6H-04

エッジAIにおけるマルチAIアクセラレータを用いた推論エンジンの並列化による性能向上

藤江涼太† 水登恵† 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クラウドを用いないエッジ AI が注目されている．外付けの推論エンジンチップや推論エンジンを搭載した SBC などの AI アクセラレータが，エッジ AI 基盤を支え始めている．クラウドを用いない利点として，リアルタイムに処理を行えると言う点がある．そこで，リアルタイムな処理が求められる自動運転システムを簡易的に作成し，並列化によって速度向上を目指す．なお，本研究は東京農工大学工学部情報工学科 3 年の実験課題として行った．

2. 使用する機材

外付けの AI アクセラレータは複数搭載できるため，並列化を前提に，Intel Compute Stick 2 を推論エンジンに用いる．また，簡易自動運転システムを実行する機体として，Raspberry Pi 4 と Zumo を使用する．これらの構成は次の図 1 が構成となり，NCS2 を Raspberry Pi 上で動作可能にする OpenVINO を使用する．OpenVINO は C++と Python を選択でき，Zumo の制御，推論，画像処理を考慮して Python を使用する．また Zumo の制御は Arduino と公式のライブラリ[1]を使用する．

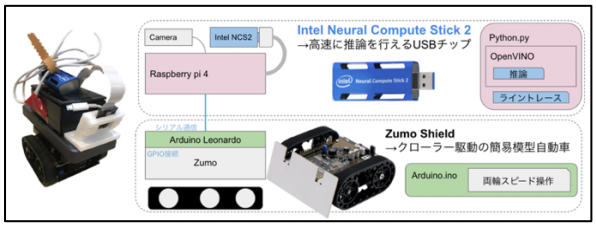

図 1 使用する機器の構成と実際の車体

3. 処理の全体構成

作成する簡易自動運転システムは，End to End で直接コースを学習させる方法と，セグメンテーションによるライントレースの二種類の走行方式を用いる．通常走行はライントレースで行い，右左折などのプログラムが難しい走行は深層学習を用いることで解決する．次の図 2 が自動運転の全体の流れとなる．

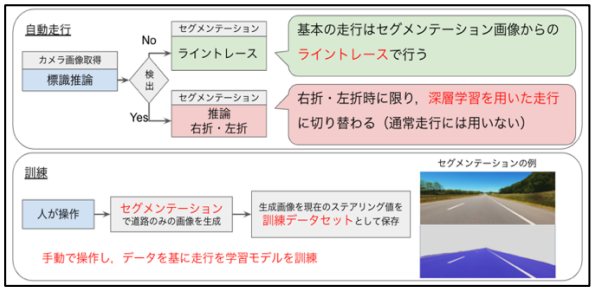

図 2 簡易自動運転システム

4. 三層 CNN でのコースの学習

カメラ画像を元に End to End でコースを学習させ走行させる．学習走行は手動で行い，カメラ画像を入力，ステアリング値を教師データとしてモデルを訓練する．モデルは LeNet を参考にした三層 CNN を，Pytorch を用いて作成する．詳しい構成は図 3 となる．なお，Zumo は二対のクローラー駆動であるから操作に二つの値が必要となるが，モデルを簡潔にするため，左右のステアリング値のみの操作となるよう設計した．

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 3, 56, 76]             228
       BatchNorm2d-2            [-1, 3, 56, 76]               6
         MaxPool2d-3            [-1, 3, 28, 38]               0
            Conv2d-4            [-1, 8, 24, 34]             608
       BatchNorm2d-5            [-1, 8, 24, 34]              16
         MaxPool2d-6            [-1, 8, 12, 17]               0
            Conv2d-7           [-1, 16, 8, 13]           3,216
       BatchNorm2d-8           [-1, 16, 8, 13]              32
         MaxPool2d-9            [-1, 16, 4, 6]               0
           Linear-10                 [-1, 256]          98,560
           Linear-11                  [-1, 64]          16,448
           Linear-12                  [-1, 10]             650
================================================================
Total params: 119,764
Trainable params: 119,764
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 0.36
Params size (MB): 0.46
Estimated Total Size (MB): 0.87
----------------------------------------------------------------
```

図 3 自作モデルの構成

5. セグメンテーションとライントレース

道路検出のためのセマティック・セグメンテーションは，OpenVINO の学習済みモデルとして配布されている road-segmentation-adas-0001[2]を

The Performance improvement by parallelizing inference engine using multi-AI accelerator in edge AI
†Ryota Fujie †Megumi Mizuto †Mitaro Namiki
†Tokyo University of Agriculture and Technology

使用する．このモデルの出力から道路を青でマスクすると，図4のような画像が得られる．ライントレースは実行時間を考慮し，ON-OFF 制御を参考に作成する．この方式で図4の赤線部分をセンサと見立て，道路であれば右へ，道路でなければ左へ走行するプログラムを作成する．

図4 道路部分のマスクと ON-OFF 制御のセンサ箇所

6. NCS2 の並列化による高速化

推論エンジンを並列で使用することで，アプリケーションのリアルタイム性の向上を図った．ただし，ここでの並列化とはキャプチャしたフレームを推論する際のスループットを向上させる並列化となる．これは NCS2 を操作する OpenVINO には同一のフレームを並列で推論する API は存在しないためである．これを 10 フレームの推論に掛かる時間の計測によって性能向上を評価した．

推論エンジンの並列化は図5となる．推論エンジンの本数分，推論スレッドを立て，キュー内のフレームを順次処理していく．単一の場合は推論スレッドが一つとなる[3]．

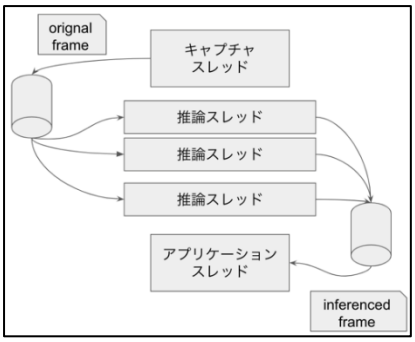

図5 キューを使った並列推論

7. 評価

4章で作成した図3のモデルの学習は図6となった．これは学習データを8:2で分けて，学習とエポック毎の正答率を評価したグラフとなる．このモデルの正解ラベルはステアリング値であるため，左右への多少ズレは走行に問題ないと考えられる．実際にこのモデルを使い，図1の車体でコースを走行させたが，コースアウトを繰り返す結果となった．この原因として，物の配置・部屋の明るさ・撮影間隔などが考えられる．

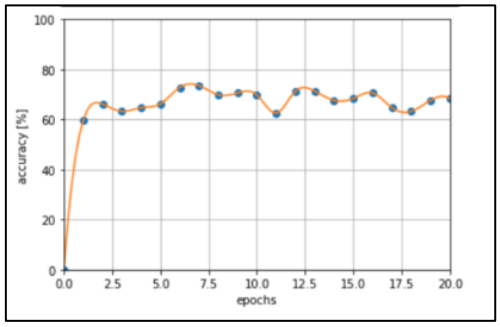

図6 自作モデルの精度

上記のモデルの評価と並行して，並列化による性能向上の評価も行った．図5を単一の推論エンジンで実行した結果，10 フレームの処理に 10 秒かかり平均 1FPS となった．これに対し，NCS2 を二本用いて実行した場合，処理時間は平均 7 秒，おおよそ 1.4FPS と，1.4 倍の結果となった．実行速度が2倍とならない理由として，推論スレッドに使用したキュー内の同期待ちのオーバーヘッドが考えられる．

8. おわりに

AI アクセラレータを用いた簡易的な自動運転システムの試作と，NCS2 の並列化による速度向上を目指し，実際に並列化による高速化を得ることができた．今後は，学習によるコース走行の調整，セグメンテーションを用いたラインとレースを統合した自動運転システムを作成し，その速度向上を図りたい．

参考

[1] pololu/zumo-shield-arduino-library,Github, https://github.com/pololu/zumo-shield-arduino-library（2020.11 参照）

[2] road-segmentation-adas-0001,OpenVINO Toolkit, https://docs.openvinotoolkit.org/2018_R5/_docs_Transportation_segmentation_curbs_release1_caffe_desc_road_segmentation_adas_0001.html（2020.12 参照）

[3] PINTO0309/OpenVINO-YoloV3,Github, https://github.com/PINTO0309/OpenVINO-YoloV3（2020.12 参照）