C-009

# SoC FPGA に向けた YOLOX_s モデルの量子化と精度比較
# Quantization and Accuracy Comparison of YOLOX_s Models for SoC FPGA

岩永 大翔‡　　中西 知嘉子‡
Hiroto Iwanaga　Chikako Nakanishi

## 1. はじめに

### 1.1 背景

近年 IoT 技術の発展により，エッジコンピューティングの重要性が高まっている。現在主流であるクラウドコンピューティングでは，端末とクラウドサーバ間のデータ送受信を行いサーバ側で処理するため，通信遅延やセキュリティリスクが課題となる。これに対し，エッジコンピューティングでは，データ発生地点に近いデバイスで処理を行うことで，リアルタイム性やネットワーク負荷の軽減，セキュリティ向上といったメリットがある。しかしながら，エッジデバイスは演算性能やリソースが限られており，画像認識など高負荷な処理では精度と速度の両立が課題となる。このため，エッジ環境に適した軽量かつ高速な AI モデルの開発と効率的に動作させる技術の確立が求められている。

### 1.2 目的

本研究では，物体検出モデルである YOLOX_s[1]を対象として，SoC FPGA で効率的に動作させるため，モデルを量子化し，弊研究室独自に開発した AI 推論ライブラリ「Ceras」で動作させることで，量子化がモデル精度に与える影響を評価する。さらに，量子化の影響を検証し，精度低下を最小限に抑えつつ効率的なハードウェア実装を実現するための指針を得ることを目的とする。

## 2. 使用技術・ツール

### 2.1 Ceras

Ceras[2]は，弊研究室で開発された C++ベースの AI 推論ライブラリであり，C++標準ライブラリのみで構成されるため環境構築が容易である。学習済みモデルの推論を C++で行い，FPGA との学習済 AI モデルを nnet 形式へ変換し使用している。nnet 形式とは，Ceras で推論するためのモデルのパラメータが格納されているファイルである。さらに，ソフトウェアとハードウェア間のデータコピー時間を削減するための情報などハードウェアでの効率良く動かすための情報が付与されている。ハードウェア部は C++コードで作成され，そのコードを高位合成により回路生成を行っている。この高位合成に用いる C++コードは，ソフトウェアのみでソフトウェアとハードウェアによる協調動作の検証が可能であることにより，回路実装前に検証ができることで開発機関の短縮に貢献している。

### 2.2 YOLOX_s

YOLOX_s は，物体検出モデルである YOLO シリーズの 1 つであり，高精度かつ高速な推論を目的として設計されたモデルである。従来の YOLO モデルと異なり，アンカーフリーを採用することで物体検出精度と柔軟性を向上させている。また，パラメータ数の削減により，計算負荷とメモリ消費を抑え，高い処理速度と低消費電力を実現している。

### 2.3 Vitis AI

Vitis AI[3]は，Xilinx 社が提供するエッジ AI 向けの開発プラットフォームであり，FP GA や SoC 上で効率的な AI 推論を実現するための包括的なツール群を備えている。本プラットフォームには，様々な最適化済モデルを提供する「Model Zoo」や 8bit 固定小数点への量子化を行う「AI Quantizer」などのツールが含まれている。これらのツールを用いることで，FPGA の複雑なハードウェア設計に精通していなくても，AI アプリケーションの開発を容易になり，開発の大幅な短縮が期待できる。

### 2.4 ONNX

ONNX(Open Neural Network Exchange)[4]は，深層学習モデルを表現するためのオープンな標準フォーマットである。ONNX を使うことにより PyTorch や TensorFlow など異なる深層学習フレームワーク間において学習したモデルを相互に運用できる。2.1 節で説明した Ceras を使用するためには，この ONNX 深層学習モデルを中間表現として利用する必要がある。

## 3. 提案手法

PyTorch や Keras などのフレームワークで学習された深層学習モデルを nnet 形式へ変換し，FPGA と CPU の協調動作による効率化のために導入した手法について述べる。本手法は，学習済モデルを中間表現である ONNX 形式を経由して nnet 形式へと変換することで，推論の高速化と最適化を支援するものである。

nnet 形式への変換プロセスにおいて，本研究では，モデルの量子化と協調動作のための支援機構の導入の 2 つを行った。モデルの量子化では，モデルの演算精度を FP32 から INT8 へと変換し，演算の高速化を実現する。また，協調動作のための支援機構の導入では，FPGA と CPU の協調動作を行うための層の融合化，共有メモリブロック割り当て，実行先の指定という 3 つの機構を導入する。

これら 3 つの機構は，すべてモデル変換時に静的に適用されるため，推論実行時に追加的な制御を行う必要がなく，一貫性のある高効率な協調推論実行が可能となる。以下の各節では，これらの手法について説明する。

### 3.1 モデルの量子化と nnet 形式モデルへの変換

#### 3.1.1 量子化パラメータの算出

本研究では，学習済の FP32 形式の YOLOX_s モデルに対し，Xilinx 社が提供する Vitis AI ツールチェーンに含まれる量子化ツール「Vitis AI Quantizer」を用いて量子化パラメータを算出する。このツールは，モデルの演算精度を INT8 へと変換するために必要な情報を生成するものである。具体的には，キャリブレーションと呼ばれる手法を用いて，代表的な入力データセット(キャリブレーションデータセット)を量子化前の FP32 モデルに入力し，各層における重みやバイアスの最小値や最大値といった統計的分布を解析

する。この解析結果に基づき，モデル内の各層において最適なスケールとゼロ点が自動的に決定される。ONNX 形式のモデルに，これらの情報を含んだ形で出力する。この処理は，再学習を必要としない。

#### 3.1.2 nnet 形式への変換とパラメータ格納

次に，量子化パラメータが埋め込まれた ONNX 形式のモデルを，Ceras で利用可能な nnet 形式へと変換する。この変換プロセスにおいて，前節で算出した最適なスケールとゼロ点を用い，モデル内の各層の重みを INT8 形式の整数値として格納する。一方，バイアスについては，INT8 表現にすると，加算処理にて桁落ちが発生し，精度が著しく低下する可能性がある。この問題を抑制するため，バイアスはより広いビット幅を持つ INT32 形式で保持している。これにより，Ceras での推論時に INT8 の重みと INT32 のバイアスを用いた整数演算が可能となる。

#### 3.1.3 量子化における特定層への対応

YOLOX_s モデル量子化にあたり，Sigmoid-Mul 層への対応が必要となる。Sigmoid – Mul 層は，量子化処理での精度維持が課題となる。これらの層を浮動小数点での演算を行う場合，Conv2D – Sigmoid – Mul 層と続く場合は，Conv2D 層の後に逆量子化が必要となる。しかし，逆量子化を行う手法であれば，量子化・逆量子化が頻繁に発生し，速度が著しく低下する恐れがある。そこで，以下の順で Conv2D 層への統合を検討する。

1. Sigmoid 層 + Mul 層を Swish 層へ変換
2. Swish 層を HardSwish 層へ変換
3. HardSwish 層を Conv2D 層へ融合

はじめに，Sigmoid 層+Mul 層は Swish 層へ変換した。次に，変換した Swish 層を HardSwish 層へと変換する。式(1)に Swish 層，式(2)に HardSwish 層の演算式を示す。

$$Swish(x) = x \times 1/(1 + e\hat{}(-x)) \quad \cdots (1)$$

$$HardSwish(x) = x \times \min(\max(0, x+3), 6)/6 \quad \cdots (2)$$

Swish 層における活性化関数は非線形な特性を持つため，固定小数点での実装が困難である。そのため，Swish 関数を近似することで，より単純な演算を行う HardSwish 関数への変換を行うことで，回路に組み込むことが可能となる。最後に，HardSwish 層を Conv2D 層に融合を行う。nnet 形式の Conv2D 層のモデル情報に HardSwish 処理の実行を適用するという属性を付加することにより，HardSwish 層を吸収する。この処理に伴い，HardSwish 層はモデルから削除され，Conv2D 層の出力は HardSwish 層の出力先に再出力される。また，Conv2D 層への融合を行うことにより，FPGA における回路起動数が抑えられることで，高速化にも貢献できる。これらの処理により，固定小数点演算が可能となり，層数が削減されることで，演算処理速度の向上が見込まれる。

#### 3.1.4 スケール整合性

量子化されたモデルでは，各演算層が独立したスケールを持つため， 層間でスケールが一致しない場合が発生する。このため，各層の出力を次の層にて適切に扱えるようにスケールを調整する処理が必要となる。一般的には，各演算の前後で再量子化やスケール変換が行われるが，これらは実行時の演算負荷や処理の非効率化を招く要因となる。

本研究では，モデル構造をあらかじめ静的に解析し，各層の出力スケールを次の層の期待するスケールに揃えるための情報をモデルに付加する。この処理はモデルの nnet 形式へ変換時に行い，各層の出力スケールと次層の入力スケールの差分から適切な係数を計算し，情報として埋め込むことで対応する。層の処理後に差分から得た係数を基に，ビットシフト演算を行うことで，次層の入力スケールに合わせたスケールで出力することができる。ビットシフト演算により，従来の再量子化やスケール変換に比べ，大幅に演算負荷を抑えられる。

### 3.2 共有メモリのブロック割り当て

SoC デバイスでは，CPU と FPGA が共有メモリを介してデータをやり取りする。この共有メモリを効率的に利用するため，モデル内の各層が生成する中間データの生存期間を静的に解析する。この解析結果に基づき，共有メモリをブロック単位で管理し，異なるデータであっても生存期間が重複しない場合は同じメモリ領域を再利用するように割り当てる。これにより，データ領域の重複を防ぎつつ，モデル全体で必要なメモリ総量を削減できる。

### 3.3 実行先の指定

モデル内の各層について，演算特性や量子化を考慮し，FPGA で実行するか CPU で実行するかをモデル変換時に静的に指定する。例えば，量子化に適した Conv2D 層は FPGA に，sigmoid などの複雑な関数や量子化による誤差が大きい層は CPU に割り当てることができる。この実行先の事前指定により，推論時の動的なタスクスケジューリングを不要にし，CPU と FPGA 間のデータ転送も必要な箇所に限定できるため，協調動作時の不要なデータ転送を削減できる。

## 4. まとめ

本研究では，エッジデバイス上で物体検出モデル YOLOX_s を効率的に動作させるため，Ceras を基盤にモデルの量子化と FPGA/CPU の協調動作を最適化する手法を提案した。モデルの演算精度を INT8 へ量子化する際，重みとバイアスのビット幅を調整し，特定層に専用処理を施すことで精度低下を抑制した。さらに，FPGA/CPU の協調動作のための支援機構を導入した。これらの最適化は全てモデル変換時に静的に適用され，推論時に追加コストがなく高速な処理が可能となった。

## 5. 今後の展望

今後の課題として，より多様な AI モデル形式へのサポートが挙げられる。また，学習済モデルを容易に Ceras で使用できるツールの開発を進めることで，性能向上や開発期間の短縮を目指す。

### 参考文献

[1] YOLOX, (https://arxiv.org/abs/2107.08430) (2025/06/08 参照)
[2] Ceras, 西岡駿, 中西知嘉子, "機械学習ライブラリの C 言語化の実現", 電子情報通信学会ソサイエティ大会(2021).
[3] VitisAI,(https://www.amd.com/ja/products/software/vitis-ai.html) (2025/06/08 参照)
[4] ONNX, [https://qiita.com/motoJinC25/items/d662be70b6b9b8ebbaea] (2025/06/08 参照)

† 大阪工業大学 情報科学研究科 情報科学専攻
Graduate School of Information Science and Technology
Osaka Institute of Technology
‡ 大阪工業大学 情報科学部 情報知能学科
Department of Information and Computer Science
Osaka Institute of Technology