4K-02

Google Coral Dev Board の分散並列化による AI アクセラレーションの高速化

藤江涼太† 並木美太郎†

東京農工大学工学部情報工学科†

1. はじめに

近年，深層学習の普及に伴いセンサ側で推論を行うエッジ AI が注目され始めている．しかし，エッジ側での AI アクセラレータは消費電力などにより性能が抑えられている．そこで，本研究では AI アクセラレータの分散並列による推論速度の向上を行い，その台数効果についての検証を行う．

2. 先行研究

先行研究[1]では AI アクセラレータが搭載された SoC である Google Coral Dev Board（以下 Coral Board とする）と Nvidia Jetson Nano との比較評価，および GPU との比較を行っている．先行研究によると，Coral Board は Jetson Nano と比べて電力性能比とメモリ効率が良い．そこで，本研究では Coral Board を使用し，分散並列によるさらなる性能向上を目指す．

3. 時間方向への並列による推論速度向上

AI アクセラレータは CNN での推論が主であり，時系列を扱う RNN への対応は限定的である．そのため，入力データの分割は，空間方向への分割よりも時間方向への方が容易であり，センサが取得するデータの各フレームを異なる AI アクセラレータで推論するような並列化をお行える(図 1)．

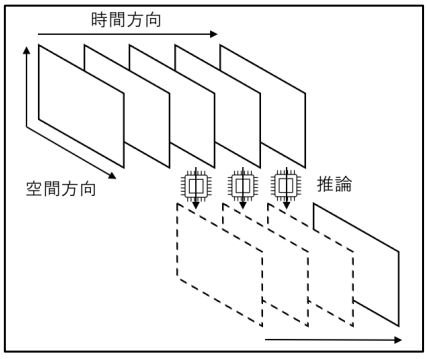

図 1 分割方向と推論の並列化

Improvement AI Acceleration by Distributing and Parallelizing Google Coral Dev Board

†Ryota Fujie †Mitarou Namiki

†Tokyo University of Agriculture and Technology

4. Google Coral Dev Board

表 1 が Coral Board の主なスペックとなる．この他に SRAM キャッシュと MAC(Multiply Accumulation)のパイプライン化による行列計算への調整と， TensorflowLite へのサポートを行う．

表 1 Coral Board のスペック

CPU	NXP i.MX 8M SOC (quad Cortex-A53, Cortex-M4F)
GPU	Integrated GC7000 Lite Graphics
TPU	Google Edge TPU coprocessor
RAM	1 GB LPDDR4
Flash	8GB eMMC
LAN	Gigabit Ethernet
OS	Mendel Linux 4.0

5. 設計

5.1. 構成

センサノードと推論ノードへ処理を分散し，ネットワークで接続する(図 2)．センサノードではセンサデータの配信と推論結果受信を行う．センサノードには Raspberry Pi 4 を，推論ノードには Coral Board を用いる．

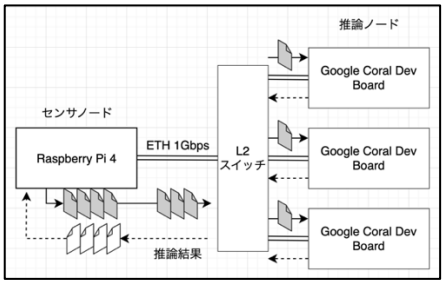

図 2 センサノードと推論ノードの接続とデータの配信

5.2. 処理の流れ

T1 で入力キューから取得したデータを，T2 で推論ノードへ TCP/IP で送信し，T3,T4 で推論を行った後に T5 でセンサノードへ推論結果を送り返し，T6 で出力キューへ戻す流れとなる．(図 3)

推論ノードの管理にはスレッドを用い，入力キュー・出力キューはスレッド間で共通である．この時の排他処理時間は T1,T6 に含まれる．

表２　分散並列数が１台の時の処理時間の内訳

	T1	T2	T3	T4	T5	T6	合計
平均[ms]	0.230	10.712	86.701	23.399	0.069	0.017	121.128
分散[ms]	0.667	1.577	1.463	3.524	0.031	0.004	
割合[%]	0.190%	8.843%	71.578%	19.317%	0.057%	0.014%	

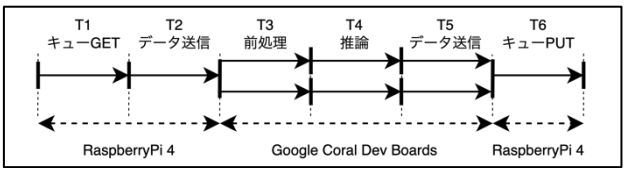

図３　処理の流れ

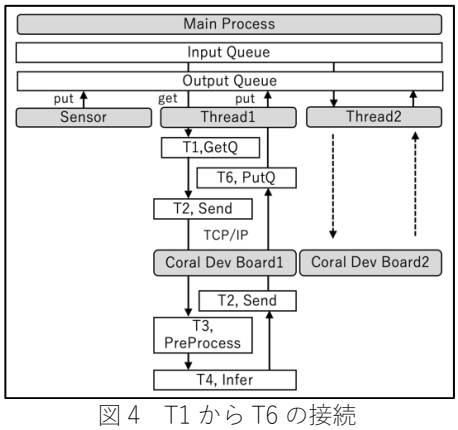

図４　T1 から T6 の接続

6.　実装

センサノードと推論ノードは全て Python3.9.13 で実装する．Coral Board での推論は Pycoral[2]モジュールを使用し，Raspberry Pi 上でのスレッドとキューには，threading モジュールとキ queue モジュールを用いる．また，LAN と L2 スイッチは共に 1Gbps である．

なお，推論には 8bit 量子化された MobileNet-V2[3]を使用し，これは物体判定を行うものである．

7.　分散並列化の評価

表２の１台並列時の内訳より，１度の推論処理時間(T3+T4)は 0.1101 秒であるため，推論処理のみの速度は$\mathbf{9.083}[FPS]$である．また，表３より，オーバーヘッドは，１台並列時で 22.93%，２台並列時で 23.41%となり，二つの線形近似は式１となる．

表３ ２台までの実測値と 300 秒中のオーバーヘッド

並列数	1	2
FPS	7.00	13.91
オーバーヘッド[sec]	68.79	70.22
割合[%]	22.93	23.41

$$Overhead(n) = 0.004471n + 0.2245 \quad \text{式 1}$$

よって，式２より推論処理時間を$InferTime = 0.1101$として，n 台並列時の推論速度を推測することができる．

$$\begin{aligned} fps(n) &= \frac{n}{InferTime}(1 - 0.2245 - 0.004771n) \\ &= \frac{n}{0.1101}(0.7755 - 0.004771n) \\ &= -0.0433317n^2 + 7.04333169n \end{aligned} \quad \text{式 2}$$

8.　台数効果の評価

表４は，並列数が３台と４台の実測時と，式２より求めた予測値を比較した表である．４台並列時の予測値との相対誤差　が 0.29%であり，予測に近い実測値を得られることが分かる．これは，式１の定数部(0.2245)を除き，並列化度が 99.38%であるからだと考えられる．

また，８台並列であれば，式２より 53.57[FPS]となり，7.67 倍となることが期待される．

表４　４台までの実測値と式２による推測値

並列数	1	2	3	4
予測値[FPS]	7.00	13.91	20.74	27.48
実測値[FPS]	7.00	13.91	20.73	27.56
予測誤差	0.0%	0.0%	0.049%	0.29%

9.　おわりに

AI アクセラレータが基本的にサポートする CNN では時系列を扱わない．そこで，時間方向へ推論の分割と処理の分散により，高い台数効果を得られることを確認でき，そのモデル化を行った．

参考文献

[1] Pilsung KANG, Jongmin JO, Benchmarking Modern Edge Devices for AI Applications, IEICE TRANSACTIONS on Information and Systems, Vol.E104-D, No.3, pp.394- 403.

[2] Google(2023), Pycoral, https://github.com/google-coral/pycoral

[3] Google(2023), Mobilenet-V2 Demo, https://github.com/google-coral/test_data/blob/104342d2d3480b3e66203073dac24f4e2dbb4c41/mobilenet_v2_1.0_224_quant_edgetpu.tflite