C-012

# ResNet のエッジデバイス上での動作高速化手法の検討
# Investigation of methods to accelerate ResNet operation on edge devices

岡本 遥仁† 中西 知嘉子‡
Haruto Okamoto Chikako Nakanishi

## 1. はじめに

近年，注目を集めているのがエッジAIである．現在，広く利用されているのはクラウドAIと呼ばれる方式であり，これはデータをクラウド上の端末に送信し，推論を行うものである．しかし，クラウドAIは推論の際に膨大なデータ通信が必要となり，それに伴うセキュリティ上の懸念など通信面での不安が大きい．これに対し，エッジAIはエッジデバイスで推論を行うことから，ネットワークへの依存度を低減し，通信遅延やセキュリティに関する懸念を軽減できる．ただし，エッジAIは処理を行うデバイスがクラウドAI に比べて性能面で劣る．そのため，高精度な AI に必要な複雑な処理をエッジデバイスに搭載し，リアルタイムに処理させることは難しい．

そこで本研究では，深層学習モデルの 1 つである「ResNet[1]」に焦点を当て，ハードウェアを活用して処理を高速化することを最終目標とする．具体的には，従来の手法では，パディング処理をソフト部で行っていたが，これを回路で行うことで高速化を目指す．また、ResNet には異なるパラメータ数を持つ複数のモデルが存在するが，今回は精度とモデルサイズのバランスが良い ResNet50 を選択し，研究を進めることとする．また，ResNet を実装する際にエッジデバイスとして，SoC FPGA ボードである Ultra96v2[2]を採用する．

## 2. 使用デバイス・AI モデル・開発環境

### 2.1 ResNet

ResNet(Residual Network)とは、Microsoft 社の Kaming He 氏らが考案したニューラルネットワークモデルである。ResNet は残差学習を提案したモデルであり、層を深く出来ないという劣化問題を解決したモデルである。本研究では，精度とモデルサイズのバランスが良い ResNet50 を採用した．

### 2.2 Ultra96-V2

Ultra96-V2 とは、AVNET 社より発売されている Arm コア内蔵の SoC FPGA ボードである。SoC FPGA は，CPU と FPGA が同一チップ上に存在するデバイスである．特徴として、CPU-FPGA 間の高速なデータ転送が可能な点、高効率な専用回路を作成することで FPGA による高並列で高速な処理を行える点などが挙げられる。先行研究[3]では、CPU で全体の処理を行いつつ、Conv2D 層などの高負荷な演算を FPGA に実装した回路で行うことで高速化を図った。

### 2.3 開発ツール

#### 2.3.1 回路生成ツール

回路は C++言語で記述したコードに高位合成を使用して RTL 言語に変換することで作成した．また、高位合成を行うツールとして Vitis HLS2022.2[4]を，回路生成を行うツールとして Vivado Design Suite[5]を使用した．

#### 2.3.2 Ceras[6]

Ceras は弊研究室にて開発された学習済み ONNX モデルの推論を C++言語で実行するライブラリである.. C++言語の標準モジュールのみで構成されており，環境構築が不要な点や，層内部の処理単位にまで着目できる点，開発環境を C++言語で統一できる点から使用した．本研究では，Ultra96-V2 上で動作させるモデルに対し，用いている．

## 3. ベース回路

今回用いる先行研究で作成された回路は，主に Conv2D 層の処理を高速に行う回路であり，ResNet に存在する Conv2D 層，Add 層と ReLU 層を統合した AddReLU 層に対応している．また，CPU による制御で回路を実行できるよう設計されている．

回路からの出力は，CPU と回路でデータを共有する場合には，CPU と回路の両方がアクセス可能なバッファ領域である共有メモリを使用する．

また，この回路は，回路で処理可能な層が連続する場合，CPU を介さずに前の層の出力データを再利用可能であり，これにより，ソフトウェアと回路間でデータのやり取りをする共有メモリの読み書きが少ないという特徴がある．

推論を行う上で，出力サイズの調整や入力データの端の特徴を取り入れるために入力データの周りをゼロで埋めるパディング処理が存在する． 図 1 にパディング処理のイメージ図を示す．次の層でパディングが必要な場合，その前の層で図のようにパディング付きの出力を生成する．

具体的には，パディング付きの出力は，CPU のプログラム上でパディング部分にあらかじめ 0 を埋め，回路から 1 行ずつパディング部分を飛ばして，共有メモリ上にデータを転送して生成している．

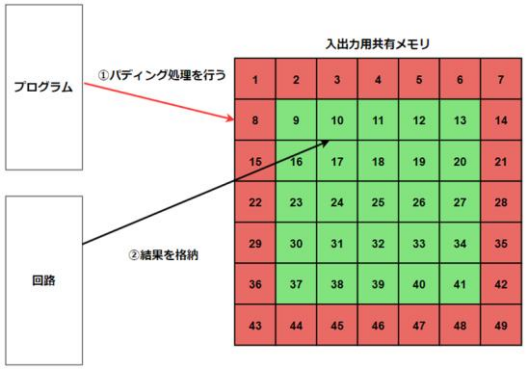


図 1 パディング処理のイメージ図

### 3.1 入力方法

この回路では，入力データを 1 行ずつ読み込みながら畳み込み演算を行う構成となっている．カーネルサイズが 3x3 でパディング処理が必要な場合は，まず，前述の通り，前の層で パディング処理行う．次に，回路部では，2 行分を 1 行分の入力データを格納するラインバッファに格納し，

† 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



Copyright © 2025 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

3 行目以降は畳み込み演算部に順次データを入力しながら演算を実行する．図 2 は，入力データが 3x3，カーネルサイズが 3x3 の場合のパディング処理と畳み込み演算の流れを示している．図から，1 チャネル分の 3x3 のデータを格納するウインドウバッファを用いて，入力データを 1 行ずつ読み込みつつ，バッファ内のデータをずらしながら畳み込み演算を行っていることが分かる．この処理を入力データの最後まで繰り返すことで，畳み込み演算を完了する．

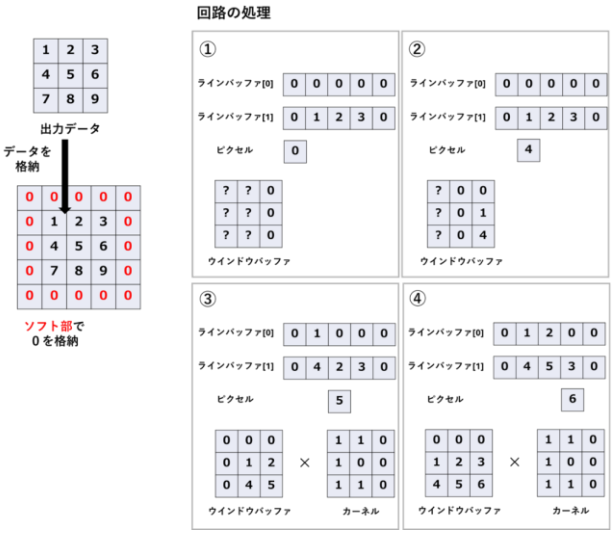


図 2 カーネルサイズが 3x3 の時のパディング処理と畳み込み演算

### 3.2 回路の課題点

前述のパディング処理方法では，パディングが必要な層の前の層で，CPU のプログラムでパディング処理を行っている． 回路は，演算結果を共有メモリ上に連続で転送を行う．このため，このパディング処理方法では，0 が埋まっている部分を飛ばして回路から出力しなければならず，回路から 1 行ずつ転送する必要がある．

## 4. 提案手法

前述の通り，従来の手法では，パディング処理を CPU で実施していたため，回路はパディング部分を除いて，データを 1 行ずつ出力し，行数分回路を起動して転送を行う必要があった．このため，連続的な演算や出力できないという課題があった．

そこで本提案では，連続した演算を可能にできるようにするための第 1 段階として，このパディング処理を回路内で実施する． 図 3 に回路内での処理を示す．従来の手法では，すでにパディングされた入力データを 1 行ずつラインバッファに格納していた．一方，提案手法では，1 行目の入力タイミングで，ラインバッファ[0]に 0 を格納し，1 行目のデータは，両端に 0 を加えてラインバッファ[1]に格納するように変更した．3 行目からは，図のように，入力データの横幅と入力されたデータの個数からデータの両端を判定し，両端には 0 を格納しながら順番にデータを読み込んでいき，ウインドウバッファを更新していく，これをカーネルと掛け合わせ，畳み込み演算を行う．図 4 には，下段パディング処理のイメージ図を示す．下段パディングでも同様に，0 を格納しながら．畳み込み演算を行う．

これにより，前の層の出力結果をそのまま入力し，パディング処理を回路内で実行可能にした．

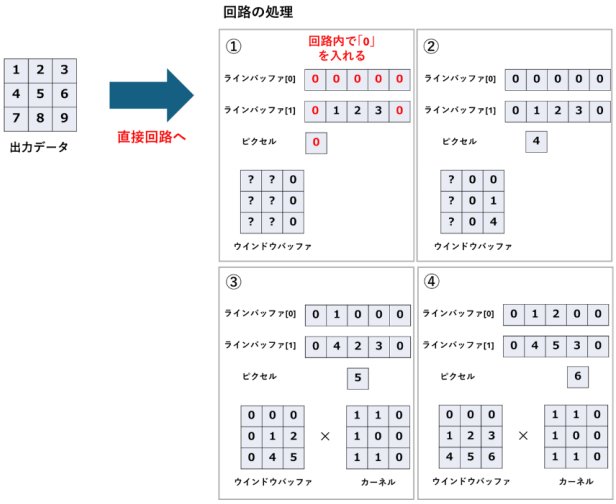


図 3 回路内でのパディング処理のイメージ図

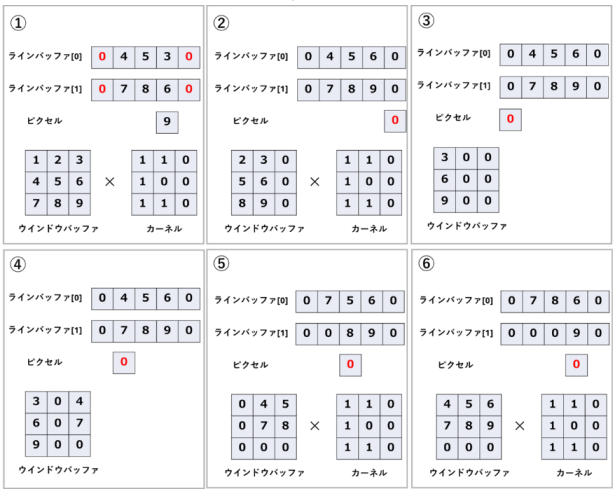


図 4 下段パディング処理のイメージ図

## 5. 結果

図 5 は，従来の手法と提案手法の推論処理時間を比較したものを示す．図 5 より、パディング処理を回路で行ったが，まだ，回路には 1 行ずつデータを入力する方法で，演算を行っており，連続で演算を行うことはしていないため，ほとんど処理速度は変化していないことが分かる．

今後，まとめて演算を行うことができれば，更なる高速化が期待される．

| 従来の手法 | 提案手法 |
|---|---|
| 4502.4 | 4479.3 |

図 5 推論時間の比較

### 参考文献

[1] Deep Residual Learning for Image Recognition <https://arxiv.org/abs/1512.03385> (2025/01/6 参照)
[2] AVNET, ULTRA96-V2 <https://www.avnet.com/opasdata/d120001/medias/docs/193/5365-pb-ultra96-v2-v4a.pdf>(2025/01/6 参照)
[3] 田嶋夏己，中西知嘉子，“SoCFPGA による深層学習モデル「RegNet」高速化手法の検討”，FIT2023(2023).
[4] <https://japan.xilinx.com/member/forms/download/xef.html?filename=Xilinx_Unified_2022.2_1014_8888_Lin64.bin>
[5] <https://japan.xilinx.com/member/forms/download/xef.html?filename=Xilinx_Unified_2022.2_1014_8888_Lin64.bin>
[6] 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(2021)

† 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



Copyright © 2025 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.