FPGA実装を指向した部分2値化オートエンコーダに基づく人型ロボットによる柔軟物操作システム

大原 慧[1,a)]　粟野 皓光[2,b)]

概要：FPGA 上に実装可能なニューラルネットワークを用いて人型ロボットで柔軟物を操作するシステムを提案する. 柔軟物操作はロボットによる実現が困難なタスクとして知られているが, 近年ニューラルネットワークを用いたシステムが実用化された. しかし, これらのシステムは GPU に依存しており, 推論システムのロボット本体への組み込みは困難であった. そこで, 性能を落とさずに FPGA に実装できるようネットワークの一部を 2 値化したオートエンコーダを提案する. Xilinx 社の ZCU102 上に実装した結果, 3.1W の電力で 41.1FPS を達成し, Core i7 6700K と RTX 2080 Ti に実装したシステムと比較し, それぞれ 10 倍, 3.7 倍の性能向上を実現した.

キーワード：ヒューマノイドロボット, 機械学習, FPGA, 柔軟物

Binary Neural Network in Robotic Manipulation: Flexible Object Manipulation for Humanoid Robot Using Partially Binarized Auto-Encoder on FPGA

Satoshi Ohara[1,a)]　Awano Hiromitsu[2,b)]

Abstract: A system for manipulating flexible objects with a humanoid robot using a neural network that can be implemented on an FPGA is proposed. Manipulating flexible objects is known to be a difficult task for robots, but recently, systems using neural networks have been put to practical use. However, these systems rely on GPUs, making it difficult to integrate the inference system into the robot itself. In this paper, we propose an auto-encoder where a part of the network is binarized so that it can be implemented on an FPGA without compromising the performance. We implemented the system on Xilinx ZCU102 and demonstrated 41.1FPS at 3.1W power, which is 10× and 3.7× better than the systems implemented on Core i7 6700K and RTX 2080 Ti, respectively.

1. はじめに

衣服のような柔軟物の扱いは, 我々の日常生活において基本的な活動であるため, この種のタスクを担うロボットの実現にますます注目が集まっている. しかし, 柔軟物の形状はロボットのエンドエフェクタの軌道によって大きく変化するため, ロボットによる操作が困難なタスクとして一般的に知られている. この問題を解決するために, ロボットが人間の教示した動作を直接学習する End-to-End 学習が報告されている [1], [2], [3]. Pin-Chu Yang らが提案した手法では, 2 段階の深層学習モデルが導入された. このモデルではまず, Deep Convolutional Auto-Encoder (DCAE) がカメラ入力から低次元の画像特徴を抽出し, 次に Time Delay Neural Network (TDNN) が次時刻の画像特徴とロボットの関節角度を予測する [4]. この手法により彼らはタオルを折り畳むことに成功したが, 数百ワットの電力を消費する GPU ベースのプラットフォームに実装されており, ロボットに組み込むためには小型化や低消費電力化が必要である. そこで我々は FPGA ベースのシステムを提

[1] 大阪大学 大学院情報科学研究科 情報システム工学専攻
[2] 京都大学
[a)] s-ohara@ist.osaka-u.ac.jp
[b)] awano@i.kyoto-u.ac.jp

案する．FPGA は, CPU や GPU のような命令ベースのプロセッサとは異なり, 専用のハードウェア回路によって構成されるため, エネルギー効率が飛躍的に向上する．しかし, FPGA は GPU に比べてメモリ容量が限られているため, 大幅なモデル圧縮が必要となる．モデル圧縮手法の一つである Binarized Neural Network (BNN) は, パラメータの大部分を 2 値化しハードウェア実装に適した特徴を持つことで知られており, MNIST や CIFAR10 の画像分類などいくつかのタスクで成功が報告されている [5]．しかし, DCAE の単純な 2 値化ではネットワークを伝播する勾配が遮断され, 抽出する画像特徴が損なわれることが分かった．そこで我々は Partially-Binarized DCAE (PB-DCAE) と呼ばれる部分的な 2 値化手法を提案する．この手法では, DCAE のエンコーダ部分のパラメータのみを 2 値化し, デコーダ部分のパラメータは変更しない．なお, デコーダ部分は学習時にのみ必要で推論時には削除されるため, 完全に 2 値化した DCAE と比較して追加のメモリを必要としない．本論文の貢献は以下の通りである．

- 我々の知る限り, 初めて BNN をロボット分野に適用した研究である．
- 人型ロボットによる柔軟物操作の End-to-End 学習を実現する PB-DCAE と呼ばれる軽量ニューラルネットワークを提案した．
- Xilinx 社の ZCU 102 に実装された提案モデルは, Core i7 6700K CPU および RTX 2080Ti GPU に実装されたモデルと比較して, エネルギー効率がそれぞれ 10 倍および 3.7 倍向上したことを示した．

2. 関連研究

2.1 ロボットによる物体操作の End-to-End 学習

ロボットによる物体操作の End-to-End 学習は, (1) データ収集, (2) モデル学習, (3) ロボットへのモデル展開の 3 つの段階で構成される．

2.1.1 データ収集

ロボットを遠隔操作する作業環境を構築し, 人間が 3D マウス等を介してロボットのエンドエフェクタを操作できるようにする．遠隔操作の際は, ロボットの頭部に取り付けられたカメラや, ロボットの関節の角度を監視するセンサーなどからロボットの状態を記録する．

2.1.2 モデル学習

ロボットによる End-to-End な物体操作を実現する典型的なネットワーク構造は, 画像から低次元の特徴を抽出する DCAE と画像特徴量と関節角度の時系列データを予測する Recurrent Neural Network (RNN) を組み合わせることである [4], [6]．DCAE は「エンコーダ」と「デコーダ」で構成される左右対称の砂時計型の畳み込みニューラルネットワークで, 次元削減に用いられる [7]．DCAE はエンコーダに与えられた元画像を再構成できるように学習する．DCAE の中間層は, 入出力層に比べてニューロン数が少ないため, 学習した DCAE は, その中間層で元画像の圧縮表現を作成できる．学習後, デコーダは取り除かれ, エンコーダの出力は RNN に入力され, 関節角度を予測する．RNN は, 出力が再帰的に入力として与えられるニューラルネットワークの一種であり, これにより RNN は内部記憶を持ち時系列データを学習・汎化するため, 瞬間的なノイズに対してロバスト性を保つことが出来る．そのため, RNN はロボットの操作に適しており, 描画 [8], 道具の使用 [9], 箱入れ作業 [2] などの様々なタスクに利用されている．RNN はエンコーダによって抽出された画像特徴量と, ロボットの現在の関節角度が与えられると, 次時刻での関節角度の予測できるよう学習される．

2.1.3 モデル展開

まず, ロボット頭部に取り付けられたカメラの画像を学習済みのエンコーダに入力し, 低次元の特徴を抽出する．次にその特徴ベクトルを現在の関節角度と結合し, 次時刻での関節角度を予測するために RNN に与える．最後に予測した関節角度をロボットコントローラに送信し, 実際に関節を回転させる．以上のステップをロボットが物体の操作を完了するまで繰り返す．

2.2 Binarized Neural Networks

モデルサイズを削減するために，[5] はシナプスのパラメータと活性化関数出力の二値化を提案している．すなわち，実値の重みや活性化関数出力は +1 または −1 で近似される．実数の入力テンソルは以下の活性化関数を用いて 2 値化される．

$$x^b = Sign(x) = \begin{cases} +1 & x \geq 0, \\ -1 & otherwise. \end{cases} \tag{1}$$

しかし，Sign 関数の微分はほぼ全てゼロであるため，単純な二値化はネットワーク性能の低下を招く．この問題を解決するために，[5] では，Sign 関数の微分をハイパボリックタンジェントの微分で置き換える「straight-through estimator」という手法を提案している．

$$\frac{\partial}{\partial x} Sign(x) \approx \begin{cases} 1 & |x| < 1, \\ 0 & otherwise. \end{cases} \tag{2}$$

これにより, (1)float32 を基準とするとモデルサイズを 1/32 に縮小でき, (2)XNOR ゲートとビットカウンタのみで MAC (Multiply-and-Accumulate) 演算を実現できる．

2.3 Batch Normalization

straight-through estimator でも, BNN の学習は不安定で, 勾配が頻繁に爆発する傾向がある．そのため，BNN には通常，活性化関数への入力の平均と分散がそれぞれ 0 と

1 になるように標準化する Batch Normalization (BN) 層が付属している [10]．BN 層への入力を x とすると，BN 層は次のように計算する．

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{3}$$

ここで y は BN 層の出力, γ と β は学習パラメータ, μ と σ^2 はミニバッチの平均と分散, そして ϵ は極めて小さい定数である．

式 (3) には除算, 乗算処理が含まれているため，BN を追加すると通常はハードウェア資源が必要になる．しかし, BNN が Sign 関数を使用するため, 推論時の BN 層は, バイアス項をシフトするだけで実現できる．

$$Sign\{BN(x)\} = \begin{cases} +1 & x \geq \lfloor \mu - \frac{\sigma}{\gamma}\beta \rfloor, \\ -1 & otherwise, \end{cases} \tag{4}$$

ここで $\lfloor \cdot \rfloor$ は床関数である．

BNN は, straight-through estimator と BN 層により, CIFAR10 等のタスクで浮動小数点と同等の性能を得ることが報告されている [5]．しかし, 単純に DCAE を BNN に置き換えると, 抽出された画像の特徴が損なわれてしまうことがわかり, 部分二値化を開発した．

3. 提案システム

3.1 提案システムの全体像

図 1 はシステムの全体像を示したもので, カメラ画像を低次元の特徴ベクトルに変換する PB-DCAE と, ロボットアームの関節角度を生成する Long-Short Term Memory (LSTM) の 2 つのニューラルネットワークで構成される．LSTM のモデルサイズは全体の 1.2 %程度であるため, 量子化せず LSTM の性能を維持する．PB-DCAE がバイナリ演算，LSTM が浮動小数点演算という異種の演算に対応するため, プラットフォームに Zynq SoC を採用し，PB-DCAE は programmable-logic (PL) に，LSTM は ARM プロセッサを搭載した processing-system (PS) にそれぞれ実装した．このモデルは，まず GPU を用いて学習される．次に, PB-DCAE のパラメータを C++のヘッダファイルに変換し, Vivado ツールを用いて「.bit」ファイルに変換して PL を構成する．LSTM は, リソースの限られた ARM プロセッサ上で効率的に実行できるよう Numpy を用いて再実装した．

3.2 モデル構造

3.2.1 PB-DCAE による画像特徴抽出モジュール

このモデルを FPGA に実装するためには, オンチップメモリに収まるようモデルサイズを大幅に縮小する必要がある．BNN は, モデル圧縮のための有効な技術である．しかし，4 章で示すように, 素朴な 2 値化は, 抽出された画像特徴の破損につながる．この問題を解決するために，我々は部分的に二値化された DCAE（PB-DCAE）を提案する．この方法では，エンコーダ部分の重みと活性化関数出力のみが二値化される．デコーダ部分は変更されないため, 出力での誤差がネットワークを逆伝播し, より優れた汎化能力を得られる．なお, デコーダは学習段階でのみ必要であり, 推論段階では削除される．したがって, PB-DCAE は, 完全に 2 値化された DCAE と比較して, 追加のメモリコストを必要としない．

3.2.2 LSTM ベースの動作生成モジュール

RNN 部分には, 逐次データの長期的な依存関係を考慮できるようにした LSTM を採用した．LSTM は高い表現力を持つため広く利用されている．R. Rahmatizadeh らは，LSTM を用いて複数のタスクを実現する低コストのロボットアームを開発し [11]，S. Funabashi らは，CNN-LSTM を用いて手に持った物体の操作を実現した [12]．LSTM の入力は, PB-DCAE で抽出された画像特徴とグリッパーのコマンドを含むロボットの関節角度であり, 出力は次時刻の関節角度である．予測された出力を LSTM の入力に再帰的に入力することで, 連続した予測ができ, 学習した LSTM の性能評価に利用する．

3.3 学習

学習は 2 段階に分かれており, まず PB-DCAE を学習して自己組織化された画像特徴を得た後, LSTM を学習する．

PB-DCAE の学習: PB-DCAE は, エンコーダに与えられた入力画像を再構成するように学習する．PB-DCAE は砂時計型の構造をしており, 中間層のニューロン数が入力層や出力層よりも少ないため, ネットワークは圧縮された情報のみを用いて画像を再構成する．よってエンコーダは, デコーダが元の画像をうまく復元できるよう入力画像のより良い圧縮表現を学習する．学習後, デコーダは削除され, エンコーダのパラメータは固定される．

LSTM の学習: 学習済みの PB-DCAE によって抽出された画像特徴は, 関節角度とグリッパーのコマンドと結合され視覚運動シーケンス $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$ になる．ここで, $\boldsymbol{x}_t$ は時刻 t における画像特徴量と関節角度, グリッパーコマンドを含んだベクトルであり, N はシーケンスの長さである．各時間ステップにおいて, LSTM は入力 $\boldsymbol{x}_t$ と現在の状態 $\boldsymbol{h}_t$ を受け取り, $\boldsymbol{y}_{t+1}$ と更新された状態 $\boldsymbol{h}_{t+1}$ を出力する．$f_{\mathrm{LSTM}}(\cdot, \cdot)$ を LSTM の動作を表す関数とすると, 入力と出力の関係は $\{\boldsymbol{y}_{t+1}, \boldsymbol{h}_{t+1}\} = f_{\mathrm{LSTM}}(\boldsymbol{x}_t, \boldsymbol{h}_t)$ と表される．$\{\boldsymbol{y}_{t+1}, \boldsymbol{h}_{t+1}\}$ を LSTM に再帰的に与えることで, $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_t)$ を生成する．視覚運動シーケンスの予測が目的であるため, $\boldsymbol{X}$ と $\boldsymbol{Y}$ の平均二乗誤差を最小化することで学習する．しかし, このマルチステップの予測は学習が困難であり, 長いシーケンスでは特に損失が発散する傾向にある．そこで学習を安定させるため, シングルステップとマルチステップの予測による損失を組み合わ

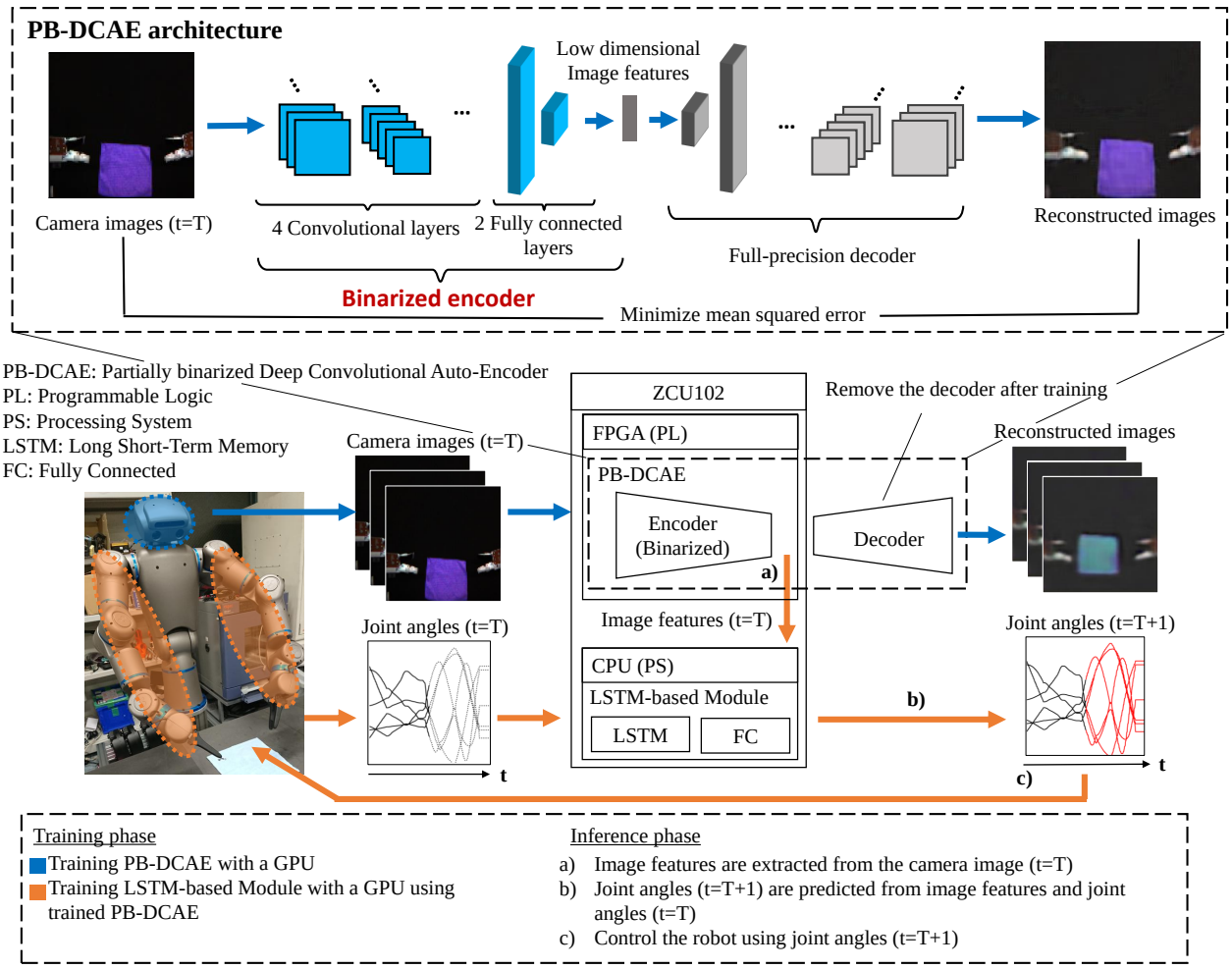

図 1 システムの全体図

せて利用することを提案する．図 2 に我々の学習の構成を示す．

1) マルチステップの予測による損失: 図 2 の下に示すように，LSTM の出力を入力に再帰的に与えることでマルチステップの予測による損失を計算する．$\boldsymbol{Y}^{\mathrm{m}} = (\boldsymbol{y}_1^{\mathrm{m}}, \boldsymbol{y}_2^{\mathrm{m}}, \cdots, \boldsymbol{y}_t^{\mathrm{m}})$ は生成されたシーケンスである．ここで，マルチステップの予測による損失 L_m は次のように表される．

$$L_m = \frac{1}{2N} \sum_{t=1}^{N} (\boldsymbol{x}_t - \boldsymbol{y}_t^m)^2. \tag{5}$$

2) シングルステップの予測による損失: マルチステップの予測と異なり，LSTM は教師データとして各時間ステップにおける視覚運動シーケンスを受け取る．すなわち，視覚運動シーケンス $\boldsymbol{x}_t$ と内部状態 $\boldsymbol{h}_t^s$ が与えられると，$\left\{\boldsymbol{y}_{t+1}^s, \boldsymbol{h}_{t+1}^s\right\} = f(\boldsymbol{y}_t^s, \boldsymbol{h}_t^s)$ を計算する．ここで再帰的に与えられるのは内部状態 $\boldsymbol{h}_{t+1}^s$ のみであり，出力 $\boldsymbol{y}_{t+2}^s$ は図 2 の上部に示すように $\left\{\boldsymbol{y}_{t+2}^s, \boldsymbol{h}_{t+2}^s\right\} = f_{\mathrm{LSTM}}(\boldsymbol{x}_{t+1}, \boldsymbol{h}_{t+1}^s)$ として計算される．この手順を繰り返すことでシングルステップの予測によるシーケンス $\boldsymbol{Y}^s = (\boldsymbol{y}_1^s, \boldsymbol{y}_2^s, \cdots, \boldsymbol{y}_t^s)$ が

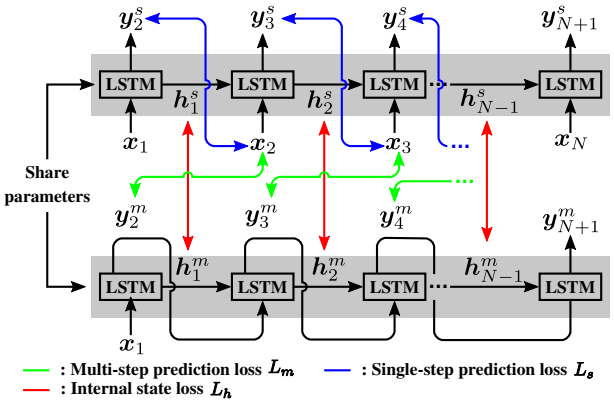

図 2 LSTM training setup.

得られる．ここで，シングルステップの予測による損失 L_s は次のように表される．

$$L_s = \frac{1}{2N} \sum_{t=1}^{N} (\boldsymbol{x}_t - \boldsymbol{y}_t^s)^2. \tag{6}$$

3) 内部状態の損失: 内部状態は教師データがないため，マルチステップの予測時に小さい誤差が蓄積し，勾配が発散する恐れがある．そこでシングルステップ予測時の内部

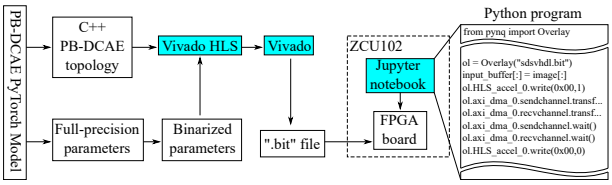

図 3　デザインフロー

状態 $\boldsymbol{h}_t^s$ とマルチステップ予測時の内部状態 $\boldsymbol{h}_t^m$ を近づけるために損失 L_h を追加する．

$$L_h = \frac{1}{2N}\sum_{t=1}^{N}(\boldsymbol{h}_t^s - \boldsymbol{h}_t^m)^2. \tag{7}$$

最終的な損失関数は 3 つの損失を線形結合した次の式で表される．

$$L = \alpha L_m + \beta L_s + \gamma L_h, \tag{8}$$

ここで α, β, γ はハイパーパラメータである．

3.4　実装

提案モデルを Zynq UltraScale+ MPSoC ZCU102 ボード上に実装した．この FPGA ボードのオンチップ RAM と LUT は非常に限られているため, 推論のスループットを維持しつつ, ハードウェア資源の使用量を削減するよう実装する必要がある．全体の設計フローを図 3 に示す．まず, PyTorch と呼ばれるフレームワークにより PB-DCAE を学習し, ネットワークのパラメータを抽出する．次に, PB-DCAE の C++モデルを作成し, Xilinx 社の 2 つの Vivado ソフトウェアを使用して高位合成し, FPGA ボードを適切に構成するための「.bit」ファイルを作成する．最後に, 生成されたすべてのファイルを Xilinx ZCU102 の FPGA ボード上にアップロードし, 性能のテストと測定を行う。ZCU102 には, Python Productivity of Zynq (PYNQ) 環境がインストールされているため, Jupyter Notebook を使いプログラムの実行や結果の確認ができる．なお, 画像特徴抽出モジュールは PL に, LSTM ベースの動作生成モジュールは PS に実装した．DCAE による画像特徴抽出モジュールはシングルステップの推論に必要な全乗算の 99.94%を占めるため, LSTM を PS で実行した場合でもシステム全体の性能を大幅に向上できる．

FPGA 実装: 図 4 は FPGA に実装した PB-DCAE のブロック図である．データの移動を最小限に抑えるため, 各層に専用の演算ユニットを割り当て, 各層を AXI-Stream インターフェースで接続する．限られたハードウェア資源で効率的にデータストリームを処理するために, 演算に必要なデータのみを保持するシフトレジスタを用意し, 畳み込み処理を行う．また, 2 値化により浮動小数点の積和演算は XNOR とビットカウントで代替できる．さらに, BN は乗算器や加算器が必要だが, 後続の Sign 関数と合わせて式変換することで整数比較に変換できる．これらによって回路

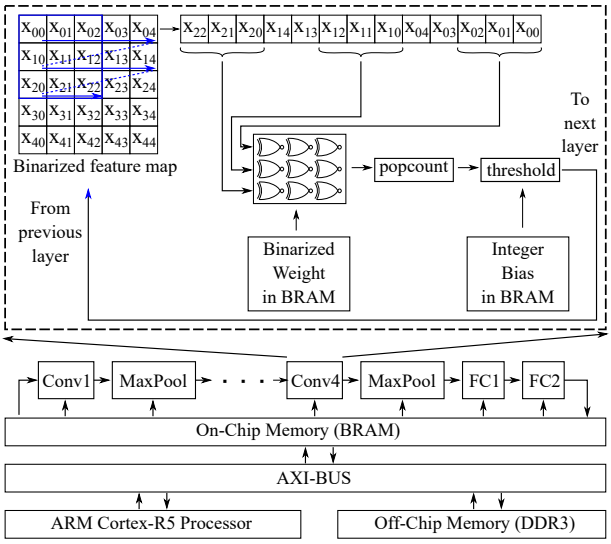

図 4　ストリーム 2 値化畳み込み回路

規模を縮小して実装できる．また, モデルサイズは 2 値化によって圧縮されすべてのパラメータをオンチップ BRAM に展開できる．したがって, PB-DCAE の計算中に必要となる特徴マップと重みをオンチップメモリに保存し, 外部 I/O を介さないため, 通信によるレイテンシを最小限に抑えることができ, 低消費電力でシステムを実行する．

4.　実験

4.1　実験環境

実験用のデータセットは, カワダロボティクス社が開発した Nextage Open を用いて収集した．Nextage Open は, ロボット頭部に取り付けられたカメラとグリッパーが付属した 2 本の腕を備えており, 各腕は 6 自由度を持つ．学習データを収集するために, ロボットの遠隔操作環境を構築し, 人間が 3D マウスを使ってグリッパーの位置を操作できるようにした．この環境で操作者がタオルを折り畳むタスクを実行し, その間にカメラ映像 (142×142×3=60,492 次元, RGB), 右腕の関節角度 (6×2=12 次元) とグリッパーコマンド (1 次元) を記録する．なお, タオルを折り畳む際は右腕のグリッパーのみを使用している．トレーニング用に 50 シーケンス, 評価用に 25 シーケンスの合計 75 のタスクシーケンスを収集した．その結果, モデルの学習には約 22000 ステップ, 評価には約 11000 ステップのデータを使用した．式 (8) 中の α, β, γ にはそれぞれ 0.1, 1.0, 0.1 を使用した．モデルの構造は表 1 に示す．

4.2　実験結果

部分 2 値化の効果を検証するためにまず, 図 5 に示すように, 浮動小数, 部分 2 値化, 完全 2 値化の 3 種類の DCAE の復元精度を比較した．一番左の画像は, タオル折り畳みタスク実行時の最初に撮影された入力画像であり, 他の 3 つの画像は 3 種類の DCAE が復元した画像である．図 5

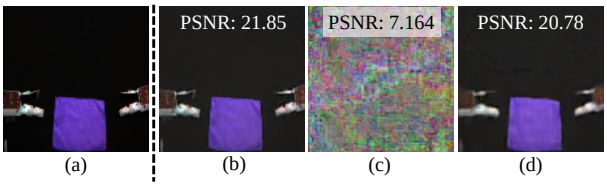

図 5 (a) Original and reconstructed images by (b) full-precision DCAE, (c) fully-binarized DCAE, and (d) PB-DCAE.

によると浮動小数型と部分 2 値化 DCAE に大きな違いはない一方で, 完全 2 値化 DCAE では破損した画像が出力されている. 復元画像の品質を定量的に比較するため, 不可逆圧縮コーデックの画質を測定する一般的な方法の一つである Peak Signal-to-Noise Ratios (PSNRs) を計算した. PSNR は次の式で表される.

$$PSNR = 10 \log_{10} \frac{MAX_I^2}{MSE}. \qquad (9)$$

PSNR が高いほど高品質であり, [13] によると 20 db 以上の PSNR は実用上十分な品質であるとされる. 図 5 に示すように浮動小数型の DCAE の平均 PSNR が 21.85 dB である一方で, 完全 2 値化 DCAE の平均 PSNR は 7.164 dB であるため, 単純な 2 値化は復元画像の破損に繋がることが分かる. また, PB-DCAE の平均 PSNR は 20.78 dB であり, 部分 2 値化 DCAE と同等の精度であることが確認された.

次に, 2 値化が関節角度とグリッパーコマンドの予測精度に与える影響を確認する. このために LSTM のマルチステップの予測を比較した. すなわち時刻 $t = 1$ における画像特徴, 関節角度, グリッパーコマンドのみを与え, LSTM の出力を再帰的に LSTM に入力することで $t = 2, 3, \cdots, N$ を予測しシーケンス全体を比較する. 浮動小数型, 完全 2 値化, 部分 2 値化の出力と評価用シーケンスの平均二乗誤差を計算するとそれぞれ 2.98°, 6.32°, 3.43° であった. 表 1 に示すように PB-DCAE では, 51.3MB から 2.20MB に大幅に削減されたにも関わらず, 浮動小数型 DCAE と同等の復元精度が得られた.

最後に, 提案手法と GPU ベースの従来手法のエネルギー効率を比較する. Zynq ZCU102 ボード，CPU，GPU の消費電力をそれぞれ「Maxim powertool」,「s-tui」コマンド,「nvidia-smi」コマンドを用いて測定した. 表 2 2 に各プラットフォームの処理時間，消費電力，エネルギー効率を示す. 表 2 から ZCU102 は市販の RGB カメラのフレームレート上限である 30FPS を満たしつつ, 最も高いエネルギー効率を達成していることが確認できる. このエネルギー効率は CPU の 10 倍, GPU の 3.7 倍に相当する.

5. 結論

ロボットによる柔軟物操作のために, PB-DCAE という軽量ニューラルネットワークを提案した. DCAE のエンコーダ部分のみを 2 値化することで，浮動小数点型の DCAE の性能を維持しつつ，モデルサイズの大幅な圧縮に成功した．PB-DCAE とその他の周辺モジュールを Zynq ZCU102 ボードに実装したところ，関節角度の予測精度の劣化を 0.45◦ に抑えつつ，モデルサイズを 95.7 %削減できることを確認した．さらに，同じモデルを Core i7 6700K と RTX 2080 Ti に実装しエネルギー効率を調べると，CPU ベースのプラットフォームでは 10 倍，GPU ベースのプラットフォームでは 3.7 倍の効率が得られることを確認した.

表 1 Network architecture and model size.

Layer	Out. F size	In. F maps	Float-besed model	Proposed model
Conv1	142×142	3	3.38 KB	0.105 KB
Max Pool	70×70	32	–	–
Conv2	70×70	32	72.0 KB	2.25 KB
Max Pool	34×34	64	–	–
Conv3	34×34	64	288 KB	9.00 KB
Max Pool	16×16	128	–	–
Conv4	16×16	128	1.15 MB	36.0 KB
Max Pool	7×7	256	–	–
FC1	1024	12544	50.2 MB	1.57 MB
FC2	64	1024	256 KB	8.00 KB
LSTM1	100	77	280 KB	280 KB
LSTM2	100	100	316 KB	316 KB
FC3	77	100	30.4 KB	30.4 KB
Total	–	–	51.3 MB	2.20 MB

表 2 Performance comparison.

	Core i7 6700K	RTX 2080Ti	ZCU102
Preprocessing [msec]	0.759	1.47	4.45
Conv1-FC2 [msec]	12.4	2.24	15.4
LSTM1-FC3 [msec]	0.420	0.315	4.48
Total [msec]	13.7	4.03	24.3
FPS [sec-1]	73.0	248	41.1
Power [W]	55.4	69	3.1
Efficiency [FPS/W]	1.32	3.59	13.3

謝辞

本研究は，JST，さきがけ，JP-MJPR18M1，JSPS 科研費 21H03409 の支援を受けたものである.

参考文献

[1] Kento Kawaharazuka, Toru Ogawa, Juntaro Tamura, and Cota Nabeshima. Dynamic manipulation of flexible objects with torque sequence using a deep neural network. In *Int. Conf. on Robotics and Automat. (ICRA)*, pp. 2139–2145, 2019.

[2] Kei Kase, Kanata Suzuki, Pin-Chu Yang, Hiroki Mori, and Tetsuya Ogata. Put-in-Box Task Generated from Multiple Discrete Tasks by aHumanoid Robot Using Deep Learning. In *Int. Conf. on Robotics and Automat. (ICRA)*, pp. 6447–6452, 2018.

[3] Namiko Saito, Nguyen Ba Dai, Tetsuya Ogata, Hiroki Mori, and Shigeki Sugano. Real-time Liquid Pouring Motion Generation: End-to-End Sensorimotor Coordination for Unknown Liquid Dynamics Trained with Deep Neural Networks. In *Int. Conf. on Robotics and Biomimetics (ROBIO)*, pp. 1077–1082, 2019.

[4] Pin-Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. Repeatable folding task by humanoid robot worker using deep learning. Vol. 2, No. 2, pp. 397–403, 2016.

[5] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In *Int. Conf. on Neural Inf. Process. Syst. (NIPS)*, p. 4114–4122, 2016.

[6] Kei Kase, Ryoichi Nakajo, Hiroki Mori, and Tetsuya Ogata. Learning Multiple Sensorimotor Units to Complete Compound Tasks using an RNN with Multiple Attractors. In *Int. Conf. on Intell. Robots and Syst. (IROS)*, pp. 4244–4249, 2019.

[7] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Int. Conf. on Artif. Neural Networks*, pp. 52–59. Springer, 2011.

[8] Kazuma Sasaki, Hadi Tjandra, Kuniaki Noda, Kuniyuki Takahashi, and Tetsuya Ogata. Neural network based model for visual-motor integration learning of robot's drawing behavior: Association of a drawing motion from a drawn image. In *Int. Conf. on Intell. Robots and Syst. (IROS)*, pp. 2736–2741, 2015.

[9] Kuniyuki Takahashi, Tetsuya Ogata, Hadi Tjandra, Yuki Yamaguchi, and Shigeki Sugano. Tool-body assimilation model based on body babbling and neurodynamical system. *Math. Problems in Eng.*, Vol. 2015, , 2015.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. on Mach. Learn. (ICML)*, pp. 448–456, 2015.

[11] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *Int. Conf. on Robot. and Automat. (ICRA)*, pp. 3758–3765, 2018.

[12] Satoshi Funabashi, Shun Ogasa, Tomoki Isobe, Tetsuya Ogata, Alexander Schmitz, Tito Pradhono Tomo, and Shigeki Sugano. Variable In-Hand Manipulations for Tactile-Driven Robot Hand via CNN-LSTM. In *Int. Conf. on Intell. Robots and Syst. (IROS)*, 2020.

[13] Nikolaos Thomos, Nikolaos V Boulgouris, and Michael G Strintzis. Optimized transmission of JPEG2000 streams over wireless channels. Vol. 15, No. 1, pp. 54–67, 2005.