとして Ceras[5]を使用した.

2.1 Ultra96-V2 [1]

Ultra96-V2 は, AVENT 社より発売されている SoC FPGA ボードである. SoC には Xlinx 社の Zynq UltraScale+MPSoC が搭載されており, 同社の高位合成ツールで回路の設計・搭載を可能とする.

2.2 ResNet [4]

ResNet は, Kaming He 氏らが考案した ニューラルネットワークである. 残差学習を実現することで, 従来の CNN モデルで発生していた勾配消失問題を解決したモデルである. 本研究では, 精度や計算効率のバランスが良い ResNet50 を採用した.

2.3 Vitis HLS [2]

Vitis HLS は, 回路の設計データである IP を生成するための高位合成ツールである. 高位合成を用いることで, C/C++で回路コードを記述でき, 開発スピードの向上やデバッグの簡易化が期待できる.

2.4 Vivado [3]

Vivado は, 2.3 節で述べた IP を基に回路設計を行うためのツールである. 論理設計, シミュレーション, 合成, 配線配置, デバッグを統合しているため, 大規模な FPGA や SoC の開発に適している.

2.5 Ceras [5]

Ceras は当研究室で開発された C++ベースの AI 推論ライブラリである. Keras や ONNX などの深層学習モデルから構造情報および学習済みの重みをテキスト形式で抽出し, それを C++のプログラム上で読み込んで推論を実行する. Ceras は AI モデルの構造や演算処理が隠蔽されておらず, 各レイヤの処理を明示的に記述・制御できるため, FPGA との協調動作においてもデータ転送や回路起動のタイミングなどを細かく制御することができる. 本研究では, この Ceras を用いることで, 柔軟なハードウェア制御と高速な AI 推論処理の両立を実現した.

3. 実装手法

3.1 使用回路

使用回路は, 畳み込み処理および活性化関数処理を行うものである. FGPA に搭載可能なリソースには限りがあり, リソース使用量に考慮しつつ, 回路が保持するデータサイズを適切に設定した. 表 3.1.1 に本研究で設計した回路の仕様を示す.

表 3.1.1　回路の仕様

入力データサイズ上限	114
チャネル上限	112
カーネル上限	112

3.2 データ処理単位の拡大

3.2.1 従来手法

先行研究[6][7]では, 2 種類の回路が提案されている. 1 つは, 「まとめて処理する回路」であり, もう 1 つは「1 行ずつ処理する回路」である. まとめて処理する回路では, 入力データをブロック単位で FPGA に転送し, DMA 転送回数を削減することで, 回路の起動回数を抑えることが可能である. しかし, 入力データが回路の保持可能なデータサイズを超える場合, 回路からの出力が, ソフトウェア側が期待する出力順と一致しない場合があるため, 出力結果を正しい順に並び替える処理が必要となる. これにより, データ整形に要する時間が増加するという課題がある. 図 3.2.1.1 にまとめて処理する回路の仕組みを示す.

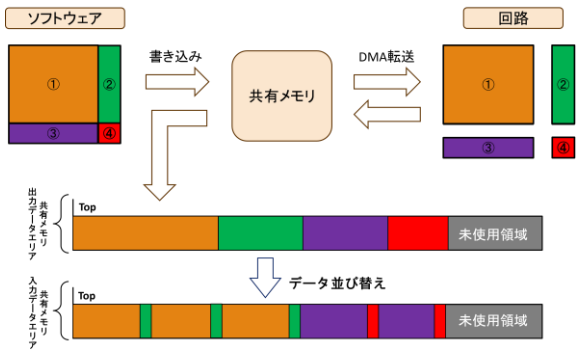

図 3.2.1.1　まとめて処理する回路の仕組み

一方, 1 行ずつ処理する回路では, 入力データの順序を保ったまま逐次処理が可能なため, 並び替えが不要になり, データ整形にかかる時間を抑えることができる. ただし, まとめて処理する回路よりも回路の起動回数が多くなり, それに伴うオーバヘッドが増加する点が課題である. 図 3.2.1.2 に 1 行ずつ処理する回路の仕組みを示す.

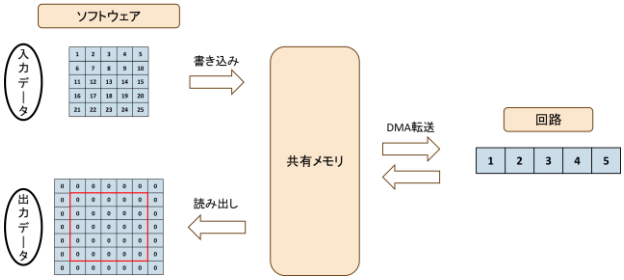

図 3.2.1.2　1 行ずつ処理する回路の仕組み

3.2.2 提案手法

本研究では, 1 行ずつ処理する回路をベースとして, 次層においてパディング処理が不要, かつカーネルサイズが 1×1 かつストライド 1 であるケースに対して, 複数行をまとめて処理する方式を導入した. なお, これらの条件を満たす畳み込み層は, ResNet50 などのモデルにおいて全体の約 60%を占める. FPGA のリソースである BRAM(Block RAM)の使用率を考慮し, 2 行または 8 行まとめて処理する回路を新たに設計した. 出力バッファとしては, 従来手法における out_buf[SIZE][KERNELS]を拡張し, 2 行処理時は SIZE=114 を踏襲, 8 行処理時は SIZE=56×8 とした. さらに, 従来の SET_ADD_RELU 処理は 1 行ずつ処理することを前提としていたため, 複数行処理に対応した新しいモードを追加実装した. 図 3.2.2.1 に複数行まとめて処理する回路の構造を示す.

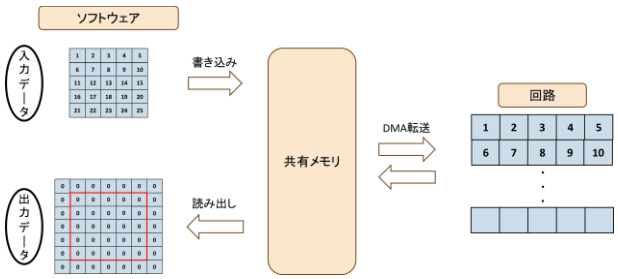

図 3.2.2.1　複数行まとめて処理する回路の構造

3.3 データパス幅の拡張

従来の回路では, ソフトウェアから回路への DMA 転送におけるデータパス幅は 32bit に設定されており, 回路の演算処理速度に比べて転送速度が律速となり, 回路側で待機状態が発生していた. 本研究では, データパス幅を 64bit に拡張することで, 1 回の DMA 転送で 32bit データを 2 つ同時に送信可能とした. これにより, 転送帯域が実質的に倍増し, データ待機による性能劣化を回避することが可能となった. この拡張に伴い, 従来は 9 つのデータが pixel[9]に格納されてからに処理が行われていたが, 新方式ではデータが一度に 2 つずつ送られるため pixel[CHANNELS]として全チャネル分を一括で格納する方式に変更し, 処理を最適化した.

4. 検証方法

本研究では, 提案手法の有効性を評価するために, 従来手法(1 行ずつ処理する回路)と, 提案手法(2 行または 8 行まとめて処理する回路)を Ultra96-V2 上で実行し, 回路の起動回数, 全体の推論時間, BRAM の使用率を用いて, 処理効率およびリソース使用率のバランスについて定量的に評価した.

5. 結果

表 5.1 に従来手法および提案手法の各回路における評価結果を示す.

表 5.1　評価結果

	従来手法	提案手法	
	1行ずつ処理	2行まとめて処理	8行まとめて処理
回路起動回数 (回)	17443	11468	6768
全体推論時間 (ms)	4269	3992	3881
BRAM使用率 (%)	77.08	77.31	99.54

表 5.1 の結果から, 2 行まとめて処理する回路では, 従来手法と比較して回路の起動回数を 5795 回削減でき, それにより推論時間は 277ms 短縮された. BRAM の使用率増加はわずか 0.23%に抑えられている. 8 行まとめて処理する回路では, 回路の起動回数を 10675 回削減し, 推論時間を 388ms 短縮することができた. 一方で, BRAM の使用率は 22.46%増加しており, 大きなリソースの追加消費が発生している.

6. まとめ

本研究では, FPGA を用いたエッジ AI 推論処理において, 1 回あたりの回路のデータ処理単位の拡大と, データパス幅の拡張による高速化を実現した. これにより, 従来手法と比較して回路の起動回数を削減し, 全体の推論時間を短縮することができた.

今後の展望として, 現状は out_buf を用いて回路側で処理データを保持しているため, リソースに限界がある. そのため, 処理データをソフトウェア側で保持することで, データ転送量を増加させることによる処理の高速化を考えている. さらに, 現在は浮動小数点で実行している畳み込み処理を, 量子化によって 8bit 固定小数点に変換し, FPGA 回路上で実行することで, 演算処理の軽量化とさらなる高速化を目指している.

参考文献

[1]AVENT, “Ultra96-V2”, https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2
[2]Xilinx, “Vitis 高位合成ユーザーガイド”, https://docs.amd.com/viewer/book-attachment/ZeLAYVm_EbESUUvSSCw7ug/~sx1OMuElsL9bmdsdTRxUw-ZeLAYVm_EbESUUvSSCw7ug
[3]Xilinx, “Vivado Design Suite ユーザーガイド”, https://docs.amd.com/viewer/book-attachment/ohhmDx4JTEfoJYbY2kZ9Kw/0mrbZtHZ0mK8lkc8GsQQJg-ohhmDx4JTEfoJYbY2kZ9Kw
[4]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”, arXiv preprint, https://arxiv.org/pdf/1512.03385
[5]西岡駿, 中西知嘉子, “機械学習ライブラリの C 言語化の実現”, 電子情報通信学会ソサイエティ大会(2021)
[6]大戸彰馬, SoCFPGA によるディープラーニングのリアルタイムの実現手法の検討
[7]田嶋夏己, 中西知嘉子, “SoC FPGA を用いたエッジ AI の推論処理の高速化手法の検討”, 電子情報通信学会総合大会(2024)

† 大阪工業大学 情報科学研究科 情報科学専攻
Graduate School of Information Science and Technology
Osaka Institute of Technology
‡ 大阪工業大学 情報科学部 情報知能学科
Department of Information and Computer Science
Osaka Institute of Technology