C-016

Jetson Orin 上での YOLO による物体検出の性能評価

Performance Evaluation of Object Detection with YOLO on Jetson Orin

田窪 慶就 †　　吉田 明正 †
Yoshinari Takubo　　Akimasa Yoshida

1　はじめに

近年，広く用いられている YOLO のような物体検出においては，リアルタイム性が重要視されている．リアルタイムな推論を実現するためには，エッジコンピューティングが不可欠であり，Jetson Orin のようなエッジデバイスの利用が期待されている．Jetson Orin における物体検出の高速化においては，TensorRT ライブラリによるモデル最適化や量子化が期待されている．量子化には，計算に使用するメモリ量を削減し，同時に計算量も減らすことが可能であり，高速化が期待できる．YOLOv4 での TensorRT によるモデル変換の有効性は確認されている [1]．そこで本稿では，NVIDIA Jetson AGX Orin 上で，YOLOv8 を対象として TensorRT[2] によるモデル最適化や量子化を行い，その性能評価を通して有効性を確認する．

2　物体検出

物体検出とは画像や動画の中から物体を検出し, その位置や種類を特定する技術であり, 畳み込みニューラルネットワークを用いることで特徴を抽出し検出している．本稿では物体検出のフレームワークとして YOLOv8[3] を用いる．

2.1　エッジコンピューティングによる物体検出

従来のクラウドコンピューティングではセンサーなどのエッジデバイスで取得したデータをクラウドに送信し, クラウド上で処理を行い, 結果をエッジデバイスに返していた．しかし, 物体検出ではリアルタイム性が重要視されることが多くなり, データ送受信の際のレイテンシが問題視されるようになった．そのためクラウドサーバより処理速度は遅くなるがレイテンシの時間を削減できるエッジコンピューティングが期待されている．本稿では NVIDIA 社の Jetson AGX Orin を用いて性能評価を行う．Orin (64GB メモリ) は最大 275TOPS の処理速度があり, 従来の Jetson AGX Xavier の 8 倍の性能がある．

2.2　YOLOv8 フレームワーク

YOLOv8[3] は YOLOv5 の公開元である Ultralytics 社により公開された YOLO の最新のバージョンであり, PyTorch をベースに作られている．従来の YOLO に比べ高速かつ高精度に設計されており, 高いパフォーマンスを実現している．さらに物体検出だけでなく画像セグメンテーション, 画像分類, 姿勢推定にも対応している．本稿では物体検出が目的であるため画像セグメンテーションなどについては扱わない．さらに YOLOv8 では大きさの違う 5 つのモデル (YOLOv8n,YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x) が公開されており,YOLOv8n が最小で YOLOv8x が最大もモデルとなっている．本稿では最も高速に物体検出ができる YOLOv8n モデルを使用し検証する．

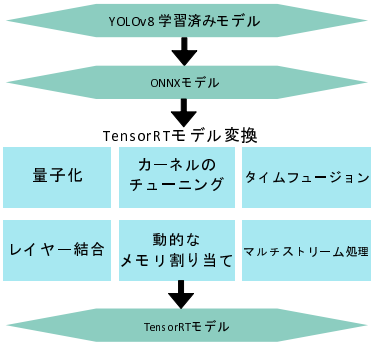

図 1　TensorRT を用いたモデル変換.

3　モデル最適化による高速化

ニューラルネットワークの学習に用いるディープラーニングフレームワークは柔軟性や精度を高めるため複雑なネットワークになっている．そのため推論に使用するとオーバーヘッドが大きくなり処理時間が多くなってしまうことがある．そこで学習済みのモデルを推論専用ライブラリ向けにモデルを変換することでオーバーヘッドを回避し推論の高速化を実現できる．学習済みモデルを変換する手法として TensorRT がある．本稿ではこの TensorRT を使用しモデルの変換を行う．

3.1　ONNX

ONNX[5] は Open Neural Network Exchange の略であり, 機械学習モデルの相互運用を可能とするために構築されたオープンソースのフォーマットである．機械学習には PyTorch[4] や TensorFlow, Caffe といった複数のフレームワークがある．従来は同一の機械学習フレームワークを用いプロジェクトを実装しなければならなかったが ONNX を介することで異なるフレームワーク間のモデルの共有が可能になった．本稿で取り扱う TensorRT においても, YOLOv8 の PyTorch モデルから一度 ONNX に変換し, その後 TensorRT モデルへ変換する．

3.2　TensorRT

TensorRT[2] は NVIDIA 社が開発したディープラーニング推論を高速に実行するための SDK であり, 推論エンジンの作成と推論の実行を可能としている [6]．図 1 では推論エンジンを作成する TensorRT のワークフローを

†明治大学大学院先端数理科学研究科ネットワークデザイン専攻
Department of Network Design, Graduate School of Advanced Mathematical Sciences, Meiji University

示している．TensorRT では各レイヤーに最適なアルゴリズムとデータ形式を選択するカーネルの自動チューニングによってレイヤー単位の最適化を行うことができる．さらにレイヤー結合といった推論実行時のノード結合により, カーネルの起動回数を少なくし, GPU メモリの最適化をすることができる．加えて, 動的なメモリの割り当てといった演算中のメモリの使用量を最小化し, メモリ割り当てのオーバーヘッドを回避する機能もある．また, モデルを FP16 に量子化することでモデルサイズやメモリの使用量を削減することができる．

3.3 モデル量子化

量子化とはモデルの精度を保ったままメモリの使用量を削減し処理時間を最適化し高速化する技術である．量子化では FP32 を FP16 に変換することで必要なメモリの量が少なくなる．FP32 では最初の 1 ビットを数値の符号, 次の 8 ビットを指数部, 最後の 23 ビットを仮数部といった数値を構成する桁を示すのに使われており, 合計 32 ビットのメモリを使用し計算を行っている．一方, FP16 は 1 ビットを符号, 5 ビットを指数部, 10 ビットを仮数部として使用しており, 合計 16 ビットのメモリを使用し数値を表している．これによりメモリの使用領域が半分に済むため, キャッシュ効果を高めることができる．代わりに表せる数値の範囲が減ってしまうため精度が落ちてしまうことがある．TensorRT には機能の 1 つとして量子化が可能であり, 本稿では量子化を適用する．YOLOv8 の TensorRT モデルでは FP32 のデータ型となっている．そこで本稿では FP32 から FP16 に量子化し, 性能評価を行う．

4 モデル最適化による物体検出の性能評価

本稿では, YOLOv8 フレームワークに対し, TensorRT によるモデル変換を行い, 変換後のモデルの評価を行う．さらに量子化を行い FP16 でのモデルの評価も行う．

4.1 性能評価環境

本性能評価では表 1 のマシンを利用し, 性能評価を行った．また, モデル変換には TensorRT8.5.2 を使用した．

表 1　性能評価に用いる NVIDIA Jetson AGX Orin.

CPU	12 core Arm Cortex A78AE v8.2 64-bit
メモリ	64GB
GPU	2048 Core NVIDIA Ampere architecture GPU with 64 Tensor Cores
OS	Ubuntu 20.04.6 LTS
処理系	CUDA11.4

表 2　YOLOv8 の TensorRT を用いた推論の性能評価.

	前処理 [ms]	推論 [ms]	後処理 [ms]	合計 [ms]
TensorRT なし FP32	7.86	39.38	4.58	51.82
TensorRT あり FP32	8.84	15.36	6.60	30.80
TensorRT あり FP16	9.02	9.82	7.69	26.52

4.2 YOLOv8 の TensorRT を用いた推論の性能評価

本性能評価では, YOLOv8 において, 学習済みモデルを ONNX と TensorRT 変換を用いて変換を行い, モデルの評価を行った．学習済みモデルは COCO データセットを用いて学習し, YOLOv8 の中でも最小のモデルである YOLOv8n を使用した．推論画像 1 枚で前処理, 推論, 後処理の時間を測定した結果が表 2 となる．推論全体を見ると YOLOv8n の PytTorch モデルの 51.82[ms] に対し, TensorRT モデル変換だと 30.8[ms] と 1.68 倍の速度向上が得られた．また TensorRT の FP16 モデルで同様に評価を行ったところ 1.95 倍の速度向上が得られた．

表 3　推論枚数毎の YOLOv8 の TensorRT を用いた推論の性能評価.

推論枚数	1 枚 [ms]	10 枚 [ms]	100 枚 [ms]	200 枚 [ms]
TensorRT なし FP32	39.38	30.64	26.28	25.58
TensorRT あり FP32	15.36	16.50	13.96	14.20
TensorRT あり FP16	9.82	11.46	10.22	9.48

4.3 推論枚数毎の TensorRT を用いた YOLOv8 の推論の性能評価

推論画像枚数を変化させ, 1 枚あたりの平均推論時間を表 3 に示す．画像 200 枚の推論で YOLOv8n の PyTorch モデルは 25.58[ms] に対し, TensorRT モデル変換では 14.20[ms] となり 1.80 倍の速度向上が得られた．また, TensorRT の FP16 モデルでは 9.48[ms] となり 2.70 倍の速度向上が得られた．

5 おわりに

本稿では, 物体検出で用いられる YOLOv8 フレームワークの最小モデルである YOLOv8n に対して, ONNX と TensorRT を用いてモデル変換を行い, さらに TensorRT によるモデル変換と量子化を行い, 推論を行った．

性能評価では YOLOv8 フレームワークに対して TensorRT モデル変換を行うことで 1.68 倍の速度向上を得ることができた．さらに TensorRT モデルに対し量子化を行うことで 1.95 倍の速度向上が得ることができた．また推論画像を増やし測定したところ画像 200 枚では TensorRT モデル変換では 1.80 倍, TensorRT モデルに対して量子化を行うことで 2.70 倍の速度向上を得ることができた．

以上の結果から, YOLOv8 の TensorRT モデル変換の有効性と量子化によるモデルの軽量化による有効性が確認された．

参考文献

[1] 大内佑一朗, 吉田明正. エッジコンピューティングによる畳み込みニューラルネットワークを用いた物体検出，情報処理学会第 84 回全国大会，7J-02, 2022.

[2] NVIDIA. https://developer.nvidia.com/tensorrt, 2021.

[3] Ultralytics. YOLOv8, https://github.com/ultralytics/ultralytics, 2023.

[4] A. Paszke, S. Gross, F. Massa, PyTorch: An Imperative Style, High-Performance Deep Learning Library, Advances in Neural Information Processing Systems 32, 2019.

[5] ONNX. https://onnx.ai/, 2019.

[6] EunJin Jeong, Jangryul Kim, Samnieng Tan, Jaeseong Lee, Soonhoi Ha. Deep Learning Inference Parallelization on Heterogeneous Processors With TensorRT, IEEE Embedded Systems Letters, 2022.