C-006

深層学習モデル「RegNet」のエッジデバイスへの実装の検討

Examination of implementation of deep learning model "RegNet" on edge devices

田嶋夏己 † 中西知嘉子 ‡
E-mail †m1m22a22@st.oit.ac.jp ‡chikako.nakanishi@oit.ac.jp
Natsuki Tajima Chikako Nakanishi

1. はじめに

近年，AI 技術が注目されている．現在はクラウド上の高性能端末にデータを送信し，推論を行うクラウド AI が主流である．しかし，膨大なデータ通信が必要な点，それに伴うセキュリティへの懸念など通信面での不安が大きい．それに対し，エッジデバイス上で完結して推論を行うエッジ AI は，ネットワーク環境に左右されず，安定した推論が可能となっている．また，エッジデバイス上でデータを処理するためセキュリティへの懸念も少ない．しかし，エッジデバイスは性能面で大きく劣っている．そのため，複雑な処理が必要とされる高精度な AI を，リアルタイムに処理させることは難しい．

本研究では，高精度な深層学習モデルとして，RegNet[1]を採用し，エッジデバイスとして SoC FPGA ボードである Ultra96V2[2]を用いる．SoC FPGA の特徴である FPGA による回路と CPU との協調動作を用いて推論処理の高速化を行いリアルタイムな推論を行うエッジ AI の実装を行う．

2. 回路化する処理の決定

本研究では，RegNet を使用する．RegNet は 2020 年に発表されたシンプルなネットワーク構造を持つ高速かつ高精度な深層学習モデルである．今回はリアルタイムな推論を目的としているため最も小さいモデルである RegNetX-200MF を採用した．

RegNet は Python 言語で動作する Pytorch モデルである．しかし，FPGA による回路の設計方法として高位合成を使用するため，C++言語で AI の推論を行えるライブラリである Ceras[3]を使用する．Ceras を用いて RegNetX-200MF の各層の処理時間を Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz が搭載されたマシンで測定した結果を表 1 に示す．

表 1　層ごとの処理時間と割合

層の種類	処理時間	割合
Conv2D	1047.5ms	95.32%
GroupConv2D	42.1ms	3.83%
Relu	3.6ms	0.03%
Add	3.6ms	0.03%
その他	2.1ms	0.02%

表 1 より，畳み込み演算を行う層である Conv2D 層，GroupConv2D 層が全体の約 99%を占めていることが分かった．そこで，Conv2D 層，GroupConv2D 層を高速化する回路を設計することとした．

次に，Conv2D 層と GroupConv2D 層の詳細を分析した．Conv2D 層の処理の詳細を表 2，GroupConv2D 層の処理の詳細を表 3 に示す．

表 2　Conv2D 層の詳細

カーネルサイズ	ストライド	層数
1	1	26
1	2	4
3	2	1

表 3　GroupConv2D 層の詳細

カーネルサイズ	ストライド	層数
3	1	9
3	2	4

表 2，表 3 より，Conv2D 層と GroupConv2D 層ではカーネルサイズ 1 のストライド 1 と 2，カーネルサイズ 3 のストライド 1 と 2 の畳み込み演算が行われていることが分かる．この分析結果からカーネルサイズ 1 のストライド 1 と 2，カーネルサイズ 3 のストライド 1 と 2 の畳み込み演算を回路化することとした．

3. 設計

3.1. 畳み込み演算

畳み込み演算とは，カーネルと同サイズの入力データの部分データに注目し，部分データとカーネルとの積和演算を行い，バイアスを加算する処理である．入力データの部分データの位置をずらす幅のことをストライドという．畳み込み演算では，入力データ，カーネルデータ，バイアスデータの 3 つのデータを使用する．例として，図 1 に入力データサイズが 5x5，カーネルサイズが 3x3，ストライドが 2 の畳み込み演算の流れを示す．図 1 では 1 チャネルのみの計算となっているが，複数チャネル存在する入力データを処理する場合はカーネルとの積和結果をチャネル方向に加算を行い，最後にバイアスを加算する．

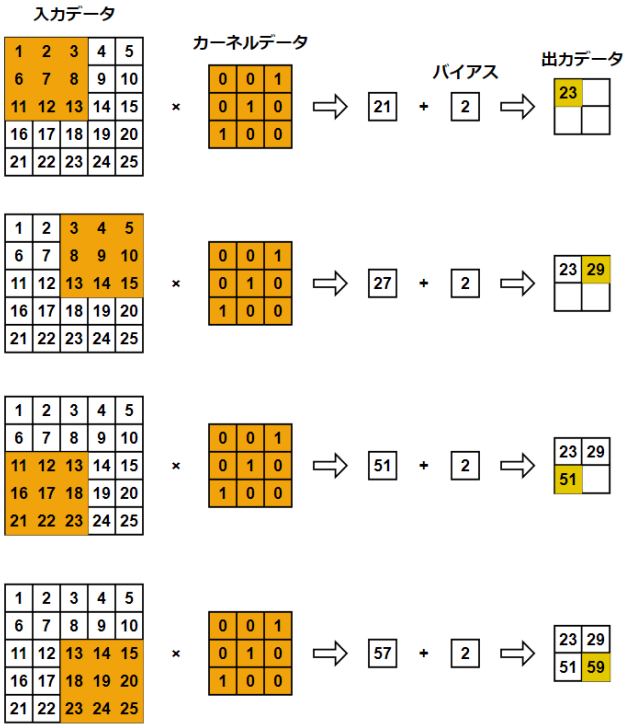

図 1　畳み込み演算の流れ

3.2. GroupConv2D 層

GroupConv2D 層とは Conv2D 層の処理を行う前に入力データのチャネルをグループ個に分割し，分割したそれぞれのデータに Conv2D 層と同じ処理を行い，それぞれの結果を最後に合成する層である．図 2 に GroupConv2D 層の処理の流れを示す．

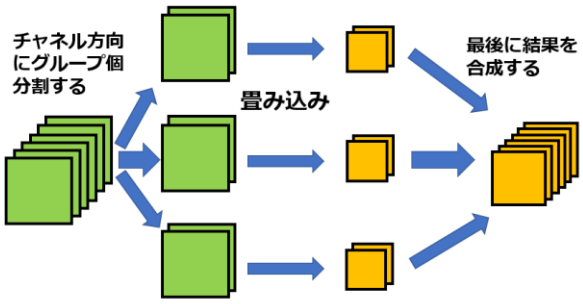

図 2　GroupConv2D 層の処理の流れ

3.3. ベース回路

本研究では，RegNet に存在する 4 種類の畳み込み演算を回路化するためのベース回路として，先行研究[4]で作成された回路を使用した．先行研究[4]で作成された回路が対応している畳み込み処理の種類を表 4 に示す．

表 4　ベース回路が対応する畳み込み演算

	ストライド 1	ストライド 2
カーネルサイズ 1	○	×
カーネルサイズ 3	○	○

表 4 より，RegNet に存在する畳み込み演算の内，カーネルサイズ 1 のストライド 2 には対応していないことが分かる．よって，この回路をカーネルサイズ 1 のストライド 2 に対応できるよう設計した．

3.4. 回路設計

回路設計を行うツールとして，C++言語から RTL への自動変換を行う高位合成ツールである Vivado HLS 2019.2，高位合成から得られた RTL を回路に組み込むためのツールとして Vivado 2019.2 を用いた．

カーネルサイズ 1，ストライド 2 の畳み込み演算の特徴として，入力データの内，畳み込み演算に使用しないデータが存在する．例として，図 3 に 7x7 の入力データを処理する時に使用するデータを示す．

図 3　カーネルサイズ 1，ストライド 2 で使用するデータ

図 3 より，入力データの内，使用しないデータが多く存在することが分かる．よって，入力データの内，使用しないデータを読み飛ばす設計をした．読み飛ばす処理を追加する際に注意した点として，処理の追加位置がある．適切でない位置に処理を追加すると，カーネルサイズ 3 の畳み込み演算の速度が低下する恐れがある．図 4 に処理を追加した周辺の動作フローを示す．

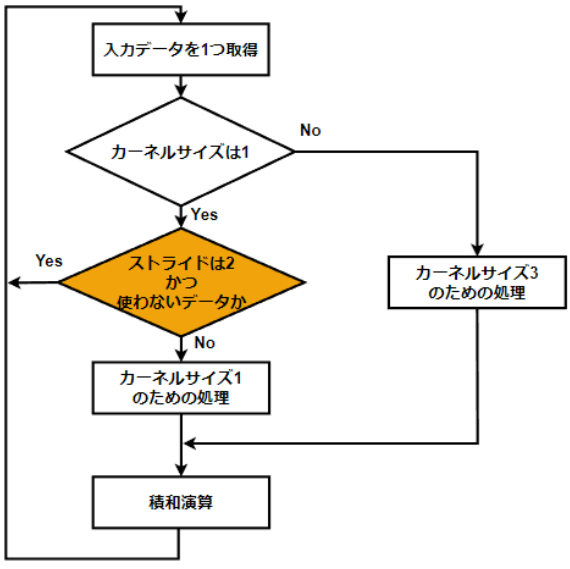

図 4　処理を追加した周辺の動作フロー

図 4 より，カーネルサイズ 3 の畳み込み演算には影響がないことが分かる．

次に，RegNetX-200MF のすべての Conv2D 層，GroupConv2D 層に対応するために必要なパラメータのサイズを表 5 に示す．

表 5　すべての層に対応するために必要なパラメータのサイズ

	必要サイズ
入力データ	368x224x224
カーネルデータ	368x368x3x3
バイアスデータ	368
出力データ	368x112x112

表 5 より，RegNetX-200MF のすべての Conv2D 層，GroupConv2D 層の畳み込み演算を回路で実行するために入力データが 368x224x224，カーネルデータが 368x368x3x3，バイアスデータが 368，出力データが 368x112x112 のサイズに対応可能な回路が必要となる．しかし，このようなサイズの回路規模の回路は FPGA のリソース不足により実装できない．そのため，今回作成した回路では入力データ，カーネルデータ，入力データとカーネルデータのチャネル数の入力サイズの上限を設定した．表 6 に各データの上限を示す．

表 6　データの上限サイズ

	上限サイズ
入力データの縦横	66
カーネルデータ	128
カーネルデータ，入力データのチャネル	64(カーネルサイズ 1 は 576)

3.5. データ転送

転送するデータは float 型であるため，CPU と回路間のデータバス幅は 32bit とした．また，PL 部に供給するクロック周波数は上限の 300MHz とした．

CPU と FPGA 間でデータのやり取りを行うために共有メモリを使用し，共有メモリから FPGA へのデータ転送は DMA 転送を採用した．DMA 転送とは，CPU を介さずに FPGA と共有メモリ間でデータ転送を行う方式である．DMA 転送には連続したメモリ領域を転送する simple DMA，不連続なメモリ領域を転送する Scatter Gather DMA が選択できる．Scatter Gather DMA は高速なデータ転送が可能であるが設計が複雑であり，使用リソースが増加する．よって Simple DMA を採用した．

Simple DMA でのデータ転送のために連続したメモリ空間を確保するために User space mappable DMA Buffer(udmabuf)[5]を採用した．udmabuf はカーネル空間の連続したメモリ空間を DMA バッファとして確保し，ユーザ空間からアクセス可能にするデバイスドライバである．データ転送の概要を図 5 に示す．

図 5　データ転送のイメージ

3.6. データ分割

Conv2D 層，GroupConv2D 層のサイズによっては，回路に転送できるデータサイズの上限を超える．そのため，データサイズの上限を超える Conv2D 層，GroupConv2D 層を処理する場合はデータを分割して回路にデータを転送する．例として，カーネルサイズ 1 の Conv2D 層に入力データの縦横サイズが 100 のデータを入力した際の処理を図 6 に示す．

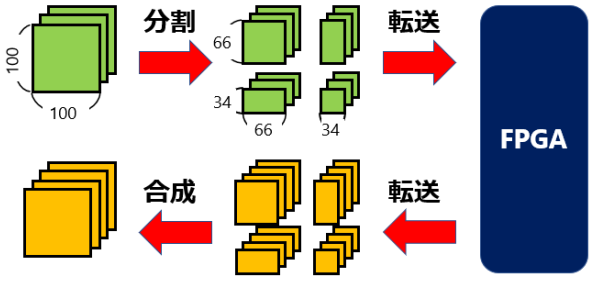

図 6　分割処理のイメージ

図 6 の処理の流れを以下に示す．

1. バイアスデータを共有メモリに書き込む
2. カーネルデータを共有メモリに書き込む(チャネル上限より大きい場合は上限まで書き込む)
3. 分割した入力データを共有メモリに書き込む(チャネルが上限より大きい場合は上限まで書き込む)
4. DMA 転送を用いて FPGA にデータを転送する
5. 処理されたデータを共有メモリから読み取る
6. 3~5 の処理を分割したデータ数分繰り返す
7. (カーネルデータが上限を超えた場合は 1~6 の処理を繰り返す)

以上の流れで回路に転送するデータの分割処理を実装した．カーネルデータやバイアスデータに関しては，フラグを用いて転送の制御を行う．

3.7. キャッシュ管理のハードウェア化

通常では，ソフトウェア側で行われるキャッシュとメモリ上の DMA バッファの一貫性を保つ処理をハードウェアに行わせることで，共有メモリへの読みだしの高速化を行う．キャッシュ管理をハードウェアで行うために変更したブロック図の 1 部を図 7 に示す

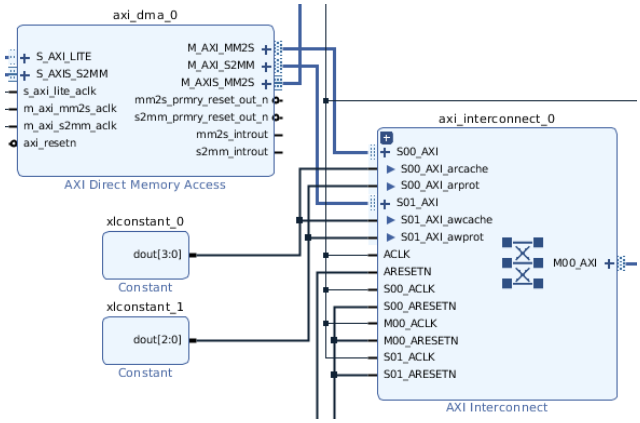

図 7　変更後のブロック図

4. 検証

検証は Ultra96V2 上で行い，CPU のみで推論を行った場合と CPU と回路で推論させた場合の推論時間で行った．検証結果を表 7 に示す．

表 7　処理時間

	CPU のみ	回路使用
推論	11750.0ms	943.1ms
畳み込み層	11135.0ms	637.0ms
グループ化畳み込み層	505.2ms	201.4ms

表 7 より，CPU のみで推論を行った場合の推論時間が 11750.0ms，回路を使用した場合が 943.1ms と約 12.5 倍高速に動作した．Conv2D 層については 11135.0ms から 637.0ms と約 17.5 倍の高速，GroupConv2D 層については約 2.5 倍高速に動作した．次に，回路を使用して推論を行った場合の回路実行の処理時間，共有メモリへの読み書きの処理時間を表 8 に示す．

表 8　回路と共有メモリへの読み書きの処理時間

回路実行	297.3ms
共有メモリ書き込み	334.6ms
共有メモリ読み込み	80.2ms

表 8 より，回路実行に推論時間の約 31%，共有メモリへの書き込みに推論時間の約 35%，共有メモリへの読み込みに推論時間の約 8%を占めていることが分かった．

5. 考察

検証結果から，回路と CPU による協調動作により，推論時間を 12.5 倍高速にできることが分かった．しかし，共有メモリへの書き込みに推論時間の約 35%を占めており書き込み時間への改善が必要だと考えられる．また，Conv2D 層の処理時間に関しては約 17.5 倍と大幅な高速化に成功したが，GroupConv2D 層に関しては約 2.5 倍とあまり高速化できていない．この原因として GroupConv2D 層は層のチャネル数が少ないため，現在のデータ分割方法では回路のリソースを上手く使えておらず，あまり高速にならなかったと考えられる．

6. 結論

本研究では，エッジデバイスである Ultra96V2 上で FPGA による回路と CPU による協調動作によって RegNetX-200MF の推論の高速化を行った．CNN の中から処理時間が長いものを回路化，キャッシュ管理のハードウェア化を行うことで推論時間を 12.5 倍高速にすることが出来た．

今後の展望として，GroupConv2D 層などのチャネルが少ない層の処理の改善，共有メモリへの書き込みの改善，さらなる処理の並列化が挙げられる．

参考文献

[1] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, Piotr Dollár,” Designing Network Design Spaces” arXiv preprint arXiv:2003.13678(2020).
[2] https://japan.xilinx.com/products/boards-and-kits/1-vad4rl.html
[3] 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).
[4] 大戸彰馬, 中西知嘉子, "推論処理における畳み込み処理の回路化の検討",電子情報通信学会総合大会(2022).
[5] https://github.com/ikwzm/udmabuf/blob/master/Readme.ja.md

† 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