C-019

エッジ端末を活用した物体検出モデル「YOLOv7」の高速化手法の検討と動作比較

Study and comparison of methods for speeding up the object detection model YOLOv7 using edge terminals

舩橋 駿介‡　中西 知嘉子‡
Shunsuke Funahashi　Chikako Nakanishi

1. はじめに

近年，エッジ AI が注目を集めている．エッジ AI は小型な端末のみで推論処理を行う AI である．通信環境を必要としないため，推論処理の際に通信遅延が無く，セキュリティが高いなど様々な利点がある．しかし，エッジ AI を搭載する機器は性能が低いため，速度と精度を両立することは難しい．

そこで我々は，SoC FPGA を活用し，ソフト部(CPU)と回路部(FPGA)を，共有メモリを介して協調動作させることで，エッジ AI を高速化させる手法を検討している．物体検出モデルの YOLOv7[1]を対象に，先行研究[2]で開発された畳み込み処理演算回路をベース回路とし，高速化を試みた[3]．その結果，データ共有に必要な処理は更なる改善の余地があることが分かった．そこで，本研究では更なる高速化のための手法を提案する．また，異なる手法で YOLOv7 を動作させ，本手法と比較を行うことにより，本手法の成果を評価する．

2. 使用機器・使用ツール・使用ネットワーク

本研究で使用した機器は Avnet 社からリリースされた Ultra96-V2[4]である．使用ツールは C++を用いて高位合成[5]を行う Vitis HLS 2020.2 と回路部設計に用いる Vivado Design Suite 2020.2 である．高速化の対象とする深層学習モデルは YOLOv7 の中で最も演算量が少ない YOLOv7-tiny である．ソフト部の処理では Vitis HLS に合わせ C++で推論処理を行う Ceras[6]というライブラリを使用する。

3. 開発手法と検討

3.1 特化回路

まず，先行研究で作成された畳み込み演算処理回路をベース回路とし，YOLOv7-tiny に特化した回路を作成した．表 3.1.1 に特化回路を使用した際の処理時間の内訳を示す．表 3.1.1 の回路動作とデータ共有は Conv2D 層処理の内訳となっている．回路動作は特化回路の処理時間，データ共有はソフト部から共有メモリに対するデータの書き込み・読み出し時間の合計を表している．また，割合にて括弧内に書かれている数値は Conv2D 層の処理における内訳である．

表 3.1.1　適用済み手法の処理時間

名称	処理時間(ms)	割合(%)
推論全体	7879.1	100
Conv2D 層 & LeakyReLU 層	5275.7	66.9 (100)
回路動作	2942.1	(55.8)
データ共有	2073.1	(39.3)

表 3.1.1 より，Conv2D 層の演算結果とは関係のないデータ共有に 2 秒以上かかっていることが分かった．

3.2 共有メモリの活用

3.2.1 既存手法

3.1 節より，ソフト部と共有メモリのデータ共有に 2 秒以上かかっていた．SoC FPGA における共有メモリとはソフト部と回路部が共有してアクセスすることができるメモリ領域である．ソフト部から共有メモリ内に存在するデータにアクセスする際はアドレスで直接指定する必要がある．本研究では ソフト部と回路部の協調動作のために利用しており，回路部へデータを入力する際．回路部へデータ入力する際，ソフト部のデータを共有メモリへ書き込み，共有メモリから回路部へ DMA を用いた転送を行う．また，回路部の出力データには反対の操作を行う．図 3.2.1.1 に Conv2D 層が連続する場合の共有メモリ使用イメージを示す．Conv2D 層が連続する際には 1 層目の出力データと 2 層目の入力データが等しいにも拘わらず，共有メモリからソフト部へ読み出す必要があり二度手間となっていた．

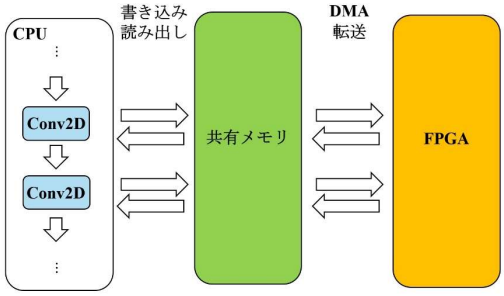

図 3.2.1.1　Conv2D 層の共有メモリ使用イメージ

そこで，先行研究では回路部の出力データをソフト部へ読み出さず，共有メモリ内にあるデータを回路へ転送するように設計した．（以下「共有メモリ活用手法」とする）図 3.2.1.2 に共有メモリ活用手法改善イメージを示す．この手法では共有メモリ上に存在する 1 つ目の層の出力データでかつ 2 つ目の層の入力データをアドレス指定で直接操作することにより，データ共有時間を削減することができる．

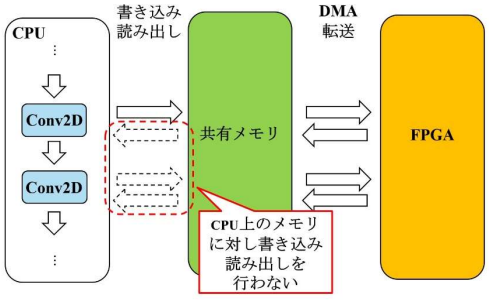

図 3.2.1.2　共有メモリ活用手法改善イメージ

また，Conv2D 層の間に他の層が存在する場合はソフト部上のメモリに対して処理を行う必要があるため，一度共有メモリからソフト部へデータを読み出す必要があった．図 3.2.1.3 に Conv2D 層の間に他の層が存在する場合の共有メモリ使用イメージを示す．

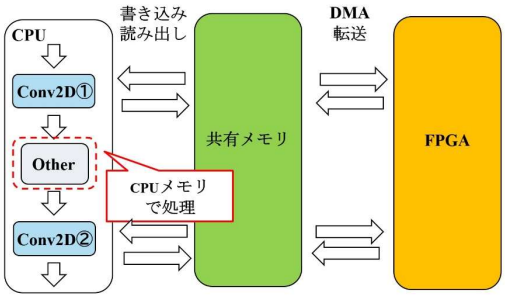

図 3.2.1.3　Conv2D 層以外の共有メモリ使用イメージ

そこで，ソフト部で処理を行う層において，直後の Conv2D 層の出力データを共有メモリから直接読み出して演算し，演算結果を直接共有メモリへ書き込むことによってデータ共有時間を削減した．共有メモリ活用手法を適用した際の処理時間を表 3.2.1.1 に示す．

表 3.2.1.1　共有メモリ活用手法適用後の処理時間

名称	適用前(ms)	適用後(ms)
推論全体	7879.1	6341.4
データ共有	2073.1	1021.4

表 3.2.1.1 より，データ共有時間にまだ 1 秒以上かかっていることが分かった．そこで，共有メモリ活用手法の適用範囲を拡大する．

3.2.2 新規手法

使用する特化回路にはリソースの都合で一度に処理することができるデータ量に上限を設定している．上限を超えた場合は入力データを分割して回路部へ転送する必要があり，分割処理をソフト部で行っていたため，共有メモリからソフト部へ一度データを戻す必要があった．そこで，データの分割処理を共有メモリ内で直接行うことにより，ソフト部へデータを戻す必要性を無くし，データの共有時間を削減した．上記の手法は，データが 2 分割される場合は先行研究にて適用しており，本研究ではデータが 2 分割を超える場合を対象に適用範囲を拡大している．また，YOLOv7-tiny の中で現在共有メモリ活用手法を適用していない層として MaxPooling 層が存在する．そこで，MaxPooling 層の処理を共有メモリ内で行い，データ共有時間を削減した．

4. 評価と結果

4.1 高速化手法適用の結果

共有メモリ活用手法の範囲を拡大した際の処理時間を表 4.1.1 に示す．

表 4.1.1　手法適用後の処理時間

名称	適用前(ms)	適用後(ms)
推論全体	6341.4	4804.9
データ共有	1021.4	638.1

表 4.1.1 より，全体の推論時間は本手法適用前と比較して約 1.3 倍高速化することができた．また，データ共有時間に関しては 1.6 倍高速化することができた．

4.2 異なる手法との比較評価

本手法の評価を行うために異なる手法で YOLOv7-tiny を動作させ，処理時間と最大消費電力を測定し比較する．比較の対象とする機器は Jetson Nano[7]である．Jetson Nano には GPU が搭載されており，FPGA を活用する本手法の効果を評価する上で適していると考え採用した．表 4.2.1 に YOLOv7-tiny を 2 種の機器で動作させた場合の処理時間と最大消費電力を示す．

表 4.2.1　処理時間と最大消費電力の比較

機器名	処理時間(ms)	最大消費電力(W)
Ultra96-V2	4804.9	7.0
Jetson Nano	1648.6	8.0

表 4.2.1 より，Ultra96-V2 は Jetson Nano に対し，最大消費電力が 1.0 しか少ないにもかかわらず，処理時間が約 2.9 倍も長いため，現状では本手法が優れた成果を発揮できていないと考える．

5. 結論

本研究では YOLOv7-tiny の特化回路に対し，データ共有時間の削減を行った．その結果，先行研究より約 1.3 倍高速化することができた．しかし，GPU と比べ優れた成果を発揮することができなかった．

今後の展望として特化回路の更なる高速化や Ultra96-V2 よりリソースが少ない代わりにロジックが多い KV260[8]を用いた高速化を行う予定である．

参考文献

[1] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, Institute of Information Science, Academia Sinica, “Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, (2022)
[2] 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).
[3] 舩橋駿介, 中西知嘉子, "共有メモリを活用した「YOLOv7」の高速化手法の検討",電子情報通信学会総合大会(2024).
[4] https://japan.xilinx.com/products/boards-and-kits/1-vad4rl.html
[5] Vivado Design Suite ユーザーガイド 高位合成, https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals_j/xilinx2020_2/ug910-vivado-getting-started.pdf
[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大会(2021).
[7] https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/jetson-nano/product-development/
[8] https://www.amd.com/ja/products/system-on-modules/kria/k26/kv260-vision-starter-kit.html

† 大阪工業大学大学院 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