HDLを用いた畳み込み層の設計と カメラ信号を用いたレイテンシー評価†

亀阪 亮紀*・星野 孝総*

畳み込みニューラルネットワーク（CNN）は画像認識分野において大きな注目を集めており、様々なアプリケーションへ適用され、近年では、IoT デバイスなどの組み込みシステムでの実用化が期待されている。しかし、CNN は計算コストが高く、高速化には一般的に GPU が必須で、小型の組み込み機器などの省電力システムへの実装は消費電力等の観点から困難である。また、近年 FPGA は自動車やセキュリティシステム、工業製品の欠陥検査などの画像処理技術分野に応用されている。ハードウェアアクセラレーションは、処理速度を向上させる手法の 1 つであり、画像処理の分野でよく用いられる。そこで、本研究では、畳み込みニューラルネットワークのハードウェア化を目的とし、畳み込みニューラルネットワークの高速化を目指す。本稿ではハードウェアベースの畳み込み層の設計・実装を行い、リアルタイム性の検証を行った結果について報告する。

キーワード：畳み込みニューラルネットワーク、FPGA、パイプライン処理

1. はじめに

深層学習は人工知能を支える大きな技術であり、様々な分野へ応用され始めている。近年、画像認識、音声認識、自然言語処理などの様々な分野で深層学習が適用され、IoT デバイス等の組み込みシステムでの利用が期待されている [1–3]。しかし、深層学習は計算コストが高く高速化には一般的に GPU が必須 [4,5] で、小型の組み込み機器などの省電力システムへの実装は消費電力等の観点から困難である。特に現在の深層学習技術によるリアルタイムのオブジェクト検出には GPU が不可欠である。深層学習の最大の処理は、GPU アーキテクチャによる畳み込み処理と行列演算で、高速実行には、高速なメモリアクセスと積和演算の並列計算パフォーマンスが必要である [6–8]。FPGA は CPU や GPU よりも優れた電力性能を備え、自動車、セキュリティシステムなどの画像処理技術分野に応用されており、重要性は日々高まっている [9–13]。FPGA に関連する研究も数多くあり、画像処理フィルターを FPGA によってハードウェア化することで、CPU に比べ優れた処理性能を発揮することが報告されている [14,15]。そこで、本研究では、畳み込みニューラルネットワークの画像処理演算部を FPGA 向けパイプライン化することで、演算コストを減らし、計算速度の向上を目指す。本稿では FPGA を用いて 2 層の畳み込み層及び、全結合層に向けた多層ニューラルネットワークを実装し、処理時間の検証を行った結果について報告する。

2. FPGA によるリアルタイム処理

FPGA は Field Programmable Gate Array の略で、ユーザー

が任意の回路をハードウェア記述言語（HDL）で設計し、プログラムとして書き込むことができる。本研究では、Intel 社製 Cyclone V SoC を搭載した DE1-SoC を使用した。OS が動作可能な CPU や 1 GB の DDR3SDRAM など搭載されている。そのため、FPGA の出力を OS 上の優れた GUI や OpenCV 画像処理ライブラリを使用して確認できる環境を 1 つのシステムにまとめることができる [16]。また、OS 側から FPGA 側のレジスタ等にアクセスできるため、畳み込みフィルター係数やニューロの重みの変更、プーリング方式の変更等、ネットワーク構造の切替えが容易に行うことができる。FPGA と CPU・GPU との処理方法の違いを図 1 に示す。カメラからの RAW データを直接処理する場合、CPU や GPU を用いる処理では、すべてのカメラデータを一度メモリに保存する必要がある。GPU を用いた演算の場合、入力画像を縮小処理した上で、さらに必要なメモリ領域を Malloc（確保）してニューロや畳み込みのパラメータ群と縮小した画像データを展開してから演算を行う [17]。この縮小、メモリ確保、パラメータの展開に一般的には 1~100 ms の時間を必要とする。また、メモリへ完全に画像データを置くためには、カメラの速度に依存するが、一般的に 30~50 ms の時間を必要とする。しかし、FPGA 使った処理ではその必要がなく、カメラからのデータを直接処理できる

図1 FPGA・CPU・GPUそれぞれの処理方法の違い

† Design of the Convolution Layer Using HDL and Evaluation of Latency Using a Camera Signal
Ryoki KAMESAKA and Yukinobu HOSHINO

* 高知工科大学大学院工学研究科
Graduate School of Engineering, Kochi University of Technology

図2 パイプライン処理

図3 ラインバッファ構造 [14]

ため、メモリへのアクセス時間を短縮できる。したがって、レジスタ転送レベルで画像処理を実行でき、カメラクロックに同期した画像処理を行うことができる。次に、本研究におけるリアルタイム処理について述べる。本研究では、カメラが N 番目の画像 1 フレームの転送が終了する前に、 $N-1$ 番目の画像処理を終えていることをリアルタイム処理と定義する。こうすることで、カメラのスループットを守りながらフレーム落ち無しに処理を行うことができている状態になる。つまり、カメラのフレーム落ちがないため、畳み込み処理後の結果をカメラデータとして扱うことができ、畳み込みカメラシステムを構築できる。そこで、本実験では、図2に示すようにカメラの取り込み終了時から、処理が完了するまでの時間を計測することとした。本研究では、このようなパイプライン処理を実現するためにラインバッファ構造を利用した [14]。ラインバッファとは横 1 ライン分の画素を格納するメモリのことで、画像処理の畳み込みフィルタが 3×3 の場合、3 本用意する必要がある。また、3 本用意した場合、 1×1 から 3×3 のフィルタ処理に対応することができる。図3にラインバッファの動作を示す。カメラから 1 画素分のデータが転送されてくるたびに破線枠で囲った部分の画素データに対して畳み込み処理を実行する。この畳み込み処理は、ラインバッファの回路と独立して並列動作するため、バッファの転送を待つ必要がない。ラインバッファに格納された画素データ群は、カメラクロックに同期してシフトしていくため、フレーム単位ではなくピクセル単位で畳み込みを実行できる。このように、カメラから画素データが転送されて来るタイミングで行うパイプライン処理構造にすることで、カメラ 1 画素ずつの速度に合わせて処理後のデータを 1 画

図4 ハードウェアアーキテクチャ

図5 FPGA のレイテンシーの定義

素ずつ出力する構造にできる。したがって、カメラからのデータを取り込みながら処理することができる。

3. 畳み込みカメラシステムの構築

3.1 システムアーキテクチャ

本実験で構築した畳み込みカメラ画像処理システムの回路を図4に示す。本実験では、2 層の畳み込み層（畳み込み演算 + MAX プーリング演算）を実装した。カメラと FPGA の動作クロックが異なるため、非同期式 FIFO を用いることで、速度の違いを吸収している。カメラから取り込まれた画像に対して、 3×3 畳み込み処理と 2×2 の MAX プーリング処理が行われ、2 層目の回路へ転送される。その後、HPS 部の Linux とのメモリ共有エリアに DMA 転送を行うことで、Linux から OpenCV を用いて処理結果を確認する。

3.2 実験方法

本実験における FPGA のレイテンシーを図5に示す。本実験では、畳み込み層に転送されてきた画像データに対して 1 フレーム分の画像データ取り込み終了時から、畳み込み処理・MAX プーリング処理を行い、1 フレーム分の画像が HPS 側の RAM に転送されるまでの処理時間を計測した。SystemVerilog 言語を用いて設計し、QuartusII を用いて FPGA へ実装した。カメラ画像は一般的な解像度規格 SXVGA (1280×960) を用いて実験を行った。また、実験に使用した FPGA ボード DE1-SoC のシステムクロックは 200 MHz、使用した CMOS カメラ (TRDB-D5M) のピクセルクロックは 92 MHz である。

3.3 実験結果

各層における 1 フレーム分の処理結果を表1に示す。1 層目の 1280×960 の画像に対する畳み込み処理・MAX プーリング

表 1 レイテンシーの比較

Device	Layer 1 [μ s]	Layer 2 [μ s]
FPGA	44.7	25.5

図 6 ニューラルネットワークアーキテクチャ

処理のレイテンシーは $44.7 \mu\text{s}$ であった。2 層目の 640×480 においては、 $22.5 \mu\text{s}$ であった。FPGA のリソースの消費量としては、1 層・2 層合わせて 21% であった。畳み込み層の多層化を想定した場合、プーリング処理等で画像サイズが小さくなるため、リソースの消費量の増加は小さくなっていくと考えられる。FPGA において、カメラから取り込まれたデータは FIFO に入れることで、動作クロックの違いを吸収している。そのため、取り込み後のデータは FPGA の動作クロック 200 MHz で動作している。つまり、1 クロックあたり、5 ns で動作している。実験結果より、1 層目のレイテンシーが約 $45 \mu\text{s}$ となり、約 9000 クロック遅れとなっている。すなわち、次フレームの約 7 ライン遅れとなっている。2 層目のレイテンシーが $25 \mu\text{s}$ であり、約 5000 クロック遅れとなっており、1 層・2 層合わせて約 14000 クロックの遅れとなる。1 層目と 2 層目合わせて合計 10 ラインの遅れとなり、次フレームの約 10 ラインが取り込み終了時に 2 層の畳み込み層での処理が行われていることになる。本実験では、フレームのサイズは 1280×960 であり、960 ラインの内の 10 ラインは、全体の 1% 程度の遅れとなっている。つまり、次フレームの取り込みが完了するまでに処理が完了していることになる。本研究では、リアルタイム処理の定義を次フレームの取り込み終了までに前の処理が完了していることとしている。レジスタ転送レベルでのリアルタイム画像処理が実行できていることを確認できた。2 層の畳み込み層におけるレイテンシーが次フレームの 1% 程度であり、FPGA のリソース消費量の観点からも、さらなる多層化にも対応可能である。

4. FPGA 多層ニューラルネットワーク

4.1 実験内容

本章では、CNN の全結合層実装に向けて、多層ニューラルネットワークを FPGA 上に実装し、予測時間の検証を行った実験について述べる。実験概要としては、4 層の多層ニューラルネットワークを FPGA に実装し、1 フレームの画像の予測を行い、予測時間に対する検証を行った。図 6 に FPGA 上に実装したニューラルネットワークのアーキテクチャを示す。構成としては、前章の実験と同様の SoC FPGA を用いて、HPS と FPGA のメモリ共有エリアに重み情報と入力画像を用意し、FPGA 側の積和演算器でニューロ演算を行った。演算結果をメモリ共有エリアに格納し、ループ型ステートマシンにより逐次

図 7 FPGA への実装フロー

表 2 量子化前後のモデルの精度比較

	量子化前 (float 型)	量子化後 (int 型)
精度 [%]	79.77	79.33
再現率 [%]	84.72	84.23

図 8 量子化前後の認識結果

処理でニューロ演算を行う。実装手順 (図 7) としては、まず、Python で TensorFlow を用いて GPU 上で学習を行い、学習済みモデルを作成した。FPGA での浮動小数点演算は FPGA のリソース枯渇の原因となる可能性がある。この学習済みモデルは浮動小数点であるため、TensorFlow Lite を用いて、8 bit 整数に量子化を行った。この量子化された重みパラメータを用いて、HDL による FPGA 上での動作ニューラルネットワークモデルを作成し、推論時間の比較検証を行った。

実装したニューラルネットワーク構造は、4 層のニューラルネットワークで、入力は 28×28 、隠れ層 1 のニューロの個数が 60 個、隠れ層 2 のニューロの個数が 20 個、出力 2 の 2 クラス分類器である。用いたデータセットは、高知県大豊町に設置が検討され、研究開発が行われている AI 搭載型害獣捕獲システムに用いられている猿・人データセット [18] を用いた。

4.2 実験結果

表 2 に TensorFlow で事前に作成した学習済みモデルの精度及び、TensorFlow Lite による量子化後の精度を示す。また、図 8 に量子化前後の混同行列を示す。量子化後のモデルは猿・人それぞれ 1 枚ずつ誤認識している結果となり、認識精度に差は 1% 未満に抑えており、量子化による精度の劣化についてはほとんど見られなかった。

次に、FPGA への実装を行い、推論時間の結果を表 3 に示す。画像 1 枚当たり、推論に $242 \mu\text{s}$ かかる結果となった。FPGA は 200 MHz で動作しているため、1 クロックあたり 5 ns で動作していることになる。本実験でのアーキテクチャでは全結合層の各層・各ニューロが逐次処理で行われている。このため重みの個数分のクロックが計算に必要となる。そのた

表3 予測時間の結果と理論値

Device	計測値 [μ s]	理論値 [μ s]
FPGA	242	241

め、合計約 48000 クロック必要になる。FPGA 側の内部クロック数は 200 MHz で、1 クロック 5 ns となる。FPGA 内部の On Chip RAM 等のレイテンシーを考慮した場合でも時間の理論値は 241 μ s となり、本実験での実測値 242 μ s とは同等の結果となっている。したがって、FPGA の処理時間が設計通りになっていることが確認できた。また、本実験のニューロ演算は積和演算器 1 つを使いまわして逐次処理で行っているため、積和演算器を複数実装した並列化をすることで、さらなる高速化を見込むことができる。理論上、リソースの問題を除けば、各層毎を並列に処理することができるため、層の数分のクロック時間で処理できることになる。したがって全結合層が 3 層である場合、理論的には 3 クロックで計算が終了することになるため、レイテンシー時間は 15 ns 程度になると予想できる。

5. おわりに

本稿では、畳み込みニューラルネットワークのハードウェア化を目的とし、パイプライン構造を使用した 2 層の畳み込み層を FPGA で実装と、CNN の全結合層に向けた多層ニューラルネットワークの FPGA 実装を行った。2 層の畳み込み層での検証、各層でのレイテンシーから、さらなる多層化にも対応できるということも確認した。しかし、畳み込みはサイズが小さく、プーリング処理も最も簡単なマックスプーリングの実装にとどまった。そのため、今後の展望としてはより複雑な係数の畳み込み層・プーリング層の実装・検証を行う。また、多層ニューラルネットワークにおいては、FPGA に実装し推論時間が理論通りになることも確認した。今後の展望としては、サイズの大きな CNN の実装や CNN のバリエーションの拡大、また多層ニューラルネットワークの並列化の設計と実装があげられる。

参 考 文 献

- [1] 我妻広明: “人工知能による運転支援・自動運転技術の現状と課題,” 計測と制御, Vol.54, No.11, pp. 808-815, 2015.
- [2] 榎原伸介: “知能ロボットによる工場自動化と IoT, AI 活用について,” システム/制御/情報, Vol.61, No.3, pp. 101-106, 2017.
- [3] 山田武士, 高橋敏, 納谷太, 池邊隆, 古川茂人: “NTT グループにおける AI 研究の取り組みと方向性,” NTT 技術ジャーナル, Vol.28, No.2, pp. 8-13, 2016.
- [4] 上野聡, 高橋恒一, 中田秀基: “特集「AI 計算資源」にあたって,” 人工知能, Vol.33, No.1, pp. 5-7, 2018.
- [5] 黒滝紘生, 松尾豊: “Deep learning 利用法と知見の体系化,” 人工知能学会全国大会論文集第 28 回全国大会, Vol.33, 3H3-OS-24a-4, 2014.
- [6] 関谷翠, 大沢和樹, 長沼大樹, 横田理央: “低ランク近似を用いた深層学習の行列積の高速化,” 情報処理学会研究報告, Vol.2017-HPC-158, No.24, pp. 1-7, 2017.
- [7] 藤井智也, 佐藤真平, 中原啓貴: “2 値化畳み込みニューラルネットワークのニューロン刈りによるメモリ量削減と FPGA 実現について,” 電子情報通信学会技術研究報告, Vol.117, No.221, pp. 25-30, 2017.
- [8] 中原啓貴, 笹尾勤, 岩本久: “Nested RNS の定数除算を用いた深

層畳み込みニューラルネットワークの FPGA 実現について,” 研究報告システム・アーキテクチャ (ARC), Vol.2016-ARC-218, No.39, pp. 1-6, 2016.

- [9] 中村祐一: “ASIC と FPGA どちらを使いますか?,” 電子情報通信学会誌, Vol.96, No.2, pp. 85-89, 2013.
- [10] 中原啓貴: “畳み込みニューラルネットワークの FPGA 実装,” 電子情報通信学会誌, Vol.103, No.5, pp. 501-506, 2020.
- [11] 浜本隆之, 相澤清晴, 長尾祥一, 喜多ちえ: “スマートイメージセンサと FPGA を用いた高速動物体追跡と奥行き推定,” 映像情報メディア学会誌, Vol.57, No.9, pp. 1142-1148, 2003.
- [12] 大明準治: “1,000fps 高速度画像処理のアクティブカメラへの応用,” 日本ロボット学会誌, Vol.23, No.3, pp. 282-285, 2005.
- [13] H. Onuma, Y. Yaguchi, H. Miyaguchi, T. Akiyama, K. Kajiyama, K. Adachi, A. Kikuchi, T. Kojima, and T. Narumi: “SVP: scan-line video processor-general purpose video processor,” 1995 Int. Symp. on VLSI Technology, Systems, and Applications, Proc. of Technical Papers, pp. 196-200, 1995.
- [14] M. Shimasaki and Y. Hoshino: “Design and verification of image processing filters on FPGA,” 16th Int. Symp. on Advanced Intelligent Systems, T5e-2, 2015.
- [15] P. Greisen, S. Heinzle, M. Gross, and A. P. Burg: “An FPGA-based processing pipeline for high-definition stereo video,” EURASIP J. on Image and Video Processing, Article No.18, 2011.
- [16] D. A. Botero Galeano, M. Devy, J. L. Boizard, and W. Filali: “Real-time architecture on FPGA for obstacle detection using inverse perspective mapping,” 18th IEEE Int. Conf. on Electronics, Circuits, and Systems, pp. 788-791, 2011.
- [17] 鈴木宏明, 富永浩文, 佐藤一馬, 中村あすか, 前川仁孝: “GPU を用いた格子ボルツマン法のループ展開を利用したメモリアクセスの局所性向上による高速化,” 情報処理学会第 77 回全国大会講演論文集, pp. 71-72, 2015.
- [18] R. Kamesaka and Y. Hoshino: “Development of a prevention system for beast damage of agricultural product by using deep learning,” 10th Int. Conf. on Soft Intelligent Systems, pp. 747-752, 2018.

(2020 年 10 月 5 日 受付)

(2020 年 10 月 14 日 採録)

[問い合わせ先]

〒782-8502 高知県香美市土佐山田町宮ノ口 185

高知工科大学

星野 孝総

TEL: 0887-53-1111

FAX: 0887-57-2000

E-mail: hoshino.yukinobu@kochi-tech.ac.jp

著 者 紹 介

かめがわ りょうき
亀阪 亮紀 [非会員]

1996 年 10 月 31 日生。2019 年 3 月高知工科大学システム工学群電子・光工学専攻卒業。2019 年 4 月高知工科大学大学院工学研究科基盤工学専攻修士課程入学し現在に至る。

星野 孝総 [正会員]

1970 年 9 月 7 日生。1995 年、米国 Westmar University, Natural Science Division, Department of Computer Science and Mathematics 卒業。1997 年、立命館大学大学院理工学研究科情報学専攻博士前期課程修了。同年、立命館大学大学院理工学研究科総合理工学専攻博士後期課程入学。2001 年 3 月同課程単位取得退学。2002 年博士号（工学）取得（立命館大学）。2002 年立命館大学講師。2007 年高知工科大学工学部電子・光システム工学科助教授。2008 年同大学システム工学群准教授となり現在に至る。ファジィシステム、進化計算、知能ロボティクス、知能システム、知的画像処理、機械学習、感性工学の研究開発に従事。また、近年では知能システム（インテリジェントシステム）のハードウェア化に興味をもち、リアルタイム知的画像処理システムをベースに SISoC（Soft Intelligent System On Chip）の研究／開発に従事。日本知能情報ファジィ学会員、システム制御情報学会員、計測自動制御学会員、電子情報通信学会員、IEEE 会員。

Design of the Convolution Layer Using HDL and Evaluation of Latency Using a Camera Signal

by

Ryoki KAMESAKA and Yukinobu HOSHINO

Abstract:

The Convolutional Neural Network (CNN) is applied to several applications and is expected to the embedded systems such as IoT devices. However, these systems need high calculation costs and high power consumption. Because in general, those system requires the using GPUs and its hard to implement on the small embedded system. In recent years, FPGAs have been applied to the auto-control systems, defect inspection systems, and the security systems. Especially applying for the image processing technologies for industrial products were adapted. Hardware acceleration is one of the techniques to improve processing speed and is often used for image processing. Our work has designed the hardware acceleration for CNN and compared it to software processing. The hardware-based modules of CNN were implemented on FPGA and tested. This paper shows the details architecture, was designed in our research, and the verification results of the real-time processing.

Keywords: convolutional neural network, FPGA, pipeline processing

Contact Address: **Yukinobu HOSHINO**

Kochi University of Technology

185 Miyanokuchi, Tosayamada, Kami city, Kochi 782-8502, Japan

TEL: +81-887-53-1130

FAX: +81-887-57-2000

E-mail: hoshino.yukinobu@kochi-tech.ac.jp