Vol.2024-ARC-256 No.23
Vol.2024-SLDM-205 No.23
Vol.2024-EMB-65 No.23
2024/3/22

Edge Device上での AIロボット用深層予測学習モデル推論処理の高速化

朱 允楷[1,a)]　梅田 弾[1,2,b)]　伊藤 洋[1,3,c)]　尾形 哲也[1,d)]　木村 啓二[1,e)]

概要：深層学習によりロボットの将来の動作やカメラ画像を予測する深層予測学習が提案されている．深層予測学習により，ロボットはカメラや各種センサーの入力を用いて自身の周囲の環境を認識し，状況に応じた適切な動作のためのアクチュエータ制御が可能となる．限られたサイズのロボットの筐体でこの深層予測学習を利用するためには，高性能かつ高電力効率なコンピュータのハードウェアとソフトウェアが必要となる．深層学習による画像処理では，必要なハードウェア資源を削減する手法として，少ないビット数で演算を行う量子化が広く用いられている．一般的な計算機による 32bit 単精度浮動小数点による深層学習処理を，16bit 半精度浮動小数点型（FP16）や 8bit 整数型（INT8）のような少ないビット数で行うことで，計算の高速化，メモリ使用量削減，ハードウェアの簡略化，及び低消費電力化といった効果が期待できる．本稿では，ロボットのカメラ画像と関節角度を入力として，畳み込みニューラルネットワーク（CNN）と回帰型ニューラルネットワーク（RNN）から構成されるロボットの予測制御モデルを Edge Devices 上で評価する．具体的には，この深層学習モデルに対し学習後の事後的な量子化手法である Post Training Quantization (PTQ) で量子化を実施し，NVIDIA Jetson Orin Nano 上で精度・性能評価を実施する．評価の結果，FP16 で推論をすることにより，FP32 に対して同程度の精度で 1.28 – 1.79 倍の速度向上が得られた．しかし，INT8 では FP16 に対して速度低下が発生するモデルがあり，速度向上率は 0.86 – 1.04 倍であった．さらに，FP16 で推論することにより，FP32 に対してフレームあたりの消費エネルギーを 12 – 39% 削減したが，INT8 では FP16 に対して，エネルギー消費が 8% 増加するモデルがあった．量子化により，推論過程においてデータ変換等オーバーヘッドにより消費エネルギーが上昇する局面があることから，電力効率向上のためにはデータ変換も含めたシステム全体の設計が重要である．

Accelerating Inference Processing of Deep Predictive Learning for AI Robots on Edge Devices

Zhu Yunkai[1,a)]　Umeda Dan[1,2,b)]　Ito Hiroshi[1,3,c)]　Ogata Tetsuya[1,d)]　Kimura Keiji[1,e)]

1. はじめに

少子高齢化とそれに伴う医療や介護の現場での労働力不足が世界的な問題となっている．これに対して，AI を搭載した自律動作可能なロボットの人間の生活空間への導入による労働力補助が，問題解決の有力な手段として期待されている [1].

医療や介護の現場で機能するロボットには高度な環境認識や柔軟な動作タスク決定を可能とする AI 処理が要求されるが，これを実現する手段として深層予測学習が提案されている [2]．これは，深層学習によりロボットの将来の動作やカメラ画像を予測し，その予測結果に基づきアクチュエータを適切に制御する技術である．この深層予測学習モデル複数を同時並行で処理することにより，各モデルに対

[1] 早稲田大学
Waseda University
[2] 株式会社ティアフォー
TIER IV, Inc.
[3] 株式会社日立製作所
Hitachi, Ltd.
[a)] zhuyunkai@kasahara.cs.waseda.ac.jp
[b)] dan.umeda@tier4.jp
[c)] hiroshi.ito.ws@hitachi.com
[d)] ogata@waseda.jp
[e)] keiji@waseda.jp

ⓒ2024 Information Processing Society of Japan

応する動作タスクからその場の状況に応じた最適なものを選択し，ロボットを制御できる [3].

このように複数モデルの導入により柔軟なロボットの自律動作が可能になるが，その一方で要求される計算機のハードウェア資源はモデルの数に応じて増加し，結果として消費電力の増大を招く．必要な計算能力の確保のために安易にハードウェア資源を増強すると，計算機モジュールが増加し，また必要となる消費電力の増大によりバッテリユニット，電源ユニット，及び冷却ユニットが肥大化し，結果としてロボットの筐体自体が肥大化して要求される柔軟な動作の妨げとなる．その一方で必要な計算能力を確保しなければロボットの動作が緩慢になり，ユーザーエクスペリエンスが悪くなってしまう．また，ロボットの周囲の環境は刻々と変化し，遅延が大きくなるとその時点での環境に応じた制御を提供できなくなる．すなわち，ロボットの筐体にコンパクトに搭載可能な低消費電力 Edge Device 上で，高性能な深層予測学習モデルを実現可能なハードウェア及びソフトウェア技術が重要な要素となる．

深層学習モデルの学習では 32bit 単精度浮動小数点による演算が広く用いられている．一方で，Edge Device での推論計算においては，量子化を行うことにより，SIMD 演算幅の拡大による推論速度が向上する．さらに，メモリ帯域の削減や演算器の簡素化によりハードウェアの消費電力や発熱の面でも有利である．これらの観点から，Edge Device における画像を対象とした深層学習の推論処理では，8bit 量子化が広く用いられている．しかしながら，量子化によりモデルの推論精度が低下する．そのため，一般に深層学習モデルを用いるアプリケーションの運用においては，量子化による性能向上，及び消費電力削減と推論精度低下のバランスが見越した量子化適用が重要である．

文献 [2] では，深層予測学習モデルを使ったロボット向けアプリケーションとして，深層学習フレームワーク PyTorch [4] を用いた Embodied Intelligence with Deep Predictive Learning (EIPL) を実装している．本稿では，EIPL の PyTorch 版実装で学習された，学習済みモデルに対して NVIDIA GPU 上で推論最適化を行うため，TensorRT [5] を使って EIPL の推論部分を再実装した．さらに，NVIDIA 製 Edge Computing 向け SoC モジュールである Jetson Orin Nano 4GB の内蔵 GPU 上で推論処理を実行し，TensorRT による量子化の効果を時間，精度，電力の面から評価を行った．

以下本稿は，2 節で先行研究で提案されたロボット向け深層予測学習モデルについて，3 節で量子化に関する先行研究とツールについて説明し，4 節で深層予測学習モデルを量子化するための実装について述べ，5 節で Jetson での量子化の効果を報告し，6 節でまとめる．

2. ロボット向け深層予測学習モデル

深層予測学習は，少ない学習データでロボットが環境認識と動作生成を可能とする [2]．この深層予測学習によるロボット動作生成ライブラリとして EIPL (Embodied Intelligence with Deep Predictive Learning) が開発された．

2.1 深層予測学習モデルの設計

EIPL が対象とするロボットは，センサとしてカメラ及びロボットアームの各関節に付随する角度センサを持つものとする．カメラにより RGB 各チャネルが UNSIGNED INT8 のカメラ画像，角度センサにより FP32 の関節角度をそれぞれ取得する．これら入力は FP32 に変換され，深層予測学習モデルにより未来のカメラ画像と関節角度を出力する．

時系列データを扱うために，EIPL では回帰型ニューラルネットワーク (RNN) を用いる．RNN は，内部状態を保持することで，連続した状態の変化を扱うのに適している．すなわち，RNN は時刻 t において，内部状態として時刻 $t-1$ までの情報を保持し，ここから時刻 $t+1$ の状態を予測する．

本稿で扱う EIPL のモデルは，RNN の一種である Long Short-Term Memory (LSTM) [6] を使用する．LSTM は，入力ゲート，忘却ゲート，及び出力ゲートの 3 つのゲートを持ち，それぞれが学習パラメータを持つ．また，内部状態はとして，セル状態と隠れ状態の 2 つを持つ．隠れ状態 h_{t-1} は短期記憶として，セル状態 c_{t-1} は長期記憶として，それぞれ機能する．実際の推論処理では，時刻 t に $[h_{t-1}, c_{t-1}, x_t]$ を入力し，$[h_t, c_t]$ を出力する．

EIPL が扱うデータは，カメラ画像と関節角度である．カメラ画像は チャンネル数 × 幅 (px) × 高さ (px) の 3 次元テンソルである．関節角度データは関節の数だけある 1 次元のテンソルである．そこで，EIPL ではカメラ画像のデータから畳み込みニューラルネットワーク (CNN) を用いて画像の特徴量を抽出し，関節角度データと結合して LSTM に入力する．

2.2 深層予測学習モデルの実装

EIPL には，いくつかのモデルが実装されている．本稿ではその中から， Spatial Attention with Recurrent Neural Network (SARNN)，CNNRNN，及び CNNRNN with Layer Normalization (CNNRNNLN) の 3 つのモデルを扱う．以下で，これらのモデルの具体的な実装についてそれぞれ説明する．

2.2.1 SARNN

SARNN は，カメラ画像からタスクにとって重要な点を明示的に抽出することにより，ロバスト性を向上させたモ

Vol.2024-ARC-256 No.23
Vol.2024-SLDM-205 No.23
Vol.2024-EMB-65 No.23
2024/3/22

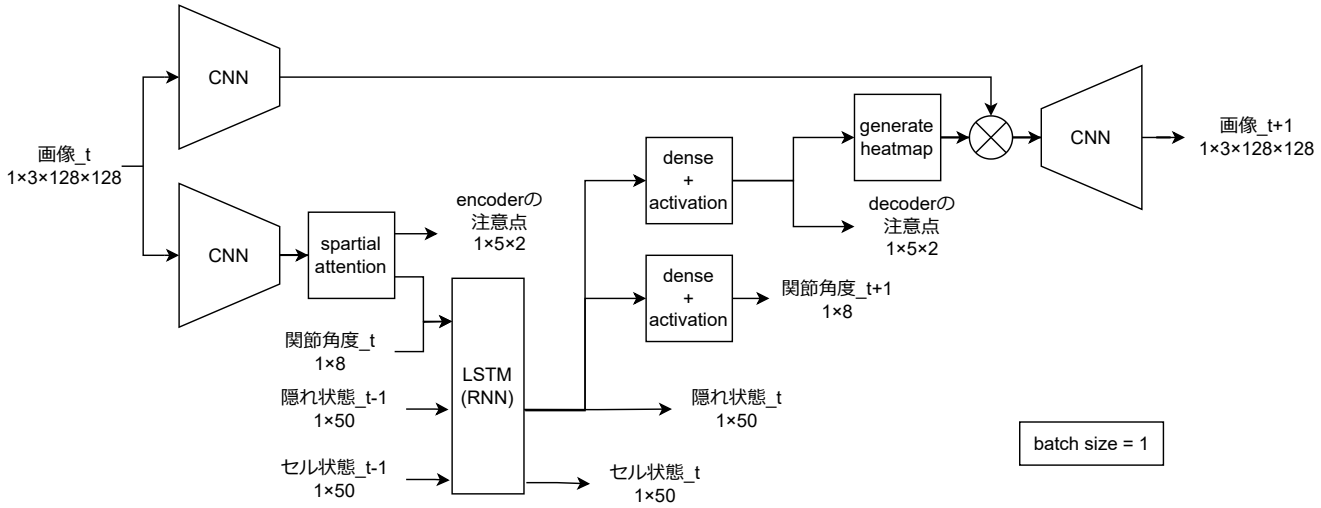

図 1 SARNN モデルの構造

表 1 SARNN モデルの詳細

モデル名	SARNN
パラメータ数	34,515
活性化関数	LeakyReLU
入力	画像, 関節角度, 隠れ状態, セル状態
出力	画像, 関節角度, 隠れ状態, セル状態, エンコーダー注意点, デコーダー注意点

表 2 CNNRNN，CNNRNNLN モデルの詳細

モデル名	CNNRNN	CNNRNNLN
パラメータ数	82,875	1,282,611
活性化関数	Tanh	ReLU
入力	画像, 関節角度, 隠れ状態, セル状態	画像, 関節角度, 隠れ状態, セル状態
出力	画像, 関節角度, 隠れ状態, セル状態	画像, 関節角度, 隠れ状態, セル状態

デルである．図 1 に SARNN の構造，表 1 に SARNN の詳細をそれぞれ示す．図中，カメラ画像（画像_t）から CNN により得られた特徴マップに対して，Spatial Attention を用いて重要な点を抽出する．Spatial Attention とは，温度付き Softmax 関数を用いて，CNN から得られた特徴マップに対して特徴量を強調し，それ以外の点を抑制する機構である．Spatial Attention の出力として，2 次元の座標データが得られる．この座標データは，関節角度（関節角度_t）と結合されて LSTM の入力となる．LSTM の出力に対して全結合層を用いて，時刻 $t+1$ の予測関節角度（関節角度_t+1）が得られる．次に，LSTM の出力を用いて生成したヒートマップと入力画像の積を取り，これを転置畳み込みを用いて，カメラ画像と同じ次元に変換する．これが，時刻 $t+1$ のカメラ画像の予測値（画像_t+1）となる．

2.2.2 CNNRNN

図 2 に CNNRNN の構造，表 2 に CNNRNN 及び CNNRNNLN の詳細をそれぞれ示す．CNNRNN は，CNN と LSTM を組み合わせて画像と関節角度を同時に扱うモデルである．これは，SARNN モデルからカメラ画像上のタスクにとって重要な点を明示的に抽出する機構を取り除いたモデルである．さらに CNNRNNLN は，CNNRNN に対して畳み込み層，転置畳み込み層，全結合層の後に Layer Normalization 層 [7] を追加したモデルである．

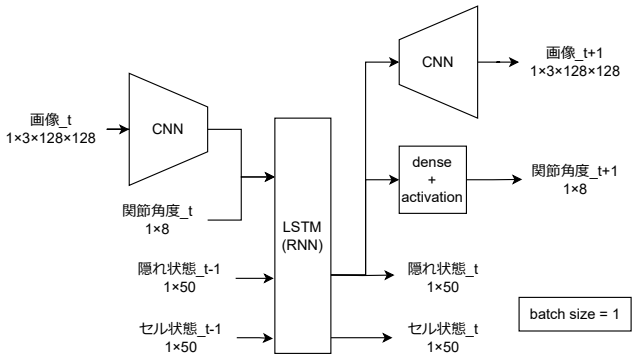

図 2 CNNRNN モデルの構造

2.3 現在の EIPL 実装の問題点

文献 [2] では，モデルの設計や学習部分について重視されている一方で，推論についてはその詳細な記述が不足している．しかしながら，実際にロボットを動かすときには，推論処理が重要になる．現在の EIPL で実装されている深層予測学習モデルの推論処理は計算遅延が大きく，ロボットの動作速度に制限がある．EIPL に付属する学習済みデータは，机の上に置かれた物体をロボットアームで掴んで動かすという一連の動作を実装している．この動作では，ロボットのインタラクト対象である物体は静止しているため，1 フレームあたりの処理時間は長くてもよい．しかしながら，動いている物体がロボットのインタラクト対象となる場合は，1 フレームあたりの処理時間を短くしイ

ⓒ2024 Information Processing Society of Japan

ンタラクト対象に遅滞なく追従する必要がある．

3. 量子化とそのツール

1 節で述べたように, 量子化は畳み込みニューラルネットワーク（CNN）等の画像処理を扱う深層学習モデルの推論処理で広く用いられる手法である．量子化は EIPL で実装されている深層予測学習についても適用可能であり，CNN と同様に推論処理の高速化を期待できる．本節では，本稿の評価で EIPL に適用した量子化及び量子化適用時に使用したツールである TensorRT について紹介する．

3.1 量子化

コンピュータ上の数値計算は一般的に 8, 16, 32, 64 ビットの浮動小数点演算や固定小数点演算が用いられる．浮動小数点演算は数値の表現範囲が広いが，演算では指数部と仮数部両方の計算が必要で，ハードウェアの演算回路が複雑になる．また，少ないビット数の計算は要求メモリバンド幅の削減により高いメモリアクセススループットを実現でき，さらに SIMD 演算幅の向上による演算スループットの向上が得られる．そのため，少ないビット数での固定小数点演算はハードウェアの観点から処理速度，消費電力や実装面積の点で優れている．その一方で，ビット幅削減による演算精度の低下はモデルの精度の低下招く．そこでモデルの精度を一定に保ちながら演算精度を落とす手法が提案されている [8], [9], [10].

文献 [8], [9] では，精度劣化を抑えて，機械学習モデルを量子化する手法を示している．また文献 [10] では，回帰型ニューラルネットワーク (RNN) ベースの音声強調モデルに対して量子化を行い，Multi-Core MicroController Unit で実行することで電力効率が向上することを示している．

3.2 TensorRT

本稿では，量子化を行うツールとして，NVIDIA が提供している深層学習コンパイラ，推論最適化ライブラリである TensorRT [5] を使用した．TensorRT は以下に示すような最適化を適用できる [11].

- Reduced Precision: モデルに対して量子化を行い，FP16 や INT8 のような低い演算精度で推論を行う．
- Layer and Tensor Fusion: モデルのレイヤーを統合し，レイヤー間のメモリ転送を減らし，GPU メモリの使用を最適化する．
- Kernel Auto-Tuning: ターゲットデバイスの GPU アーキテクチャやカーネルサイズに基づいて，最適なデータレイアウトやアルゴリズムを選択する．
- Dynamic Tensor Memory: 中間層のメモリを動的に開放することで，メモリを効率的に再利用する．
- Multi-Stream Execution: 複数の入力データストリームを同時に処理する．

4. TensorRT 版 EIPL の実装

文献 [2] では提案されたロボット向け深層予測学習モデルを PyTorch で実装している．本稿では，その推論部分を TensorRT で推論できるように移植し，TensorRT が持つ量子化等の最適化機能を適用可能とした．

まず，EIPL の学習済みモデルを共通フォーマットである ONNX [12] に変換する．次に，TensorRT の API を用いて，GPU 上での推論処理に最適化したエンジンをコンパイルする．この段階で，量子化なし，FP16，INT8 量子化を行ったモデルを作成できる．最後に TensorRT のランタイムを用いて，コンパイルしたエンジンを実行する．

TensorRT で量子化を適用するとき，すべての層が強制的に指定した演算精度になるわけではなく，各層の最終的な演算精度は，TensorRT の Kernel Auto-Tuning によって決定される．層によっては，INT8 に対応していないもの，CUDA コアの FP32 で計算すると速いものや，INT8 で計算すると INT8–FP32 間のデータの変換が必要になり量子化を適用しないほうが速い場合が存在する．ユーザが指定した演算精度にはならなずにフォールバックされ, INT8 や FP16 を指定している場合でも FP32 で計算される場合がある．5 節で示した評価結果の FP16, INT8 は，TensorRT に対して指定した演算精度である．

また，INT8 量子化を適用するときにはキャリブレーションという操作を行う．キャリブレーションでは，キャリブレーションデータに対して推論を行い，中間層での出力結果に基づいて各層の FP32 と INT8 を変換するときのスケールを決定する．これは，量子化後のモデルの精度を左右する大きな要素である．TensorRT では，キャリブレーションアルゴリズムとして，IInt8LegacyCalibrator, IInt8EntropyCalibrator, IInt8EntropyCalibrator2, IInt8MinMaxCalibrator が用意されていて，今回は IInt8EntropyCalibrator2 を使用した．キャリブレーション用の入力データセットは，EIPL 付属のテストデータセットの 0 番目のデータとそれを FP32 で推論して得られた LSTM の状態変化もあわせてキャリブレーションデータとした．

各フレームにおける推論処理は以下のように実装した．各モデルの入出力データは表 1, 2 にそれぞれ示した通りである．

(1) 入力データの前処理を行う．画像は UNSIGNED INT8 で表現されているデータであり，関節角度は FP32 で表現されているデータである．これらを $0 - 1.0$ に正規化し FP32 に変換する．
(2) 入力データを CPU メモリ (Host memory) から GPU メモリ (Device memory) へ非同期転送する．

ⓒ2024 Information Processing Society of Japan

表 3 Jetson Orin Nano 4GB の諸元

モデル名	Jetson Orin Nano 4GB
パフォーマンス	20TOPS
GPU 世代	Ampere
Tensor コア数	16
CUDA コア数	512
GPU 周波数	306MHz - 625MHz
メモリ	4GB 64-bit LPDDR4x 34GB/s
消費電力	7-15W

表 4 Jetson Orin Nano 4GB 上での推論時間 [ms]

	PyTorch	TensorRT		
モデル名	FP32	FP32	FP16	INT8
SARNN	5.13	4.42	2.47	2.88
CNNRNN	3.11	1.87	1.32	1.27
CNNRNNLN	4.13	3.55	2.77	2.88

表 5 Jetson Orin Nano 4GB 上での EIPL TensorRT 版実装の 前処理・後処理を含めた推論時間 [ms]

モデル名	FP32	FP16	INT8
SARNN	6.17	4.00	4.43
CNNRNN	3.39	2.83	2.75
CNNRNNLN	5.08	4.30	4.35

(3) GPU 上で推論計算を行う．
(4) 出力データを Device memory から Host memory へ非同期転送する．
(5) 出力データの後処理を行う．画像，関節角度はそれぞれ正規化されているデータを元のスケールに戻す．また，結果表示用に画像，関節角度，エンコーダー注意点 (SARNN のみ)，デコーダー注意点 (SARNN のみ) のコピーをメモリ内に保存する．さらに，LSTM の状態は次のフレームで使用するので，このコピーも行う．

5. NVIDIA Jetson Orin Nano 上での性能評価

5.1 評価環境

本節では，第 4 節で述べた実装に対して NVIDIA Jetson Orin Nano 4GB で評価を行った結果について報告する．評価で使用した Jetson Orin Nano の諸元を表 3 に示す．推論では，内蔵 GPU の CUDA コアや Tensor コアを使用した．

5.2 推論時間評価

EIPL のオリジナル実装 (PyTorch) と TensorRT を用いた実装のそれぞれのモデルに対して，EIPL に付属するテストデータセット 0 番目のデータを入力として与えて推論処理を行った．計測前に，Jetson の CPU と GPU のクロック周波数を最大に設定した．GPU へモデルをロードし，数フレーム分のウォームアップを行ったのち一連の推論処理を行い，1 フレームの推論にかかる時間を計測した．4 節に示した推論処理の流れの (2) – (4) を計測したものを表 4 に，(1) – (5) を計測したもの表 5 にそれぞれ示す．

表 4 より Jetson Orin Nano 4GB 上での各モデルの推論時間をそれぞれ比較すると，PyTorch 版の実装に対して，FP32 で 1.2 倍，1.7 倍，1.2 倍，FP16 で 2.1 倍，2.4 倍，1.5 倍，INT8 で 1.8 倍，2.4 倍，1.4 倍の高速化がそれぞれ得られた．同じ FP32 により演算処理をしている PyTorch 版と TensorRT 版を比較すると，TensorRT 版は PyTorch 版に対して，SARNN で 1.16 倍，CNNRNN で 1.66 倍，CNNRNNLN で 1.16 倍の速度向上をそれぞれ得ていることから，TnsorRT による量子化以外の最適化の効果が確認できる．さらに TensorRT 版で演算精度を FP16 にすることにより，TensorRT の FP32 に対して SARNN で 1.79 倍，CRNRNN で 1.42 倍，CNNRNNLN で 1.28 倍の速度向上をそれぞれ得ており，深層予測学習モデルの推論時間における半精度量子化の効果を確認できる．一方，INT8 では CNNRNN のみ FP16 に対して速度向上があり 1.04 倍の速度向上を得たが，SARNN では 0.86 倍，CNNRNNLN では 0.96 倍といずれも速度低下となった．

SARNN に着目すると，このモデルでは，表 1 で示したように活性化関数で LeakyReLU を使用していて，負の値に対して一定の値をかける計算を行う．TensorRT では，この計算に generatedNativePointwise というカーネルを使用していて，これは FP32 使用時に推論全体の約 20%，FP16 と INT8 使用時ではそれぞれ約 14%を占める．活性化関数を ReLU に変更することで，性能向上が期待できる．

CNNRNNLN では，各 Convolution 層の後に Layer Normalization 層がある．TensorRT の Normalization では，オーバーフローを回避するために，INT8 を指定した場合でも浮動小数点で計算される．すなわち，Convolution 層を INT8 で計算した後に，浮動小数点への変換処理を行い，その後に Convolution 層がある場合には浮動小数点から INT8 への変換処理を行う．これら変換処理により実行効率が低下する．

5.3 精度評価

機械学習モデルに対して量子化を適用した場合，表現能力の低下により精度劣化が生じることがある．本節では，量子化を適用したモデルの推論精度劣化を定量的に評価した．SARNN，CNNRNN，CNNRNNLN の出力は複数存在するが，実際にロボットの制御に使われるのは関節角度のデータである．すなわち，推論結果の関節角度が正解デー

ⓒ2024 Information Processing Society of Japan

Vol.2024-ARC-256 No.23
Vol.2024-SLDM-205 No.23
Vol.2024-EMB-65 No.23
2024/3/22

表 6　量子化による精度劣化

	FP32	FP16		INT8	
モデル名	MSE	MSE	Error Rate	MSE	Error Rate
SARNN	3.36.E-05	3.36.E-05	0.999	3.42.E-05	1.018
CNNRNN	1.33.E-04	1.33.E-04	1.001	1.33.E-04	0.998
CNNRNNLN	2.31.E-04	2.31.E-04	1.000	2.37.E-03	10.27

タに近ければ，ロボットの動作は正解に近いと言える．また，関節角度のデータは時系列データであることを利用して，$t+1$ の入力関節角度は，t の出力関節角度に対する正解データと見なせる．以上より，$t+1$ の入力関節角度と t の出力関節角度の差を正解データとの近さと定義する．より具体的には，$t+1$ の入力関節角度と t の出力関節角度の Mean Squared Error (MSE) により本評価における推論精度を定義する．以下，TnesorRT 版の FP32 モデルでの MSE に対して，FP16 モデルと INT8 モデルの MSE を比較し，量子化による精度劣化を評価した結果を述べる．

表 6 に精度評価の結果を示す．CNNRNNLN の INT8 を除くと，FP16 と INT8 の MSE は FP32 と比較してそれぞれ 1.0 倍程度であり，精度劣化の度合いは小さい．一方で，CNNRNNLN の INT8 では，FP32 と比較して 10.27 倍の精度劣化が見られた．すなわち，このモデルにおいては浮動小数点により広いダイナミックレンジを表現する必要があり，INT8 量子化方式は精度の観点から更なる検討の必要がある．

5.4　電力効率評価

次に，Jetson 上で推論するときのエネルギー消費について，評価をした．電力計測では，jetson-stats [13] を使用した．Jeton には，電力モニターが組み込まれていて，jetson-stats はその値を読み取っている．しかしながら，本稿で扱うモデルは 1 フレームあたりの推論時間が高々数ミリ秒であるため，毎フレーム消費電力を計測することはできない．そのため本節では，以下の手順を用いて電力を計測した．

- 推論処理を 10000 フレーム分行う．ただし，テストデータは 10000 フレーム分ないので，同じテストデータをループして使う．
- 推論処理を実行している間，同時に 1 秒ごとに Jetson の電力モニターの値を読む．
- 立ち上げの時は電力の値が安定しないため，最初の 3 秒の電力のデータを取り除き，3 秒以降データのみを使用する．
- 総エネルギーは 平均電力 × (経過時間 – 3) をすることで求まる．
- 1 フレームあたりの消費エネルギーは，平均電力 ×1 フレームあたりの時間 で計算できる．ここでの計算時間は，推論処理の流れの (1) – (5) すべての計算を行う時間のことである．

表 7 に電力計測の結果を示す．SARNN を FP32 精度で推論するときに平均電力が最も高く 7.2W である．ほかのモデルについては，平均電力が 7W 以内である．表 3 によると，Jetson Orin Nano 4GB のカタログ上の消費電力は 7-15W であるので，余裕をもって処理できているとわかる．

推論精度を FP16 にすることにより，平均電力は FP32 に対して，SARNN で 0.93 倍，CNNRNN で 1.02 倍，CNNRNNLN で 1.00 倍となった．推論精度を INT8 にすることにより，平均電力は FP32 に対して，SARNN で 0.90 倍，CNNRNN で 0.95 倍，CNNRNNLN で 0.97 倍となった．全モデルいずれも INT8 が最小値となった．しかし，FP16 と FP32 の平均電力を比較すると，SARNN では FP16 のほうが小さいが，CNNRNN と CNNRNNLN では FP32 のほうが小さくなった．量子化を行うことにより，重みパラメータや中間層の入出力テンソルサイズが減少するため，メモリアクセスが減ることで電力削減効果がある．一方，FP32 は CUDA コア上，FP16 や INT8 は Tensor コア上で実行されるが，CUDA コアと Tensor コアが受け付けるデータレイアウトが異なる．本稿における FP16 は TensorRT により FP32 にフォールバックされることがあるので，量子化しない場合と比較して演算精度やレイアウト変換処理が追加され，それらの演算処理とメモリアクセスの電力消費が加算される．INT8 と SARNN の FP16 では，電力削減効果が上回ったことにより平均電力が減り，CNNRNN と CNNRNNLN の FP16 では，反対に電力増加が発生した．

フレームあたりのエネルギーについては，平均電力だけではなく，推論時間にも影響される．平均電力と推論時間がともに最も小さい CNNRNN の INT8 がエネルギー消費が 16mJ と最も小さい．平均電力と推論時間がともに最も大きい SARNN の FP32 ではエネルギー消費が 42mJ と最も大きい．前述のように，フォーマット変換の影響から INT8 による消費電力は CNNRNN で FP32 に対して高々 10%，FP16 に対して 6%程度の削減である．さらに，表 5 における推論時間の比較より，INT8 は FP16 に対して CNNRNN で高々 3%の性能向上であり，SARNN では 11%の性能低下となる．結果として，FP16 と INT8 の 1 フ

ⓒ2024 Information Processing Society of Japan

表 7　推論時のエネルギー消費

モデル名	演算精度	平均電力 [mW]	総エネルギー [mJ]	フレームあたりのエネルギー [mJ]
SARNN	FP32	7,219	397,395	42
	FP16	6,689	234,353	26
	INT8	6,490	259,854	28
CNNRNN	FP32	6,218	180,513	20
	FP16	6,350	152,544	17
	INT8	5,934	136,619	16
CNNRNNLN	FP32	6,601	297,334	32
	FP16	6,622	265,167	29
	INT8	6,429	250,989	28

レームあたりの消費エネルギーを比較すると，CNNRNN で INT8 が FP16 に対して高々 6%の削減となっており，SARNN では 8%程度大きい．以上より，深層予測学習処理の高性能・高電力効率化では，畳み込み層だけではなく，フォーマット変換も含めたデータ入出力処理まで考慮したシステム設計が重要であることがわかる．

6. まとめ

本稿では，ロボットの入力カメラ画像とセンサー入力から将来のカメラ画像とロボットの動作を推論する深層予測学習に対して，このロボット向け動作生成ライブラリの PyTorch 実装を TensorRT に移植を行い，FP16・FP32 と INT8・FP32 混合精度に量子化を行った．さらに NVIDIA Jetson Orin Nano 4GB 上でその効果を評価した．

評価の結果，FP16 で推論を行った場合，FP32 に対して精度が等倍程度で，最大 1.79 倍の高速化が得られた．INT8 量子化を行う場合，FP16 に対して速度低下や精度低下が発生するモデルあり，速度向上率は最大 1.04 倍であった．さらに，FP16 で推論することにより，FP32 に対してフレームあたりの消費エネルギーを最大 39% 削減した．INT8 では FP16 に対して，最大 9% のエネルギー削減となった．ロボット向け深層予測学習モデルの中間層において量子化しても FP32 で実行される計算があるため，推論過程において FP32 とのデータ変換による演算処理とメモリアクセスによる消費電力増大の局面があった．電力効率向上のためデータ変換も含めたハードウェア・ソフトウェアの全体システム設計が重要である．

謝辞　本研究の成果の一部は JST【ムーンショット型研究開発事業】【JPMJMS2031】の支援を受けたものです．

参考文献

[1] Broadbent, E., Stafford, R. and MacDonald, B.: Acceptance of Healthcare Robots for the Older Population: Review and Future Directions, *International Journal of Social Robotics*, Vol. 1, p. 319–330 (online), DOI: https://doi.org/10.1007/s12369-009-0030-6 (2009).

[2] Suzuki, K., Ito, H., Yamada, T., Kase, K. and Ogata, T.: Deep Predictive Learning : Motion Learning Concept inspired by Cognitive Robotics (2023).

[3] Ito, H., Yamamoto, K., Mori, H. and Ogata, T.: Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control, *Science Robotics*, Vol. 7, No. 65, p. eaax8177 (online), DOI: 10.1126/scirobotics.aax8177 (2022).

[4] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems 32* (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E. and Garnett, R., eds.), Curran Associates, Inc., pp. 8024–8035 (online), available from ⟨http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf⟩ (2019).

[5] NVIDIA: TensorRT SDK, NVIDIA Developer (online), available from ⟨https://developer.nvidia.com/tensorrt⟩ (accessed 2024-01-09).

[6] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).

[7] Ba, J. L., Kiros, J. R. and Hinton, G. E.: Layer Normalization (2016).

[8] Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M. and Blankevoort, T.: A White Paper on Neural Network Quantization (2021).

[9] Wu, H., Judd, P., Zhang, X., Isaev, M. and Micikevicius, P.: Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation (2020).

[10] Rusci, M., Fariselli, M., Croome, M., Paci, F. and Flamand, E.: Accelerating RNN-based Speech Enhancement on a Multi-Core MCU with Mixed FP16-INT8 Post-Training Quantization (2022).

[11] *TENSORRT SWE-SWDOCTRT-001-DEVG_vTensorRT 5.1.1 RC* (2019).

[12] : onnx/onnx: Open standard for machine learning interoperability, GitHub (online), available from ⟨https://github.com/onnx/onnx⟩ (accessed 2024-02-10).

[13] Bonghi, R.: rbonghi/jetson_stats, GitHub (online), available from ⟨https://github.com/rbonghi/jetson_stats⟩ (accessed 2024-02-04).

ⓒ2024 Information Processing Society of Japan