

修士論文

題目

VitisAIを用いた
YOLOv7-tinyのFPGA実装

指導教員

高木 一義 教授

2024 年

三重大学大学院 工学研究科 情報工学専攻
コンピュータアーキテクチャ研究室

古市 諒成 (422M525)

VitisAIを用いた YOLOv7-tiny のFPGA実装

古市 諒成

内容梗概

本研究ではFPGAを用いたYOLOv7-tiny アクセラレータを提案する。開発ツールとしてAMD（旧Xilinx）社が提供するVitis AIを使用した。物体検出はコンピュータビジョンにおける挑戦的なタスクである。YOLOのような深層学習アルゴリズムではGPUが広く使用されているが、消費電力やハードウェアリソースに制限のあるエッジデバイスでの利用には向かない。そこで本研究では、組込みシステムを対象としてFPGAを使用した物体検出システムを提案する。AI推論用の開発プラットフォームであるVitis AIを用いて、YOLOv7-tinyを組み込み開発ボードのUltra96v1に実装した。Vitis AIを用いる利点は、機械学習モデルのネットワークの深さに依存しない一定のリソース消費量である点や高度に抽象化されたハードウェア設計で開発期間を短縮することができる点である。

Vitis AIで提供されるDPU（Deep Learning Processor Unit）を使用して畳み込み演算の高速化を図った。また、モデルはVitis AIクオンタイザを用いて8ビットの整数型に量子化した。DPUはSoCデバイスのプログラマブルロジック（PL）内に1つのブロックとして統合され、プロセッシングシステム（PS）に直接接続可能である。物体検出の前処理と後処理をPSで行い、推論部分をPLのDPUで処理する。前処理として、画像のロードとリサイズを行う。そして、リサイズされた画像を入力としてDPUで推論処理を行う。後処理では、DPUの出力の整形やシグモイド関数による正規化、非最大値抑制処理（NMS）を行う。設計したシステムは5000枚のテスト画像による評価でスループットは7.6FPS、平均消費電力は6.18Wであった。

FPGA Implementation of YOLOv7-tiny Using Vitis AI

Ryosei Furuichi

Abstract

In this study, we propose an FPGA-based YOLOv7-tiny accelerator. We used Vitis AI provided by AMD (formerly Xilinx) as a development tool. Object detection is a challenging task in computer vision. Although GPUs are widely used in deep learning algorithms such as YOLO, they are not suitable for edge devices with limited power consumption and hardware resources. Therefore, we propose an FPGA-based object detection system for embedded systems. We implemented YOLOv7-tiny on an embedded development board Ultra96v1 using Vitis AI, a development platform for AI inference. The advantages of using Vitis AI are constant resource consumption independent of the depth of the machine learning model network, and a highly abstracted hardware design to shorten the development time.

The Deep Learning Processor Unit (DPU) provided by Vitis AI is used to accelerate convolutional operations. The models were quantized to 8-bit integers using the Vitis AI quantizer, and the DPU was integrated as a single block in the SoC device's programmable logic (PL), which can be directly connected to the processing system (PS). The PS performs the pre-processing and post-processing of object detection, while the inference part is processed by the DPU in the PL. Image loading and resizing are performed as pre-processing. The resized images are then used as input for inference processing in the DPU. Post-processing includes shaping the DPU output, normalization with a sigmoid function, and non-maximum suppression (NMS). The designed system had a throughput of 7.6 FPS and an average power consumption of 6.18 W when evaluated on 5000 test images.

目 次

1	緒論	1
2	準備	3
2.1	物体検出	3
2.2	YOLOv7-tiny	4
2.3	プログラマブル SoC	6
2.4	Vitis AI	7
2.4.1	概要	7
2.4.2	DPU	8
3	関連研究	11
4	物体検出システムの構成	13
5	実装と評価	17
5.1	開発フロー	17
5.2	開発環境	18
5.3	評価方法	18
5.4	評価結果	22
5.5	考察	26
6	結論	28
	謝辞	29
	参考文献	30

図 目 次

2.1	YOLOv7-tiny アーキテクチャ	6
2.2	Vitis AI ワークフロー	8
2.3	DPUCZDX8G アーキテクチャ	10
4.4	設計したシステムの概要	13
4.5	データフロー図	14
4.6	DPUCZDX8G アーキテクチャの詳細 [29]	15
5.7	Avnet 製 Ultra96v1 ボード	19
5.8	Precision Recall curve[34]	22
5.9	B1600 での消費電力	24
5.10	B2304 での消費電力	25
5.11	アプリケーション非動作時の消費電力	25
5.12	マルチスレッドによる処理の流れ	27

表 目 次

2.1	DPUCZDX8G アーキテクチャ別のリソース量	9
5.2	DPU パラメータ	20
5.3	ハードウェアリソース使用量	20
5.4	実行時間の内訳と FPS	23
5.5	精度測定結果	23

1 緒論

物体検出は、顔認識 [1],[2],[3], 歩行者検出,[4],[5],[6], 農作物検出 [7],[8],[9] などの多くのアプリケーションで広く利用されている。近年、深層学習の発展に伴い、物体検出の技術は急速に発展してきている。一般に物体検出は GPU (Graphics Processing Unit) プラットフォームで実装されてきた。しかし、GPU は消費電力が大きく、物理的なスペースを必要とするため、リソースに制約のあるエッジデバイスでの利用には向いていない。また、物体検出アルゴリズムの設計のためには、継続的な反復改善が必要であるため、設計の柔軟性が重要である。そこで、カスタマイズ可能な構造と並列性を備えている FPGA を物体検出のためのプラットフォームとして採用する研究が増えている。実際、「FPGA」と「物体検出」をキーワードとする論文数は過去 10 年増加し続けている [10]。

FPGA アクセラレータの設計アプローチは広く研究されている。ハードウェア記述言語を使用してレジスタトランスファーレベル (RTL) 設計を作成する方法や命令形言語 (C や C++) を使用して高位合成 (HLS) を利用する方法がある。加えて、最近では Vitis AI[11] が登場し、より高度で抽象化されたアクセラレーションフレームワークが利用可能となった。これにより、RTL 設計やハードウェア記述言語の高度な専門知識が

不必要となり，開発期間を短縮することができる．また，Vitis AI を用いるとネットワークの深さに依存しない一定のリソース消費量でハードウェア実装が可能である．そのため，高位合成などではハードウェアリソースの制限により実装が困難なモデルを実装することができる．本研究では，物体検出アルゴリズムの一つである YOLOv7-tiny を組み込み開発ボードの Ultra96v1 に実装し，実行速度，精度，消費電力を評価した．

以下，第 2 章では，物体検出，AMD（旧 Xilinx）の提供する AI 推論アクセラレータ環境について述べる．第 3 章では，Vitis AI を用いた物体検出の関連研究を紹介する．第 4 章で本研究で作成したシステム構成について説明する．第 5 章で評価結果と今後の改良手法を示す．最後に第 6 章で結論を述べる．

2 準備

2.1 物体検出

物体検出とは、画像を入力として物体の位置とクラスを出力とするタスクである。深層学習モデルに基づく物体検出アルゴリズムは2つのカテゴリに分類することができる。1つは、R-CNN[12], Fast R-CNN[13], Faster R-CNN[14] などの2ステージ法である。2ステージ法は、物体の候補領域を抽出し、その後分類を行う。もう1つは、YOLO[15] や SSD[16] などの1ステージ法である。1ステージ法は、特徴抽出から検出までを単一の処理で行う。一般に1ステージ法は2ステージ法に比べ、精度が低い、処理速度が速いという特徴を持つ。そのため、1ステージ法はより高いリアルタイム性が要求される条件下での適用に適している。長年にわたり、YOLO アルゴリズムの最初のバージョン (YOLOv1) に対するいくつかの改良が提案されてきた。YOLOv1 は、Redmon らによって提案された初めての1ステージ法の物体検出アルゴリズムである。YOLOv2[17] は、YOLOv1 をベースにして、全結合層を削除し、アンカーボックスを導入した。また、新しいネットワーク Darknet-19 が提案された。YOLOv3[18] は、Redmon らによって提案された最後のバージョンである。Feature Pyramid Network (FPN) [19] を導入し、3つの異なるスケールでボック

スを予測する。これにより、画像内の物体のサイズが異なる場合も対応が可能とした。YOLOv4[20] は Alexey らによって提案された。物体検出の構成要素には、特徴抽出を行うバックボーン、検出を行うヘッド、バックボーンとヘッドの間で特徴マップを結合するネックがある。バックボーンに計算量を大幅に削減可能な Cross Stage Partial Network (CSP) [21] を取り入れた CSPDarknet-53 を提案した。ヘッドは YOLOv3 と同等のものを、ネックには Spatial Pyramid Pooling (SPP) [22], Path Aggregation Network (PAN) [23] を採用した。その後も、2021 年 8 月には、Zheng らによってアンカーフリーモデルの YOLOX[24] が、2022 年 7 月には、Wang らによって YOLOv7[25] が発表された。CSPDarknet を発展させて分岐を増やした Extended Efficient Layer Aggregation Networks (ELAN) モジュールを取り入れた。ELAN モジュールは複数の畳み込みモジュールで構成され、学習と収束の向上を促進するために最短および最長の勾配パスを調整する。

2.2 YOLOv7-tiny

YOLOv7-tiny は YOLOv7 をベースにした軽量の YOLO シリーズの 1 つである。YOLOv7 のパラメータ数は 3760 万に対して、tiny モデルは

620 万である。軽量なモデルは、検出精度が低くなるが、ネットワーク構造が比較的単純であり、検出速度が速い。そのため、エッジデバイスに適している。図 2.1 に 80 種類の物体を検出する場合の YOLOv7-tiny アーキテクチャを示す。YOLOv7 に比べ、ELAN モジュールは層数が削減され、簡略化されている。さらに計算を簡単にするために活性化関数は、YOLOv7 で使用される SiLU 関数を使用せず、LeakyReLU 関数を使用する。バックボーンには、いくつかの CBL ブロック、ELAN モジュール、および max-pooling 層で構成される。CBL には、畳み込み層、バッチ正規化層、LeakyReLU 活性化関数が含まれており、さまざまなスケールでの画像特徴の抽出を容易にする。ネックにはパスアグリゲーションフィーチャーピラミッドネットワーク（PAFPN）構造が採用されている。これは、空間ピラミッドプーリングおよび畳み込み空間ピラミッドプーリング（SPPCSPC）構造、ELAN の組み込みとともに、いくつかの CBL ブロックで構成される。SPPCSPC 構造は、空間ピラミッドプーリング（SPP）構造内に畳み込み空間ピラミッド（CSP）構造を統合し、特徴抽出を最適化する大きな残差エッジを統合することで、モデルの受容領域を拡張する。ネックは、深い特徴マップのアップサンプリングを浅い特徴マップに統合して浅いネットワークの意味情報を強化する一方、浅い

アに振り分けて実行できる組込みシステムの設計が可能となる。プログラマブル SoC によって、低消費電力、低コスト、基板の小型化を実現できる。FPGA とは、内部の論理回路を再構成可能な半導体デバイスである。FPGA は CPU や GPU に比べて低消費電力であり、ハードウェア的にプログラム可能であるという利点がある。また、特定用途向け集積回路 (ASIC) と比較して、設計の柔軟性が高く、開発コストが低いという利点がある。

2.4 Vitis AI

2.4.1 概要

Vitis AI は、AMD (旧 Xilinx) 社製のハードウェアプラットフォーム上で AI 推論を実現するための開発プラットフォームである。Vitis AI は図 2.2 に示す構成となっている。これには、様々な最適化済みモデルを提供する Model Zoo やプルーニングを行う AI オプティマイザ、量子化を行う AI クオнтаイザが含まれる。また、高い効率性と使いやすさを考えて設計されており、AMD (旧 Xilinx) 製 FPGA での AI アクセラレーションや深層学習の性能を最大限に引き出すことができる。Vitis AI 開発環境により、FPGA の複雑さが抽象化されるため、FPGA の設計経験が少ないユーザでも容易に AI 推論アプリケーションを開発できる。また、ハー

ドウェア記述言語による RTL 設計や C, C++による高位合成などの専門知識が不要となり，システムの開発時間を短縮することもできる．

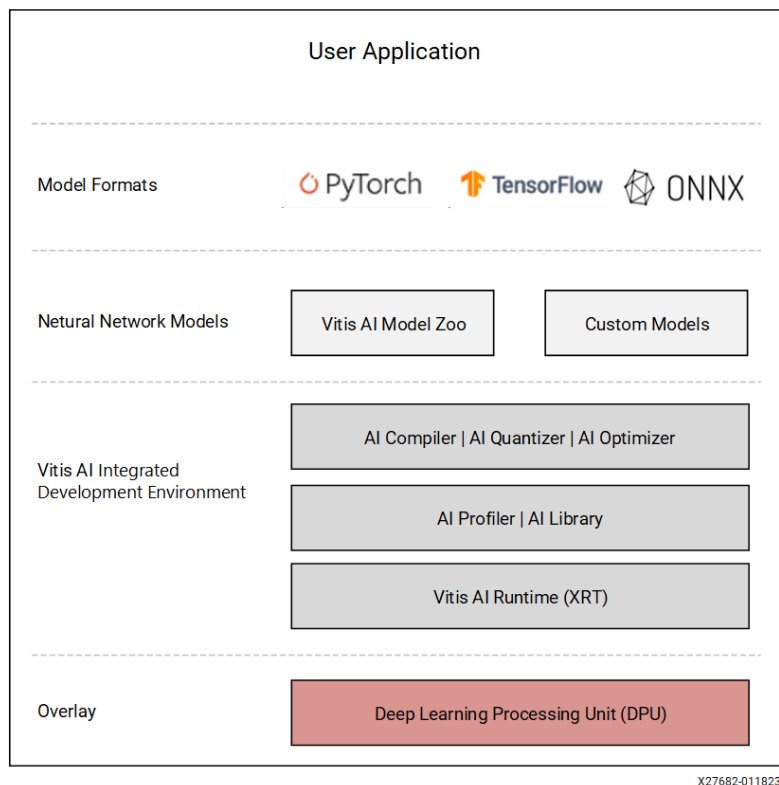


図 2.2: Vitis AI ワークフロー

2.4.2 DPU

DPU (Deep Learning Processor Unit) は，ディープニューラルネットワーク向けに最適化されたプログラマブルなエンジンである．既にハードウェアに実装されたパラメータ指定可能な IP コアであり，配置・配線を実行する必要がない．DPU は特殊な命令セットを使用することで，畳み

込みニューラルネットワークの演算を高速に実行できるように設計されている。本研究で使用した DPUCZDX8G アーキテクチャの概要を図 2.3 に示す。DPU は Vitis AI コンパイラによって生成されたマイクロコードを実行し、入力データの読み取りや結果や一時データの保存のために DDR4 などの高速メモリへのアクセスを必要とする。DPU IP はデバイスのプログラマブルロジック (PL) にブロックとして統合し、プロセッシングシステム (PS) に直接接続できる。この DPU は複数のサイズが用意されており、様々なアプリケーションやハードウェアのリソース要件に対応している。表 2.1 に ZCU102 プラットフォームでの DPUCZDX8G シングルコアのリソース使用状況のサンプルを示す。

表 2.1: DPUCZDX8G アーキテクチャ別のリソース量

DPUCZDX8G アーキテクチャ	LUT	レジスタ	ブロック RAM	DSP
B512	26922	34543	72	118
B800	29721	41147	90	166
B1024	34074	48057	104	230
B1152	32169	47374	121	222
B1600	38418	58831	126	326
B2304	42127	68829	165	438
B3136	46714	79710	208	566
B4096	52161	98249	255	710

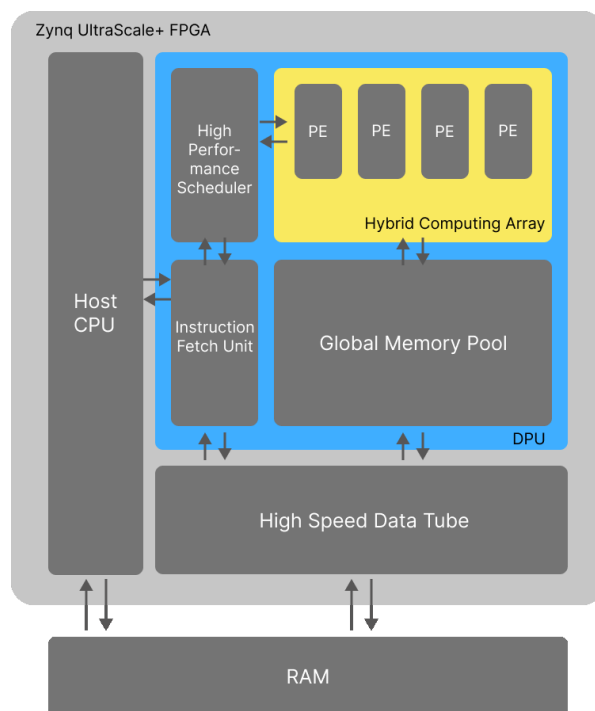


図 2.3: DPUCZDX8G アーキテクチャ

3 関連研究

Wang らは ARM+FPGA アーキテクチャに基づく YOLOv3 アクセラレータを提案した [26]. YOLOv3 モデルは Vitis AI ツールを用いて量子化及びプルーニングされ, Xilinx Zynq ZCU104 ボードに実装された. この研究では PASCALVOC データセットでテストされ, スループットを表す FPS (Frame Per Second) は 84.56, 消費電力は 25W であった. GPU の GeForce GTX1080 と比較して, より高いスループット, 電力効率を達成した. Chen らは道路凹凸検出システムのために Vitis AI を用いて YOLOv4 モデルを Xilinx Zynq ZCU104 ボードに実装した [27]. 88.8% の認識精度と 28FPS を達成し, リアルタイム検出を実現した. Li らは光学式リモートセンシング画像の物体検出システムのために RFA-YOLO モデルを提案し, Xilinx ZCU104 ボードに DPU ベースのハードウェア実装を行った [28]. FPS は 27.97, 平均消費電力は 15.82W であった. CPU の Intel i7-6700, GPU の Tesla V100 と比較して, 検出速度の向上と低消費電力を達成し, オンボードでのリアルタイム検出要件を満たした.

本研究では, より省リソース・低消費電力な実装を目指すために DPU を搭載可能な SoCFPGA のうち比較的小型なものを使用する. また, それに伴って YOLO の軽量バージョンのモデルを使用する. そして, 本研

究では特定のタスクに絞らず，80 クラスの一般物体検出を行う．

4 物体検出システムの構成

本研究で設計したシステムの概要を図 4.4 に示す．物体検出の処理には大きく分けて，前処理，推論処理，後処理の3つがある．DPU の推論処理を PL の FPGA に委譲し，前処理と後処理を PS で行う．図 4.5 にこの流れを示す．まず，前処理として画像をメモリにロードする．その後，DPU 入力用にレターボックス処理によって，アスペクト比を保って， 416×416 ピクセルにリサイズする．

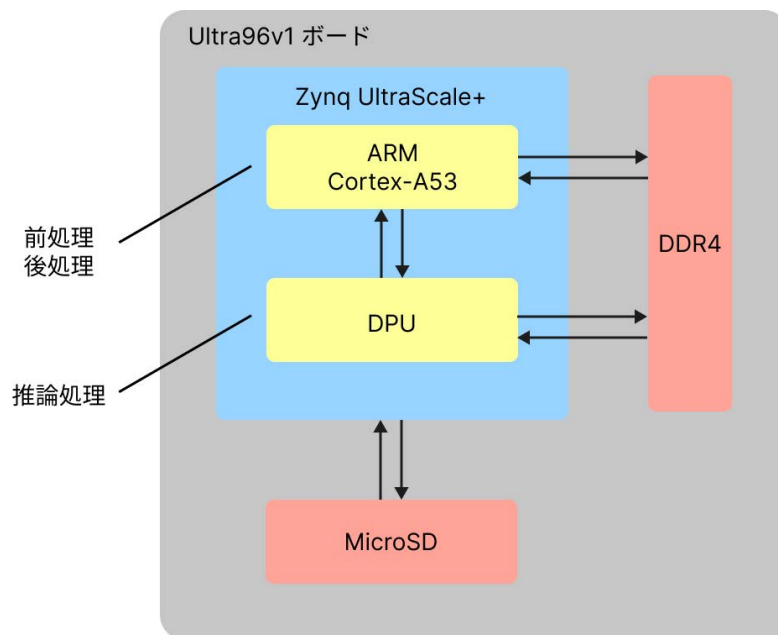


図 4.4: 設計したシステムの概要

レターボックス処理では， 416×416 ピクセルの RGB 値各 128 のグレーの画像を作成する．その後，元画像の長辺が 416 ピクセルになるように

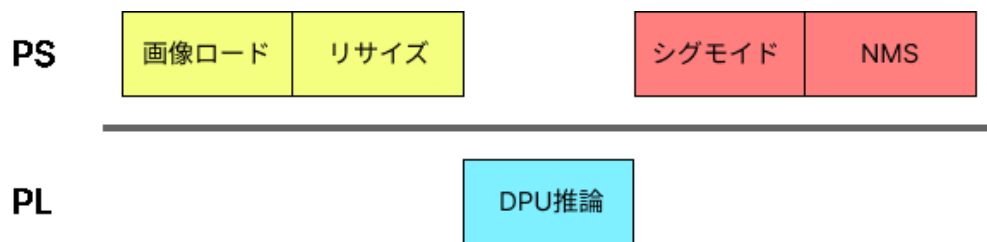


図 4.5: データフロー図

リサイズし，グレーの画像の中央に重ねて描画する．

前処理の完了後，DPU によって推論処理を行う．図 4.6 に使用した DPU アーキテクチャの詳細を示す．DPU はオフチップメモリから命令をフェッチし，演算エンジンの動作を制御する．命令は，レイヤー融合など高度な最適化機能を持つ Vitis AI コンパイラによって生成される．オンチップメモリを中間層特徴マップや出力メタデータのバッファとして使用することで，高スループットと高い効率性を実現する．また，外部メモリの帯域幅要件を軽減するため，なるべく多くのデータを再利用するようになっている．DPU は，リサイズされた画像を入力として推論結果を 3 つのスケールで出力する．出力のスケールは $13 \times 13 \times 255$ ， $26 \times 26 \times 255$ ， $52 \times 52 \times 255$ である．これには，予測したバウンディングボックスの座標値や物体，クラスごとの確信度が含まれている．

後処理は，DPU の出力の特徴マップの整形，スコアや座標値の演算，シグモイド関数による正規化，NMS（Non-Maximum Suppression）処理

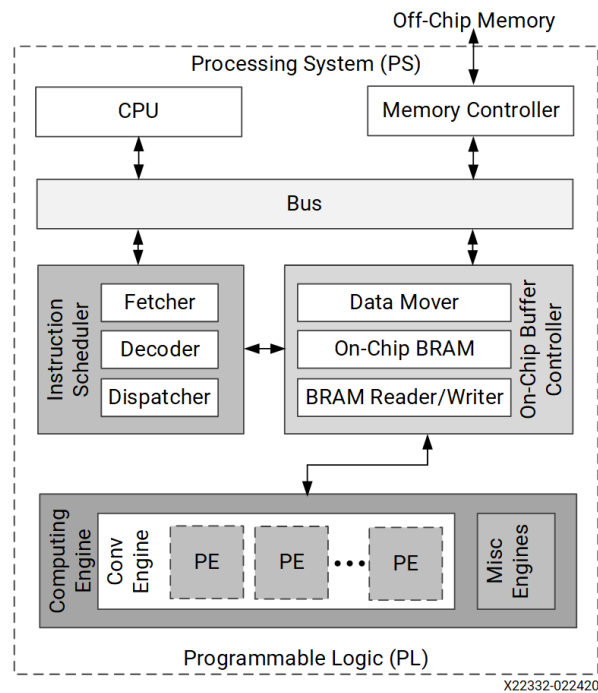


図 4.6: DPUCZDX8G アーキテクチャの詳細 [29]

がある。

NMS 処理は、同一の物体に対して検出された複数のバウンディングボックスのうち最もスコアの高いものを残し、それ以外を除去する処理である。バウンディングボックスとは、物体の位置を囲う矩形のことである。ここでは、式 (1) で表される IoU (Intersection over Union) を閾値として用いる。IoU は 2 つのバウンディングボックスがどれだけ重複しているかを表す指標であり、1.0 に近づくほど重複している。この式では、Area of Intersection は 2 つのバウンディングボックスの重複する部分の面積を表し、Area of Union は 2 つのバウンディングボックス全体の

合計面積を表している.

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}} \quad (1)$$

NMS 処理の具体的な手順は以下の通りである.

1. 最もスコアの高いバウンディングボックスを選択する.
2. 他のすべてのバウンディングボックスとの IoU を計算し, 閾値を超えるものを除去する.
3. 最もスコアの高いバウンディングボックスを除いて, 手順1に戻る.
4. プロセスを繰り返し, 最終的な検出結果を得る.

5 実装と評価

5.1 開発フロー

Vitis AI で TensorFlow 形式のモデルを入力として、量子化を行った後 FPGA 用のモデルファイルを作成し、物体検出アプリケーションは Jupyter Notebook で Python 言語で開発を行った。本研究では、YOLOv7-tiny の学習済み重みファイルを使用した。PyTorch 形式の学習済み重みを使用し Vitis AI コンパイラの入力とすると、サポートされていない演算が含まれていた。そこで、Darknet 形式の学習済み重みから TensorFlow 形式に変換する手順を取った。Darknet 形式の重みは Vitis AI でサポートされていないので、サードパーティ製のスクリプト [30] を用いて、Darknet 形式の重みを Keras モデルに変換した。その後、フリーズされた TensorFlow モデルに変換した。Vitis AI の AI クオンタイザを用いて、32 ビットの浮動小数点の重みを 8 ビットの整数に変換した。Vitis AI クオンタイザでは、4 ビットや 16 ビットなど様々なビット幅のモデルを作成することができる。だが、現状ではそれらのビット幅は DPU ハードウェアでは対応しておらず、シミュレーションにのみ使用することができる。量子化後、AI コンパイラによりニューラルネットワーク計算が最適化され DPU での実行に適した形式の機械学習モデルを含む.xmodel ファイルが生成さ

れる.

5.2 開発環境

本研究では, ARM Cortex-A53 CPU, Xilinx Zynq UltraScale+ MPSoC ZU3EG を集積したプログラマブル SoC を搭載する Avnet 製 Ultra96v1 を使用した. 図 5.7 に Ultra96v1 ボードを示す. AMD (旧 Xilinx) のオープンソースプロジェクトである PYNQ[31] を使用することで, Jupyter Notebook 上で Python 言語で開発を行った. DPU の処理部分は, AMD (旧 Xilinx) 社が提供する VART プログラミング API[32] を使用した. DPU はコンフィグファイルの書き換えにより, パラメータを調整することができる. 使用した DPU のパラメータを表 5.2 に示す. また, DPU は表 2.1 に示したもののうち, Ultra96v1 に組み込み可能な B1600 と B2304 を使用した. 表 5.3 に Ultra96v1 のハードウェアリソースの使用量を示す. また, DPU の動作周波数は 400MHz である.

5.3 評価方法

YOLOv7-tiny は Microsoft Common Objects in Context (MSCOCO) データセット [33] で学習されており, 5000 枚の画像をテストとして使用した. MSCOCO は人間, 乗り物, 動物, 家具など 80 種類のクラスを含



図 5.7: Avnet 製 Ultra96v1 ボード

む画像データセットである。

評価指標としてスループットを表す Frame Per Second (FPS), 精度を表す mean Average Precision (mAP), 平均消費電力を使用した. FPS は単位時間あたりに検出される画像の和を表し, 値が大きいほど検出速度が速いことを示す. mAP は各クラスの Average Precision (AP) の平均である. これを計算するために, 4 で説明した IOU と Precision, Recall が必要となる. なお本研究では, Wang らの YOLOv7 のコードに則り, 精度測定時の検出スコアの閾値を 0.001, NMS 処理に用いる IOU の閾値を 0.65 と設定し mAP の最大化を図った.

精度測定において, IOU は, 予測したバウンディングボックスと正解

表 5.2: DPU パラメータ

ULAM	DISABLE
RAM_USAGE	LOW
CHANNEL_AUGMENTATION	DISABLE
ALU_PARALLEL	DEFAULT
CONV_RELU	LEALYRELU_RELU6
ALU_RELU	LEAKYRELU_RELU6
DSP48_USAGE	LOW
LOWPOWER	DISABLE

表 5.3: ハードウェアリソース使用量

	B1600	B2304	Available	B1600 での使用率 (%)	B2304 での使用率 (%)
LUT	38092	43193	70560	53.99	61.21
FF	63146	74437	141120	44.75	52.75
BRAM	126	165	216	58.33	76.39
DSP	246	342	360	68.33	95.00

のバウンディングボックスがどれだけ重なっているかを表す。

Precision は、予測したバウンディングボックスの正解率を表す。IOU の閾値を 0.5 とすると $IOU \geq 0.5$ である予測数を True Positive (TP), $IOU < 0.5$ である予測数を False Positive (FP) と呼ぶ。Recall は、検出すべきバウンディングボックスを検出できている比率を表す。検出されなかったバウンディングボックスの数を False Negative (FN) と呼ぶ。

Precision と Recall は以下の式で表せる。

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

AP とは、Precision と Recall を組み合わせた指標である。図 5.8 は縦軸に Precision、横軸に Recall をプロットした Precision Recall curve の例である。この曲線下の面積が AP であるが、グラフは Recall 値を Precision に対し単調非増加にするため近似して計算を行う。そこで、AP の計算のため赤い破線のように Precision は、各 Recall 値に対して、それ以上の Recall 値での最大の Precision に置換する。そして、MSCOCO では $[0.00, 0.01, \dots, 0.99, 1.00]$ の 101 点の Recall における Precision を平均することで AP を求める。このとき、AP は以下の式で表せる。

$$AP = \frac{1}{101} \sum_{R \in \{0.00, 0.01, \dots, 0.99, 1.00\}} P(R)$$

mAP は全てのクラスに対する平均精度なので、次のように表せる。

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

ここで、 AP_i は i 番目のクラスの AP、 N はクラスの総数である。

実行時間の測定区間は、画像をメモリにロードするところから後処理の NMS 処理が完了するまでである。2 種類の DPU ごとにそれぞれ測定を 5 回行い、その平均値を使用した。消費電力の測定には Ultra96v1 に組み込まれている電源管理バスである PMBus を使用した。PYNQ にお

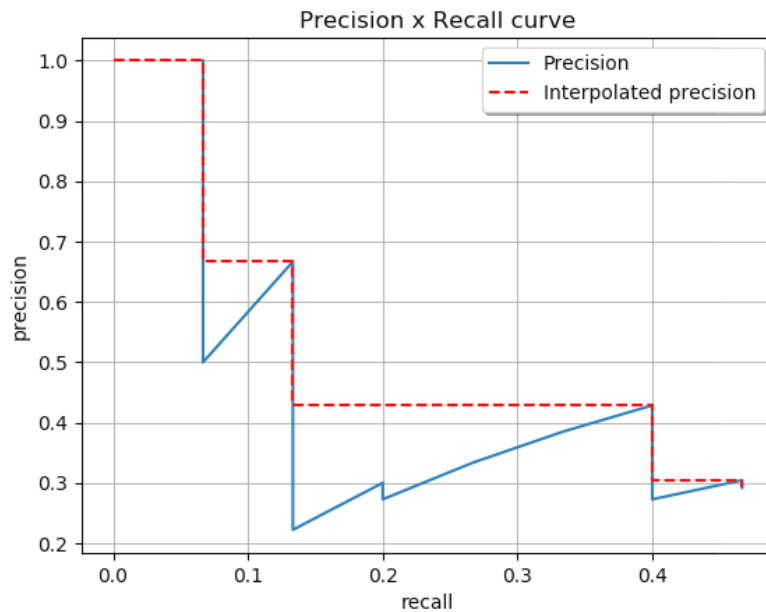


図 5.8: Precision Recall curve[34]

いて PMBus を制御するためのクラスが提供されており、それを用いて物体検出アプリケーション動作中のシステム全体の消費電力とアプリケーション非動作時の消費電力を測定した。

5.4 評価結果

テスト画像 5000 枚の実行時間の内訳と FPS を表 5.4 に示す。検出スコアの閾値を 0.25, IOU の閾値を 0.45 で評価を行った。DPU の性能差により、B2304 使用時の推論時間は 35 秒短くなっている。しかし、前処理と後処理が全体に対して占める割合が大きいため、FPS の差は軽微なものとなっている。

表 5.4: 実行時間の内訳と FPS

DPU	前処理 (s)	推論処理 (s)	後処理 (s)	FPS
B1600	202	258	220	7.2
B2304	202	223	222	7.6

表 5.5 に整数 32 ビットの YOLOv7-tiny モデルと本研究で使した 8 ビットのモデルの精度を示す。ここで、mAP50, mAP75 は IOU の閾値が 0.5, 0.75 のときの mAP である。また、mAP は $[0.5, 0.55, \dots, 0.95]$ の 10 個の異なる IOU 閾値で mAP を計算し、それらを平均したものである。mAPS, mAPM, mAPL は検出すべき物体の大きさごとの精度である。mAPS は物体面積が 32^2 ピクセル未満、mAPM は 32^2 から 96^2 の間、mAPL は 96^2 より大きい物体である。本研究のモデルの mAP は 6.4% 低下している。また、物体の大きさが大きくなると精度低下も大きくなっている。

表 5.5: 精度測定結果

	mAP	mAP50	mAP75	mAPS	mAPM	mAPL
元モデル (FP32)	0.333	0.499	0.353	0.121	0.361	0.544
本研究 (INT8)	0.269	0.460	0.288	0.112	0.294	0.416

図 5.9, 図 5.10, 図 5.11 に B2304 と B1600 での実行時の消費電力と非動作時の消費電力を示す。図の赤い破線は測定時間における平均電力を表す。図 5.9 と図 5.10 から DPU の違いにより、0.24W の差が確認できる。

また、ピーク電力も 0.5W 程度差があることがわかる。また、アプリケーション動作時と非動作時の差は 1.59W から 1.83W であった。

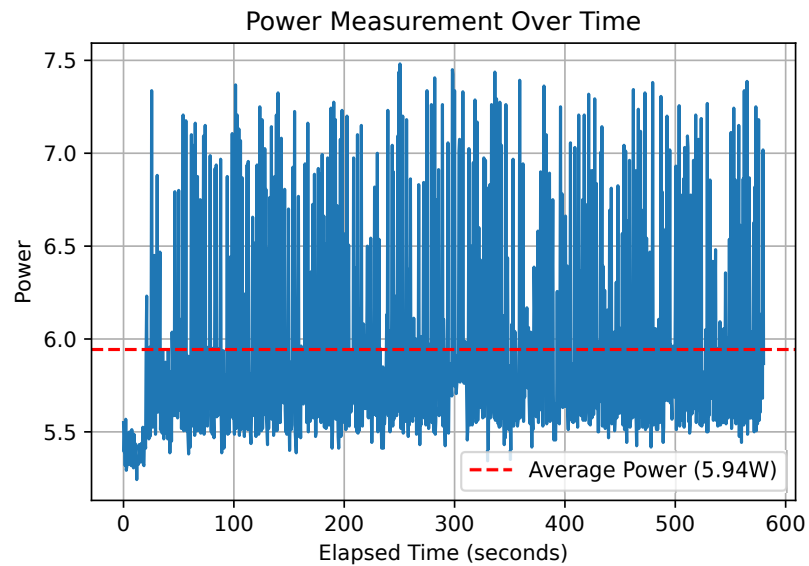


図 5.9: B1600 での消費電力

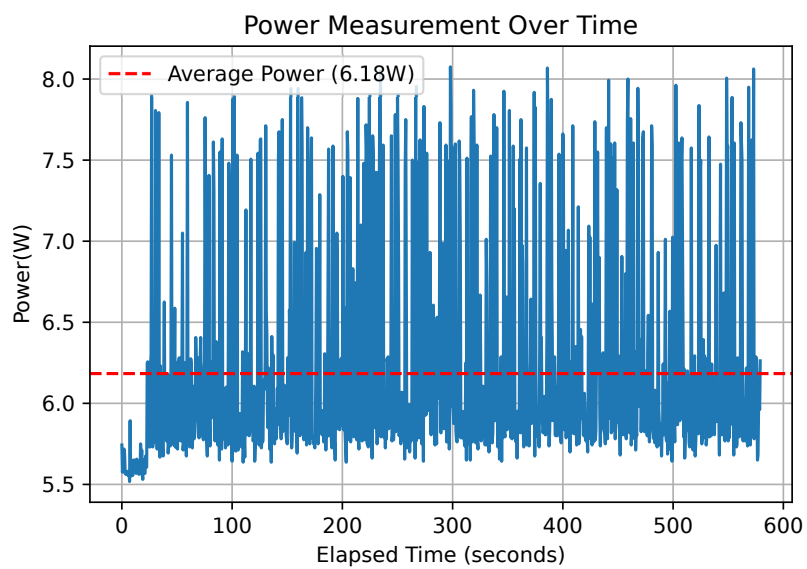


図 5.10: B2304 での消費電力

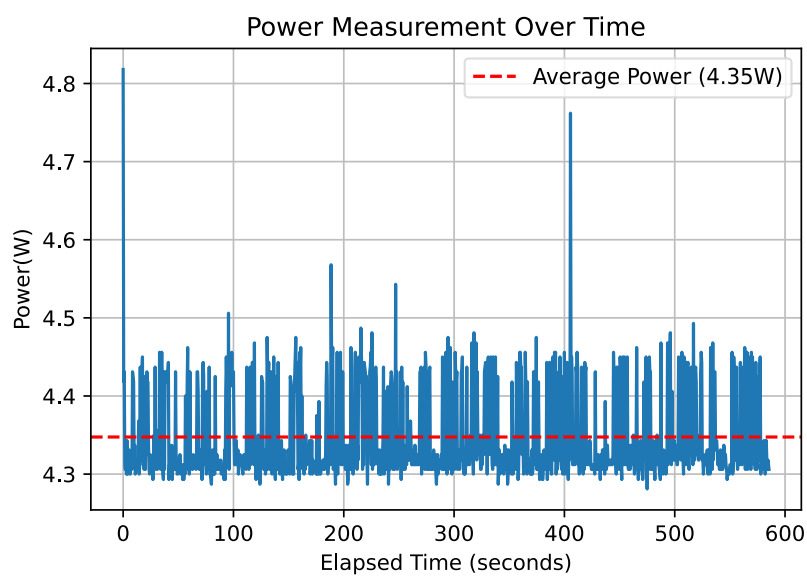


図 5.11: アプリケーション非動作時の消費電力

5.5 考察

YOLOv7-tiny モデルは正常に動作し、エッジデバイスでの推論が行えている。だが、現状はスループットが7.6FPS、精度は26.9%であり、まだ改善の余地がある。まず、実行速度の改善について述べる。現在、実行速度はPSが担当するソフトウェア部分がボトルネックとなっている。[35],[37]の研究からも、低速なCPUによる前後処理がクリティカルパスになり得ることが示されている。そのため、並列化可能な処理を検討し前後処理の効率化を図る必要がある。図 5.12 に示すように、マルチスレッド処理を行うことでシングルスレッド処理のときと比較して高速化できる可能性がある。番号は処理する画像の番号である。表 5.4 から、DPU に B2304 を使用するときの画像 1 枚あたりの処理時間は、前処理が 40.4 ミリ秒、推論処理が 44.6 ミリ秒、後処理が 44.4 ミリ秒とわかる。5000 枚の画像をマルチスレッドで処理するときの実行時間は以下ようになる。

$$40.4 + 2500(44.6 \times 2 + 44.4) = 334,040.4 \text{ ms} = 334.0404 \text{ s}$$

マルチスレッドによる処理はシングルスレッドと比較して、48%の高速化が期待できる。また、コードをC,C++に書き換えることやリファクタリングで処理方法を見直すことで前後処理の実行時間を短縮することがで

きると考えられる。

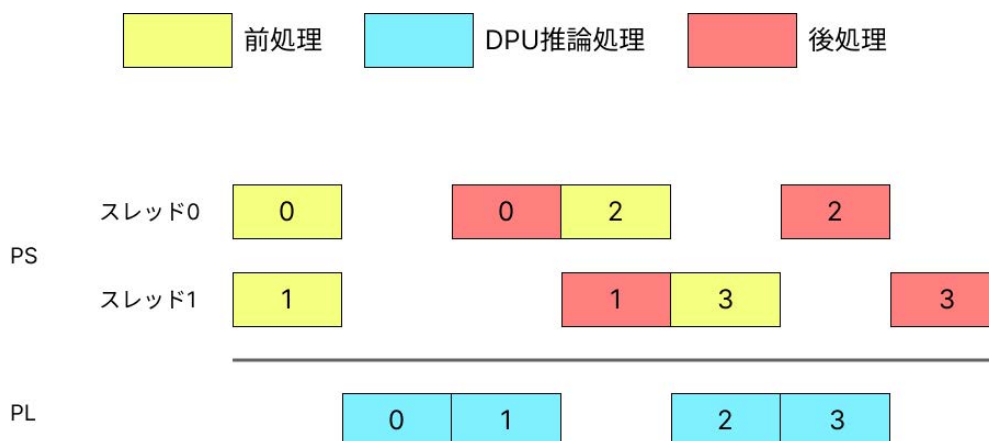


図 5.12: マルチスレッドによる処理の流れ

次に、精度低下の改善について述べる。Vitis AI クオンタイザはほとんど精度低下無しに量子化することができる。[35]の研究では、量子化による精度低下は 2.2%、[36]の研究では 1.86%の低下であった。本研究での精度低下はこれらよりも大きい。精度低下を抑えるために、モデルの変換手順の確認や量子化の調整が必要であると考え。本研究ではモデルの形式を Darknet 形式から TensorFlow 形式に変換した後、量子化を行っている。この変換時に精度低下が起きている可能性があるため確認が必要である。Vitis AI クオンタイザでは重みと量子化パラメータをレイヤーごとに調整する高速微調整機能や性能向上のための量子化ストラテジを設定できる。これらを利用し、精度を確認しながら調整することも有効であると考えられる。

6 結論

本研究では YOLOv7-tiny を Vitis AI を用いることで FPGA に実装し、高速化の可能性を提案した。Vitis AI 開発環境により容易に量子化を行い、DPU でハードウェア設計を抽象化して、AI 推論アプリケーションを動作させることができた。設計したシステムは、5000 枚のテスト画像による評価でスループットは 7.6FPS、平均消費電力は 6.18W であった。

今後の課題として、マルチスレッド処理、DPU 推論処理の並列化による高速化や量子化後の精度低下を抑えることが挙げられる。ソフトウェア部分の前後処理がボトルネックである。そのため、前後処理を推論処理の裏で行うように並列化すると効果的であると考えられる。実際には、スレッド間のデータ受け渡しやオーバーヘッドを考慮する必要がある。また、C/C++ にコードを書き換えることやリファクタリングによって、処理手順を変更せずとも処理時間を短くすることができると考えられる。量子化による精度低下を抑えるためには、モデルの変換手順を確認することや Vitis AI クオンタイザーの高度な量子化調整機能を用いて精度を確認すると有効であると考えられる。

謝辞

本研究を進めるにあたり，日頃よりご指導やご助言いただいた高木一義教授，大野和彦講師，並びに日々の研究活動のサポートもしていただいた深澤祐樹技術員に深く感謝いたします．そのほか日々の生活でお世話になりましたコンピュータアーキテクチャ研究室の皆様にも心からお礼申し上げます．

参考文献

- [1] Garg, D., Goel, P., Pandya, S., Ganatra, A., and Kotecha, K. “A deep learning approach for face detection using YOLO,” 2018 IEEE Punecon. IEEE, (2018): 1-4.
- [2] Menon, S., Geroje, A., Aswathy, N., and James, J., “Custom Face Recognition Using YOLO. V3,” 2021 3rd International Conference on Signal Processing and Communication (ICPSC). IEEE, (2021): 454-458.
- [3] Ghimire, A., Werghi, N., Javed, S., and Dias, J, “Real-Time Face Recognition System,” arXiv preprint arXiv:2204.08978, 2022.
- [4] Li, Guilan, Jie Yang, and Zhuang Kang, “Pedestrian detection algorithm based on improved yolov3_tiny,” Proceedings of 2021 Chinese intelligent automation conference. Springer Singapore, (2022): 98-106.
- [5] Kuramochi, R., Shimoda, M., Sada, Y., Sato, S., and Nakahara, “Fpga-based accurate pedestrian detection with thermal camera for

- surveillance system,” 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, (2019): 1-5.
- [6] Li, Wei, “Infrared image pedestrian detection via YOLO-V3,” 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). IEEE, (2021): 1052-1055
- [7] Horng, Gwo-Jiun, Min-Xiang Liu, and Chao-Chun Chen, “The smart image recognition mechanism for crop harvesting system in intelligent agriculture,” IEEE Sensors Journal 20.5 (2019): 2766-2781.
- [8] Bazame, H. C., Molin, J. P., Althoff, D., and Martello, M, “Detection, classification, and mapping of coffee fruits during harvest with computer vision,” Computers and Electronics in Agriculture 183 (2021): 106066.
- [9] Zhang, J., Karkee, M., Zhang, Q., Zhang, X., Yaqoob, M., Fu, L., and Wang, S, “Multi-class object detection using faster R-CNN and estimation of shaking locations for automated shake-and-catch apple harvesting,” Computers and Electronics in Agriculture 173 (2020): 105384.

- [10] Zeng, K., Ma, Q., Wu, J. W., Chen, Z., Shen, T., and Yan, C, “FPGA-based accelerator for object detection: A comprehensive survey,” *The Journal of Supercomputing* 78.12 (2022): 14096-14136.
- [11] AdvancedMicroDevicesInc,<https://japan.xilinx.com/products/design-tools/vitis/vitis-ai.html>,(2024.01.11).
- [12] Girshick, R., Donahue, J., Darrell, T., and Malik, J, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2014): 580-587.
- [13] Girshick, R, “Fast r-cnn,” *Proceedings of the IEEE international conference on computer vision*, (2015): 1440-1448.
- [14] Ren, S., He, K., Girshick, R., and Sun, J, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, (2015): 28.
- [15] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A, “You only look once: Unified, real-time object detection,” *Proceedings of the*

IEEE conference on computer vision and pattern recognition, (2016):
779-788.

- [16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C, “Ssd: Single shot multibox detector,” Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11 – 14, 2016, Proceedings, Part I 14 Springer International Publishing, (2016): 21-37.
- [17] Redmon, J., and Farhadi, A, “YOLO9000: better, faster, stronger,” Proceedings of the IEEE conference on computer vision and pattern recognition, (2017): 7263-7271.
- [18] Redmon, J., and Farhadi, A, “YOLO9000: better, faster, stronger,” Yolov3: An incremental improvement,” arXiv preprint arXiv:1804.02767, 2018.
- [19] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S, “Feature pyramid networks for object detection,” Proceedings of the IEEE conference on computer vision and pattern recognition, (2017): 2117-2125.

- [20] Bochkovskiy, A., Wang, C. Y., and Liao, H. Y. M., "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
- [21] Wang, C. Y., Liao, H. Y. M., Wu, Y. H., Chen, P. Y., Hsieh, J. W., and Yeh, I. H. A., Wang, C. Y., and Liao, H. Y. M., "CSPNet: A new backbone that can enhance learning capability of CNN," Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, (2020): 390-391.
- [22] He, K., Zhang, X., Ren, S., and Sun, J., "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE transactions on pattern analysis and machine intelligence 37.9, (2015): 1904-1916.
- [23] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J., "Path aggregation network for instance segmentation," Proceedings of the IEEE conference on computer vision and pattern recognition, (2018): 8759-8768.
- [24] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J., "Yolox: Exceeding yolo series in 2021," arXiv preprint arXiv:2107.08430, 2021.

- [25] Wang, C. Y., Bochkovskiy, A., and Liao, H. Y. M., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (2023): 7464-7475.
- [26] Wang, J., and Gu, S., "Fpga implementation of object detection accelerator based on vitis-a," 2021 11th International Conference on Information Science and Technology (ICIST). IEEE, (2021): 571-577.
- [27] Chen, W. H., Hsu, H. J., and Lin, Y. C., "Implementation of a Real-time Uneven Pavement Detection System on FPGA Platforms," 2022 IEEE International Conference on Consumer Electronics-Taiwan. IEEE, (2022): 587-588.
- [28] Li, C., Xu, R., Lv, Y., Zhao, Y., and Jing, W., "Edge Real-Time Object Detection and DPU-Based Hardware Implementation for Optical Remote Sensing Images," Remote Sensing 15.16 (2023): 3975.
- [29] AdvancedMicroDevicesInc, <https://docs.xilinx.com/r/en-US/pg338-dpu/Hardware-Architecture>, (2024.1.31).

- [30] [https://github.com/david8862/keras-YOLOv3-model-](https://github.com/david8862/keras-YOLOv3-model-set)
set,(2024.01.31).
- [31] AvnetInc,<https://github.com/Avnet/Ultra96-PYNQ>,(2024.01.31).
- [32] AdvancedMicroDevicesInc,[https://docs.xilinx.com/r/2.5-](https://docs.xilinx.com/r/2.5-English/ug1414-vitis-ai/Python-APIs)
English/ug1414-vitis-ai/Python-APIs,(2024.1.31).
- [33] COCOdataset,<https://cocodataset.org/>,(2024.1.31).
- [34] Padilla, R., Netto, S. L., and Da Silva, E. A, "A survey on performance metrics for object-detection algorithms," 2020 international conference on systems, signals and image processing (IWS-SIP). IEEE, (2020): 237-242.
- [35] Yarnell, R., Hossain, M., and DeMara, R. F,"Image Quantization Tradeoffs in a YOLO-based FPGA Accelerator Framework," 2023 24th International Symposium on Quality Electronic Design (ISQED). IEEE, (2023): 1-7.
- [36] Zhao, Yonghui, Yong Lv, and Chao Li,"Hardware Acceleration of Satellite Remote Sensing Image Object Detection Based on Channel Pruning," Applied Sciences 13.18 (2023): 10111.

- [37] Verucchi, M., Brilli, G., Sapienza, D., Verasani, M., Arena, M., Gatti, F., ... and Solieri, M,”A systematic assessment of embedded neural networks for object detection,” 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Vol. 1. IEEE, (2020): 937-944.