C-014

ディープニューラルネットワークを利用したシステムに対する高効率な検証法
A method of highly efficient verification for systems using deep neural networks

○白石 忠明† 高橋 寛‡ 王シンレイ‡
Tadaaki Shiraishi Hiroshi Takahashi Senling Wang

1. はじめに

信頼される AI（Trusted AI）の創出が望まれており，そのために， AI システムの信頼性・安全性を確保する技術の開発は喫緊の課題である[1]．一般的に，開発したディープニューラルネットワークの判別精度を求めるためには，膨大なテストケースをディープニューラルネットワークに与えて，予想と正解を比較して判別精度を求める．現状の判別精度の検証においては，膨大テストケースによって長大な検証時間が必要であり，さらに，判別精度が悪い場合に，その理由を説明するために必要な情報も得られない．

そこで，本稿では，開発されたディープニューラルネットワークを高効率・高精度に検証できる検証システムを提案する．提案する検証システムの特徴は，ディープニューラルネットワークの一つの層と等価な回路を FPGA 上に実現し，ディープニューラルネットワークの各層を並列実行し，高効率にテストケースを実行できる．また，FPGA で計算された層の計算結果と期待値を比較することによって，許容範囲を超えた計算結果をもつ層を指摘することができるので，判定精度の低下の原因となる層を指摘できる．

評価実験においては，物体検出アルゴリズムの YOLO を実装した AI システムの検証を実施し，ソフトウェアとして YOLO を検証する時間のおよそ 100 分の 1 以下の検証時間に短縮できることを明らかにする．また，誤った結果を得た際に，許容範囲を超えた計算結果をもつ層を指摘できることを示す[2].

2. AI の FPGA 実装

ここでは，AI アルゴリズムを FPGA に実装することに関して述べる．

画像 Convolutional Neural Network（CNN）系の推論処理は，処理量が膨大であるため，端末側での処理では動画像 30fps に対し，1fps 以下の処理速度になる．このため，水冷設備の伴うハイエンドな GPU 等の装備を有するクラウドサーバーに画像データを送信し，サーバーにて推論処理を実行した後，その結果を端末側に戻すクラウド処理型が一般的な構成となっている．

この構成では，推論処理に常に通信が伴い，通信品質の影響を直接受ける事となる．そこで，「エッヂ AI」の言葉の元に端末側で推論処理を完結させる研究が 2015 年以降，世界的に進められて来た．その中で，GPU を FPGA にて代替する取組みがあり，本研究に於いて Full YOLO Version3 と呼ばれる AI アルゴリズムを，Xilinx 社のミッドレンジクラスの FPGA である Kintex-7 に搭載した．

2.1 量子化学習

AI アルゴリズムにおいて，Python や TensorFlow 等のソフトウエア記述された処理を直接移植したのでは回路化できない部分や冗長な部分がある．そのために量子化学習により軽量化や最適化が必要である．

2.1.1 学習データ

学習に際し，国際標準データである COCO2014 を使用した．同データは 80 種類（クラス）の検知対象物を網羅している．検知性能は，mAP（mean Average Precision）値で評価されるが，割合の異なるクラスの混在評価では，誤検出等にて他のクラスの影響を受け正確性を欠くため，person（人）に絞り学習を行った．表 1 に示す如く person クラスが半数以上を占めるデータセットである．

表 1　学習データ

No	Data Set	全体枚数	内 person 枚数	person 率
1	train2014	82,783 枚	45,174 枚	54.6%
2	val2014	40,504 枚	21,634 枚	53.4%

2.1.2 乗算回路の削減

CNN 処理に於ける主要な演算は乗算である．乗算回路は多数の加算回路から構成され，回路規模が大きく，動作遅延も大きな回路である．CNN 処理では，特徴マップ値と重み係数の乗算が最初に行われるが，重み係数は学習により人工的に作られる数値であり，これは数値制御可能な範囲が広い．

そこで，重み係数値を 2 のべき乗値に成る様に学習コントロールする．2 入力の乗算に於いて，一方の数値が必ず 2 のべき乗値である事が保証されている場合は乗算演算を bit シフト演算に置換えられる．

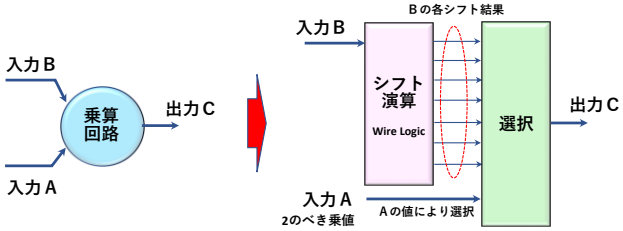

図 1　乗算器の置換えイメージ図

図 1 に於いてシフト演算は，シフトレジスタ等の回路では無く，配線の組み換えだけの配線論理である．入力 A の範囲に応じて，予めシフトしておいた配線群の中から，入力 A の値で選択出力する．

2.1.3 学習コントロール

本稿で作成したシステムでは，32bit 浮動小数点型を，最終的に 8bit 整数型になる様に学習コントロールしている．特徴マップや重み係数に対し，当該処理層での平均値を取り，平均値との差分により数値幅の推定を行う．数値幅を 256 で除算し，分解能を生成する．

数値群を「分解能×階級番号」の形式とし，bit 数がより少ない階級番号で演算した後，分解能を乗算して数値を元に戻す．重み係数は，学習時に 2 の log 演算を行い，2 のべき乗化を図ると共に，5bit 化を行う．

2.2 AI 専用プロセッサ

ここでは，本稿で実装した AI 専用プロセッサに関して述べる．

エッヂ AI においては，AI 専用プロセッサの回路規模を可能な限り小さくする必要がある．CNN 系の処理は，演算量が膨大である．一方，各中間層で特徴マップのサイズや数は異なっているが，基本的に類似する処理を繰返している．

これらのことに着目して，類似する処理を抽出して共通化し，層毎に処理動作を定義したμコードを設計した．このμコードに従い処理する事で，使用する回路の共通化を図った AI 専用プロセッサ回路を設計した．

2.2.1 μコード

YOLO では CNN 演算の他に，過去の層の特徴マップ値と現在の処理層の特徴マップ値との和を取る「Residual」処理や，異なる特徴マップを連結してサイズ変換を行う「Up sample」処理がある．また，正規化処理として，「Batch Normalization」処理や，活性化関数として「ReLU」処理等があり，当該処理層でこれらをどの様に処理するかを定義するものである．表 2 にその例を示す．

表 2　μコードの例

No	μコード種別	値
1	コマンド	0:conv 1:add 2:up_sample2d 15:end
2	層番号	0～105
3	Leaky-ReLU	0:off 1:on
4	Batch Normalization	0:off 1:on
5	Batch Normalizationアドレス	DDR上の物理アドレス
6	Kernel アドレス	DDR上の物理アドレス
7	入力チャネル数(8種類)	3/ 32/ 64/ 128/ 256/ 512/ 768/ 1,024
8	出力チャネル数(6種類)	32/ 64/ 128/ 256/ 512/ 1,024
9	入力サイズ（6種類）	0/1/2/3/4/5：（416、208、104、52、26、13）
10	出力サイズ（6種類）	0/1/2/3/4/5：（416、208、104、52、26、13）
11	Kernel サイズ	0:3×3　1:1×1
12	Stride	0:1　1:2
13	小数点位置	0:01
14	入力CHアドレス	DDR上の物理アドレス
15	入力CHバイト数	0/ 416/ 1,664/ 6,656/ 23,296/ 86,528/ 346,112
16	出力CHアドレス	DDR上の物理アドレス
17	出力CHバイト数	0/ 416/ 1,664/ 6,656/ 23,296/ 86,528/ 346,112

本方式の特徴的な利用形態を挙げる．物体検知の処理内容の異なる複数のカメラに対し，或る時間にどれか一つしか使われず，且つ，その時，どのカメラが使用されているか分かっているケースにおいては，コスト削減を目的にカメラ毎に物体検知用の FPGA を持たずに，単一の FPGA の共用使用が可能である．

一般的に，単一の FPGA に複数の役割を持たせる場合，それぞれの回路データをダウンロードし直す必要があり，それだけで 1 秒以上の書き換え時間が必要となる．この切替えのトータル時間が長いほど，ユーザーが操作性に違和感を覚える．本方式では，それぞれの処理のμコードを予めメモリに書込んでおき，カメラの切り替えに応じてそのメモリのアドレスを変えるだけで異なる物体検知を形成しうる．アドレスの切り替え操作は，1 ミリ秒以下にできる．

2.2.2 回路規模

ソフトウエア処理に対する回路処理の特徴として，並列処理性がある．回路処理では，処理回路を並べる事で，同時処理数を増やす事が可能である．同時処理数が多いほど，全体の処理時間は短く成り，その結果，処理の高速化が図れる．回路規模は並列度に比例する．

本実装では表 3 の FPGA に，表 4 の並列数にて搭載した．

表 3　搭載 FPGA

メーカー	シリーズ	型	SoC	DDR
Xilinx	Kintex-7	XC7K325T	無し	64bit×1

表 4　処理並列数

総合並列度	画素並列	演算並列
128（16×8）	16	8

なお，外部 SDRAM（DDR）とのアクセス速度 1.6Gbps と内部処理クロック速度 200MHz との関係より，内部バスは 256bit 幅としている．画素並列とは，1 回のバスサイクルで何画素分が纏めて処理可能かを表す．先述の条件での実装した場合の回路情報を表 5 に示す．並列度に応じて回路規模は変動し，演算並列度が低くなると，比例して回路規模も小さくなる関係である．

表 5　回路情報

HDL 行数 Verilog-HDL	LUT as Logic	レジスタ数	Block RAM
16,978	172,270	168,068	340.5

3. AI アルゴリズムの検証法

ここでは，AI アルゴリズムの検証方法を提案する．

量子化学習の過程にて，所望する bit 幅やデータ型にて検知性能に対する影響を繰り返し評価する．これにより最終的に得られた推論シミュレータと回路の機能的等価性を検証する必要があり，ここでは，これを品質保証と定義する．その際，推論結果と云う幅のある数値の比較に留めず，各中間層で生成される特徴マップ値を 1bit 単位で全て比較するのが好ましい．

提案する検証方法は，全ての試験データを評価することに加えて，設計過程のデバッグに使用できるため，品質保証に留まらず設計効率を向上できる．

3.1.1 検証時間における課題

AI アルゴリズムの検証を実行する際，1 枚の画像に対する推論処理において，1 億回を超える畳込み演算を行う過程で，62,556,195 画素の特徴マップを生成する．これを，標準試験データ 21,634 枚に対し実行しなければならない．商用の HDL-Simulator（Model-SIM）を用いた場合，1 枚の実行に約 24 時間（1 日）を要する．これを全ての試験データで実行した場合，約 55 年間を要する計算となり，検証が不可能である．

3.1.2 対策

検証の高効率化のために，提案する検証法では，推論シミュレータと推論 FPGA を，実時間動作させて，検証を実行する方式である「リアル動作検証技術」を開発・実装する．提案する検証法の概要を図 2 に示す．

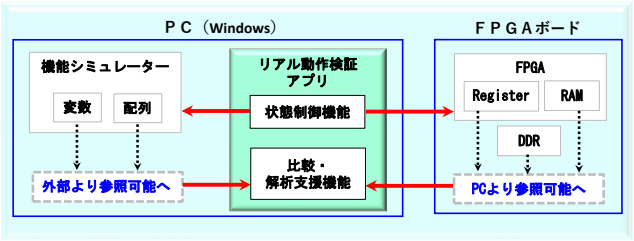

図 2　提案する検証法の概要

3.1.3 効果

⑴ 高速化

FPGA の実動作に近い速度で検証を行う事で，高効率な検証を可能とし，HDL-Simulator に対し 100 倍以上の速度が得られた．

リアル動作検証方式の実行速度は，推論シミュレータの実行速度がボトルネックとなる．ここでの推論シミュレータは，回路アーキテクチャーに合わせたソフトウェアコードであり，ビヘイビアレベルで記載されたシミュレータよりも処理負荷が大きい．

また，同一機能でも，Python と C++の記述言語の違いや，使用するライブラリィおよび，ソフトアーキテクチャーに加え，実行する環境により大きく実行速度が異なる．したがって，FPGA の処理速度に対し一概に決定することが困難であり，HDL-Simulator との速度比を正確に評価することは容易ではない．定性的には，高速な推論シミュレータであれば，FPGA の実動作に近い速度のリアル動作検証が行える関係となる．

なお，画像系の CNN 処理は，推論処理が遅い代表格であるが，それ以外の一般的なデザインであれば，FPGA の実行時間とほぼ同等な処理速度である．

⑵ ファイルレス化

一般的な検証手法では，出力信号結果をファイルに記録し，期待値を記述したファイルとそれを比較する．この場合，検証サイクル数の長大化に応じて大容量のファイルが生成され，ディスク容量を圧迫する．本開発技術では，比較はメモリの値を直接比較する方式のため，出力結果の記憶用のファイルは必要としない．

⑶ デバッグ性の向上

Full YOLO Version3 では 105 層の中間層の処理を経て，推論結果が表示されるが，早い段階の層でエラーが検出された場合，それ以降の層の処理を実行する必要はない．

リアル動作検証技術では，回路の内部信号の比較が容易であり，処理の早い段階で不一致を検出できる．このため，出力結果から入力に向かってのバックワード解析ではなく，入力から出力に向かったフォワード解析が可能であるため，CNN に対するデバッグ性が高められる．

4. 信頼される AI システムに向けて

今後，益々市場で稼働し，社会基盤を構築する AI システムにおいて，その信頼性を向上させるために必要不可欠な技術が AI システムの核となる AI 専用プロセッサの検証およびその検証結果に基づく AI 専用プロセッサのデバッグ（改修）の効率化である．これらの技術は，AI システムの機能安全を保証するためにも必要不可欠である．

信頼される AI システムを実現するためには，AI システムを構成するニューラルネットワークにおいて，その層毎の重みづけの状況を短時間に観測できる方法を確立し，所望の結果を得られなかった AI システムの原因を説明することが必要である．この技術を確立することが，結果を説明できる AI システムの構築に貢献できると考えている．

5. むすび

従来の FPGA 検証環境は，数百万サイクルを超える長大な検証は困難であり，1000 万円超の高額な ASIC エミュレータの導入が必要であった．一方，本稿で提案した検証方法では，10 万円～20 万円程度の開発キットボードで同等な検証を可能とする．

AI の信頼性を保証するためには，推論シミュレータ通りに FPGA や GPU が振舞っているか否かを詳細に検証することが必要である．さらに，AI システムにおいて検出すべき物が検出されない未検出や，検出されてはいけない物が検出される誤検出が生じた場合に，誤りを生じる原因となる重み係数の影響を解析する手法が必要である．さらに，この検証結果に基づいて自動学習補正が可能であると考える．

本稿では，エッヂ AI 化の適用事例を元に，検証法を開発した．今後は，検証法を利用して，ＡＩシステムの誤検出に対する補完機能の強化法を検討する．

商標

⑴ Xilinx は，ザイリンクスインコーポレーテッドの登録商標である．

⑵ その他の会社名，製品名は，それぞれの会社の商標又は登録商標である．

参考文献

[1] S. Kundu et al. “Effective In-field Testing of Deep Neural Network Hardware Accelerator”, Proc. VTS, 2022, pp.1-4. (2008).

[2] Joseph Redmon, Ali Farhadi : ” YOLOv3: An Incremental Improvement”,arXiv preprint arXiv:1804.02767v1(2018.4)

† 三菱電機ソフトウエア株式会社
Mitsubishi Electric Software Corporation
‡ 愛媛大学
Ehime University