計算工学講演会論文集　Vol.28 (2023年5月)　　計算工学会

物体検出用深層学習モデルの IoT向けエッジデバイスへの実装と性能評価

Implementation and performance evaluation of deep learning models for object detection in edge devices for IoT

藤本 彩斗[1)]，小林 伸彰[2)]
Ayato Fujimoto and Nobuaki Kobayashi
1) 日本大大学院 精密機械工学専攻 大学院生（〒274-8501 千葉県船橋市習志野台7-24-1, E-mail: csay23033@g.nihon-u.ac.jp）
2) 博(工) 日本大学 理工学部 准教授（〒274-8501 千葉県船橋市習志野台7-24-1, E-mail: kobayashi.nobuaki@nihon-u.ac.jp)

A method of rescuing people by using object detection with a camera mounted on a drone, for example, is being considered. However, single-board computers such as edge devices that can be mounted on small and lightweight drones have various limitations, such as power consumption and area size. In this study, we examined the differences in performance depending on the type of CPU (Central Processing Unit) or GPU (Graphics Processing Unit) used in the SoC (System on a chip). We used three commercially available edge devices: the Raspberry Pi 4 Model B, the Jetson Nano 2GB developer's kit, and the Jetson AGX Xavier developer's kit. Object detection was performed using these three models, and edge devices suitable for installation on a drone were selected from the viewpoints of power consumption and inference speed. YOLOv5, a deep learning-based model, was used for object detection.
Key Words : Edge devices, single board computers, deep learning, YOLOv5

1.　緒言

近年，人命救助や物流にドローンなどの小型軽量なエッジデバイスを用いるための研究が行われている．エッジデバイスとは， 末端の端末で取得したデータをそのままクラウドに送るのではなく，末端の端末上で必要な情報処理を行い，クラウドや通信回線に対する負荷を低減するエッジコンピューティングを用いる機器のことである．例として，スマートフォンや家電製品，自動車，産業用機械などがあげられる．その中でもIoT(Internet of Things)向けエッジデバイスには，用途によって様々な制限が存在する．

今回はドローンに着目する．ドローンを用いる際の制約条件の筆頭として挙げられるのは消費電力や大きさ（サイズ）である．消費電力，大きさはどちらも活動可能時間に影響を与え，必要十分な処理能力を有しながら消費電力，大きさを抑える必要がある．そこで，ドローンに搭載するシングルボードコンピュータとしてRaspberry Pi 4 Model B，Jetson Nano 2GB開発者キット，Jetson AGX Xavier開発者キットの比較を行う．これらは，搭載されているCPU(Central Processing Unit)，GPU(Graphics Processing Unit)の違いから選定した．既存のドローンに搭載することを想定するためOSはUbuntuとし，これに物体検出モデルであるYOLOv5を実装する．YOLOv5を用いた推論にはカメラから取得した映像や，既存の画像，映像を用いることができる．本研究では，カメラの性能や取得した映像のフレームレートなどの影響を取り除くため，COCOデータセットの画像を用いる．ドローンに搭載した際のことを考慮しCUI環境，SSH接続にて推論速度および推論時の消費電力を測定する．これにより，本研究では，物体検出モデルをIoT向けエッジデバイスに実装した際の性能評価を行うことを目的としている．

2.　物体検出用深層学習モデルについて

使用する物体検出用深層学習モデル，YOLOは2015年に公開された物体検出モデルである．[1]

本研究で用いるYOLOv5は，YOLO，YOLOv2，v3，v4の後継に当たり，2020年に公開されたものである．[2~5]

3.　実験

(1) 実験機器

本実験では，次の機器を用いた．実際にドローン等に搭載する際の環境に近づけるため，CUI環境，ssh接続を用いて測定を行う．

・Raspberry Pi 4 Model B
OS　　：Ubuntu 22.04.1 LTS
SoC　　：Broadcom BCM2711
メモリ容量　　：4GB

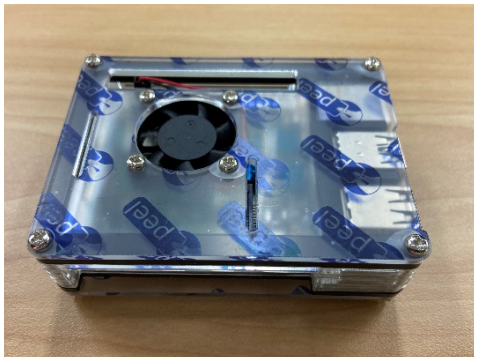

図-1 Raspberry Pi 4 Model B

・NVIDIA Jetson Nano 2GB 開発者キット
OS　　：Jetpack R32 7.3(Ubuntu 20.04.5 LTS)
SoC　　：Tegra X1
メモリ容量　　：2GB

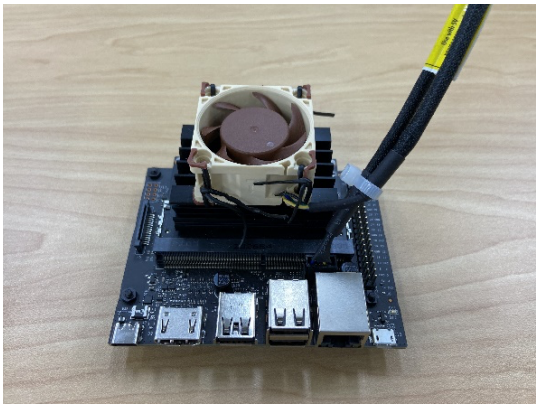

図-2 NVIDIA Jetson Nano 2GB 開発者キット

・NVIDIA Jetson AGX Xavier 開発者キット
OS　　：Jetpack R35 1.0(Ubuntu 20.04.5 LTS)
SoC　　：Xavier
メモリ容量　　：32GB

図-3 NVIDIA Jetson AGX Xavier 開発者キット

・USB電流電圧テスター
ATORCH UD18

図-4 ATORCH UD18

・データセット
Microsoft COCO 2017 [6]
val2017.txtに記載されているリストより上部1000枚を用いた.

・学習モデル
YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x
学習モデルは参考文献[5]よりダウンロードし，使用した.

(2) 測定

USBテスターを各エッジデバイスに接続し，COCOデータセットおよび学習モデルYOLOv5s, YOLOv5m, YOLOv5l, YOLOv5xを用いてdetect.pyを実行した際の消費電力を測定した．また，推論に用いた時間を取得し，推論速度と消費電力から比較を行う.

本研究では時間測定の簡易化のため，detect.pyに対して次のように追記した.

```
30
31  import argparse
32  import os
33  import platform
34  import sys
35  from pathlib import Path
36
37  import torch
38
39  #推論速度の計測用に追加
40  import time
41
42  FILE = Path(__file__).resolve()
43  ROOT = FILE.parents[0]  # YOLOv5 root directory
44  if str(ROOT) not in sys.path:
45      sys.path.append(str(ROOT))  # add ROOT to PATH
46  ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative
47
```

図-5 追記箇所

```
    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(), Profile(), Profile())

    #計測開始
    time_start = time.time()

    for path, im, im0s, vid_cap, s in dataset:

    #計測終了
    time_end  = time.time()
    #計算
    time_result = time_end - time_start
    #推論速度を出力。アンダーバーの後に追記すること。
    print(time_result)
    with open('yolov5_inf_time_.csv','a') as t:
        t.write(f'{str(time_result)}\n')
```

図-6 追記箇所

4. 結果

(1) 推論速度

Jetson Nano 2GBではメモリ不足によりYOLOv5xを実行することができなかったため，測定不能とした．

各エッジデバイスの推論速度を表-1に，1枚当たりの推論速度を図-7に示す．

本結果より，Jetson AGX Xavier が最も推論速度が速く，次いでJetson Nano 2GB，最後にRaspberry Pi 4 Model Bとなることがわかる．

表-1 各エッジデバイスの推論速度(n=1000) [s]

モデル	Raspberry Pi 4 Model B	Jetson Nano 2GB	Jetson AGX Xavier
YOLOv5s	1879.87	189.938	63.9607
YOLOv5m	3648.89	376.579	111.8133
YOLOv5l	6232.01	631.417	175.3225
YOLOv5x	10098.74	-	290.7200

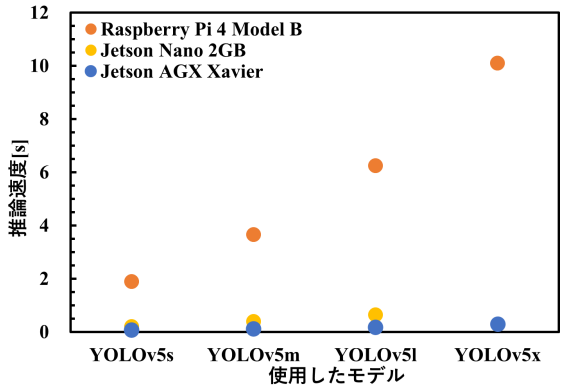

図-7 各エッジデバイスの1枚当たりの推論速度

(2) 推論時の消費電力

推論時の消費電力を測定した結果を表-2，図-8に示す．本結果より，Jetson AGX Xavier が最も推論時の消費電力が高く，Jetson Nano 2GBとRaspberry Pi 4 Model Bは僅差であることがわかる．

表-2 推論時の消費電力

モデル	Raspberry Pi 4 Model B	Jetson Nano 2GB	Jetson AGX Xavier
YOLOv5s	7.90	8.024	17.2394
YOLOv5m	8.10	8.653	20.6703
YOLOv5l	8.39	9.066	20.7778
YOLOv5x	8.54	-	22.3897

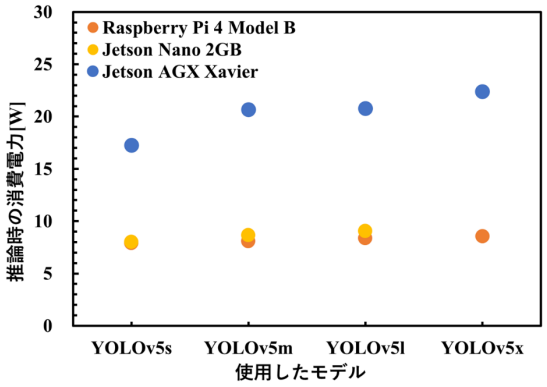

図-8 推論時の消費電力

(3) 推論時の消費電力

以上の結果より各エッジデバイスの推論時の消費電力と推論速度の関係を図にすると図-9のようになる．

Jetson AGX Xavier が最も推論時の消費電力が高く，Jetson Nano 2GBとRaspberry Pi 4 Model Bは僅差であり，Jetson AGX XavierとJetson Nano 2GBの推論速度は僅差となることがわかる．

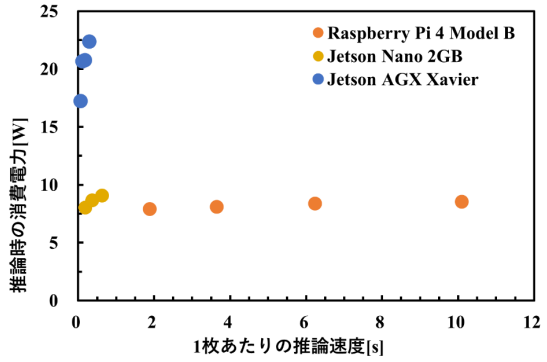

図-9 推論時の消費電力と推論速度の関係

5. 考察

本研究では，YOLOv5を用いてRaspberry Pi 4 Model B，Jetson Nano 2GB，Jetson AGX Xavierの性能評価を行った．

図-7より，推論速度はJetson Nano 2GB，Jetson AGX Xavierが優れている結果となった．また，Jetson Nano 2GBがメモリ不足でYOLOv5xを実装することができないことが分かった．これは，YOLOv5xが最も大きいモデルであるためであるが，量子化を行うことでモデルを縮小化すれば実装することが可能となるのではないかと推測される．

図-8より，推論速度はJetson Nano 2GB で必要十分であると考えられ，消費電力が大きいため，Jetson AGX XavierはYOLOv5を用いた物体検出を行う上では過剰な性能であると考えられる．また，Raspberry Pi 4 Model B とJetson Nano 2GBの消費電力は同等であることがわかる．GPUを搭載していないRaspberry Pi 4 Model Bは推論速度が非常に遅いため，障害物を検出するなどの検出速度が求められるケースにおいては使用を避ける必要があると考えられる．

以上より，本研究で物体検出用深層学習モデルYOLOv5を実装した三機種において適しているエッジデバイスは，Jetson Nano 2GBであると考えられる．

6. 結言

本研究では，物体検出用深層学習モデルであるYOLOv5をRaspberry Pi 4 Model B，Jetson Nano 2GB，Jetson AGX Xavierに実装し，性能評価を行った．その結果，三機種においては Jetson Nano 2GBがYOLOv5を実装するのに最も適しているという結論が得られた．

7. 今後について

本研究はCPU，GPUの違いが物体検出用深層学習モデルの推論にどの程度影響を及ぼすかを改めて確認するものである．今後FPGAにYOLOv5を実装し，比較を行う際の指標として本研究の結果を用いたいと考える．

参考文献

[1] Joseph Redmon，Santosh Divvala，Ross Girshick，Ali Farhadi，“You Only Look Once: Unified, Real-Time Object Detection”，arXiv preprint，arXiv: 1506.02640 (2016)

[2] Joseph Redmon, Ali Farhadi,“YOLO9000: Better, Faster, Stronger”，arXiv preprint，arXiv:1612.08242 (2016)

[3] Redmon, Joseph, Ali Farhadi,“YOLOv3 : An incremental improvement.”, arXiv preprint, arXiv: 1804.02767 (2018)

[4] Alexey Bochkovskiy，Chien-Yao Wang，Hong-Yuan，Mark Liao，“YOLOv4: Optimal Speed and Accuracy of Object Detection”，arXiv preprint，arXiv:2004.10934 (2020)

[5] YOLOv5，https://github.com/ultralytics/yolov5

[6] COCOデータセット，https://cocodataset.org/#home