Vol.2019-DPS-181 No.4
2019/12/26

畳み込みニューラルネットワークにおける Octave Convolution のパラメータ数削減手法の提案

井上 勇[1]　後藤 佑介[1]

概要：近年，大規模化したデータを分析するため，計算機による機械学習が大きな注目を集めている．特に，画像認識とオブジェクト検出の分野において，畳み込みニューラルネットワーク (CNN) が広く用いられている．CNN に関する研究では，精度の向上を目的としたモデル，およびパラメータ数や計算コストの削減を目的とした軽量なモデルがそれぞれ提案されている．Octave Convolution (OctConv) は，従来の畳み込み層による処理を OctConv 層による処理に置き換えることで，モデルのメモリコストおよび計算コストを削減するとともに，精度を向上できる手法である．しかし，OctConv で用いるパラメータ数は，従来の畳み込み処理の場合と比べてほとんど変わらない．本研究では，OctConv で用いるパラメータ数を削減してより軽量なモデルを作成するため，パラメータ数の削減手法である Pointwise convolution を OctConv に組み合わせた手法として Pointwise Octave Convolution を提案する．提案手法では，OctConv 層で行われる各経路の畳み込み処理の前後で Pointwise convolution をそれぞれ実行することでパラメータ数を削減する．ResNet-56 を用いた評価では，$\alpha = 0.25$ の場合，分類精度の低下を 0.02% に抑えるとともに，パラメータ数を約 24.4% 削減した．

1. はじめに

近年，大規模化したデータを分析するため，計算機による機械学習が大きな注目を集めている．特に，画像認識とオブジェクト検出の分野において，畳み込みニューラルネットワーク（CNN）が広く用いられている [1]．CNN において，精度の向上を目的としたモデルとして，GoogLeNet [2]，VGG [3]，ResNet [4]，および DenseNet [5] が提案されている．また，パラメータ数や計算コストの削減を目的とした軽量なモデルとして，SqueezeNet [6]，MobileNets [7]，ShuffleNet [8]，および MobileNetV2 [9] が提案されている．

CNN における計算コストの削減手法である Octave Convolution (OctConv) [10] は，従来の畳み込み層による処理を OctConv 層による処理に置き換えることで，モデルの計算コストおよびメモリコストを削減するとともに，精度を向上できる手法である．しかし，OctConv で用いるパラメータ数は，従来の畳み込み処理の場合と比べてほとんど変わらない．

本研究では，OctConv で用いるパラメータ数を削減してより軽量なモデルを作成するため，パラメータ数の削減手法である Pointwise convolution (Pointwise Conv) を OctConv に組み合わせた手法として，Pointwise Octave Convolution (Pointwise OctConv) を提案する．提案手法では，OctConv 層で行われる各経路の畳み込み処理の前後で Pointwise Conv をそれぞれ実行することで，パラメータ数を削減する．

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

2. 畳み込みニューラルネットワーク

畳み込みニューラルネットワーク（以下，CNN）は，複数段の層をもつニューラルネットワークであり，特に画像認識の分野で高い性能をもつ．本章では，CNN の構成要素である畳み込み層とプーリング層について説明する．

2.1 畳み込み層

畳み込み層では，カーネルと呼ばれる小さな特徴検出器の集まりを用いて，元の画像から特徴点を抽出する．各カーネルは，画像上で一定の領域をピクセルごとにスライドし，それぞれの領域で重み付き和を計算することで，カーネルが表す特徴的な構造を抽出できる．このとき，入力画像から特徴を抽出した出力データを特徴マップと呼ぶ．

畳み込み計算は，局所領域でカーネルを介して実行するため，画像上におけるすべての場所で特徴を抽出できる．この性質を移動不変性と呼ぶ．

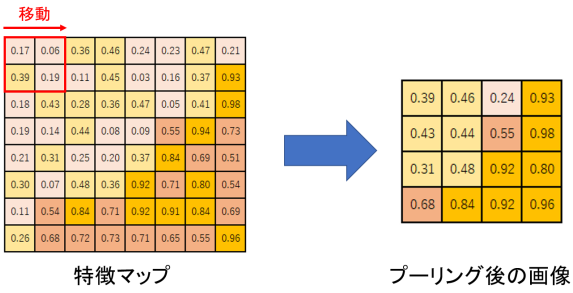

図 1　Max Pooling による画像圧縮の例

2.2　プーリング層

プーリング層では，特徴となる重要な情報を欠落させずに元の画像を圧縮する．**図 1** に，小さな領域に対して領域内の最大値を選択する手法である Max Pooling を用いた画像圧縮の例を示す．はじめに，左上からピクセルごとに移動して，2×2 の 4 ピクセルを順番に選択する．図 1 に示すように，選択した 4 ピクセルが 0.17，0.06，0.19，0.39 である場合，最大値である 0.39 を選択し，プーリング後の画像に対応するピクセルに挿入する．以下，同様の手順を繰り返し，特徴マップから新たな画像を生成する．プーリング層で画像を圧縮することで，画像の移動および回転といった位置変化による影響を減少できる．

3.　畳み込み層におけるパラメータ数削減手法

畳み込み層におけるパラメータ数削減手法として，Pointwise convolution（Pointwise Conv），および Depthwise convolution（Depthwise Conv）が挙げられる．本章では，一般的な畳み込み層，Pointwise Conv，および Depthwise Conv の 3 種類について，パラメータ数および計算コストについて説明する．

3.1　一般的な畳み込み層

はじめに，一般的な畳み込み層について述べる．**図 2** に示すように，入力特徴マップのサイズを $w \times w$，入力チャネル数を c，カーネルサイズを $k \times k$，および出力チャネル数を f とする．

入力特徴マップ上の畳み込みについて，一箇所あたりの計算量は k^2c となる．この計算量を入力特徴マップにおける w^2 箇所に適用することで，1 チャネルの出力特徴マップが生成される．このため，出力特徴マップが f 個の場合，計算コストは k^2cw^2f となる．また，パラメータ数について，パラメータ数が k^2c となる畳み込みが f 個あるため，k^2cf となる．

3.2　Pointwise convolution

図 3 に示す Pointwise convolution (Pointwise Conv) [7] は，GoogLeNet [2] における Inception module，および

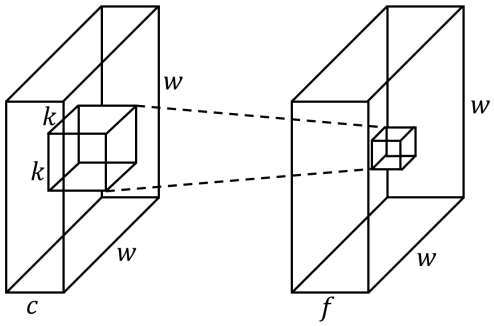

図 2　一般的な畳み込み層

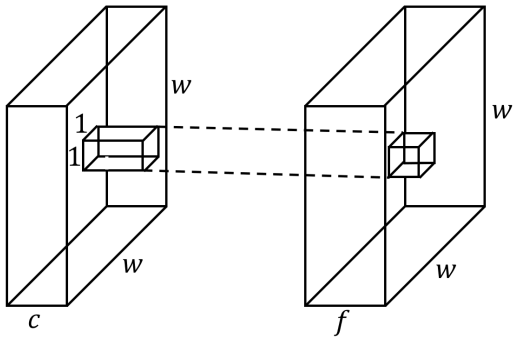

図 3　Pointwise convolution

ResNet [4] における bottleneck 構造でそれぞれ利用される 1×1 の畳み込みである．Pointwise Conv では，特徴マップの空間方向に対して畳み込みを行わず，チャネル方向に対する畳み込みを行うことで，パラメータ数を削減できる．また，特徴マップにおけるチャネル数の増減を目的とする場合にも用いられる手法である．Pointwise Conv は $k = 1$ の場合における一般的な畳み込み層であり，計算コストは w^2cf，パラメータ数は cf となる．

3.3　Depthwise convolution

図 4 に示す Depthwise convolution (Depthwise Conv) [7] は，特徴マップのチャネルごとに空間方向の畳み込みを行う．チャネル方向の畳み込みを行わないため，一回の畳み込みにかかるコストは k^2 となる．また，Depthwise Conv の場合，入力チャネル数と出力チャネル数が等しくなるため，$c = f$ となる．以上より，Depthwise Conv の計算コストは w^2ck^2，パラメータ数は k^2c となる．

MobileNets [7] , MobileNetV2 [9]，および Xception [11] といったモデルでは，Pointwise Conv と Depthwise Conv を組み合わせて適用する．このため，空間方向とチャネル方向を同時に畳み込む一般的な畳み込み層と比べて，少ないパラメータ数および計算コストで近似できる．

4.　Octave Convolution

Octave Convolution (OctConv)[10] は，特徴マップを高

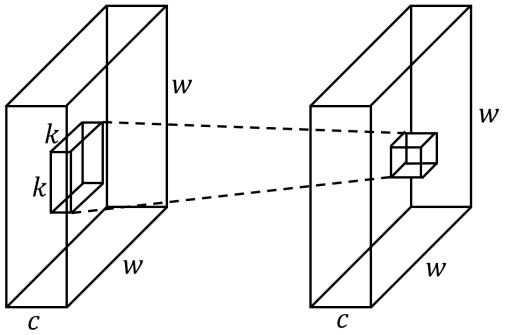

図 4　Depthwise convolution

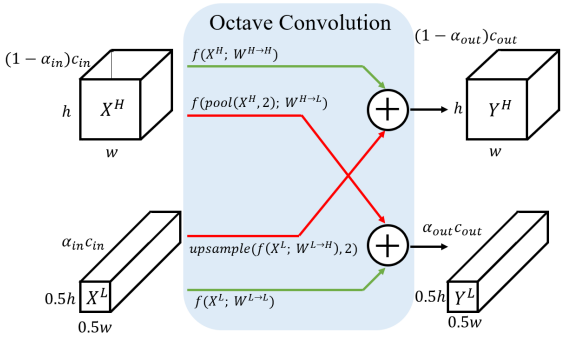

図 5　OctConv の詳細設計

周波成分と低周波成分の 2 種類に分解した上で，各成分で畳み込みを行う手法である．

4.1　OctConv の設計

$X, Y \in \mathbb{R}^{c\times h\times w}$ を満たす X を入力テンソル，Y を出力テンソルとする．また，入力 X を高周波成分と低周波成分の 2 種類に分解し，それぞれ $X^H \in \mathbb{R}^{(1-\alpha)c\times h\times w}$，$X^L \in \mathbb{R}^{\alpha c\times \frac{h}{2}\times \frac{w}{2}}$ とする．このとき，h および w は空間次元，c はチャネル数，$\alpha \in [0,1]$ は低周波成分に割り当てたチャネルの比率をそれぞれ表す．

同様に，出力 Y の高周波成分を Y^H，低周波成分を Y^L とし，それぞれ $Y^H = Y^{H\to H} + Y^{L\to H}$，$Y^L = Y^{L\to L} + Y^{H\to L}$ で与える．ここで，$Y^{A\to B}$ は，特徴マップのグループ A からグループ B に対する畳み込みの更新を表す．具体的には，$Y^{H\to H}$ および $Y^{L\to L}$ は周波数内で情報を更新し，$Y^{H\to L}$ および $Y^{L\to H}$ は異周波数間で情報を更新する．

$W \in \mathbb{R}^{c\times k\times k}$ は $k\times k$ の畳み込みカーネルとし，2 種類の構成要素 $W = [W^H, W^L]$ に分割して，X^H および X^L の畳み込みを行う．また，各構成要素は，$W^H = [W^{H\to H}, W^{L\to H}]$，および $W^L = [W^{L\to L}, W^{H\to L}]$ といった周波数内部分と異周波数間部分にそれぞれ分割できる．

OctConv の詳細設計を**図 5** に示す．OctConv は，4 種類の計算経路から構成されており，出力 $Y = \{Y^H, Y^L\}$ を以下のように書き換えできる．

$$Y^H = f(X^H; W^{H\to H}) + \text{upsample}(f(X^L; W^{L\to H}), 2) \quad (1)$$

$$Y^L = f(X^L; W^{L\to L}) + f(\text{pool}(X^H, 2); W^{H\to L}) \quad (2)$$

ここで，$f(X; W)$ は，パラメータ W を用いた畳み込み計算を示す．$\text{pool}(X; k)$ は，カーネルサイズが $k\times k$ および移動幅を表すストライドが k の場合における Average Pooling の実行を示す．また，$\text{upsample}(X; k)$ は，Nearest-neighbor interpolation における係数 k の UpSampling の実行を示す．図 5 における 2 種類の緑色の矢印は，高周波成分内および低周波成分内における特徴マップの情報更新にそれぞれ対応する．また，2 種類の赤色の矢印は，2 種類の周波数間による情報交換に対応する．

以上のように，特徴マップを空間上の高周波成分と低周波成分に分けた上で情報交換することで，精度を向上できる．また，低周波成分を低解像度化することで，計算コストとメモリ使用量を削減できる．

4.2　OctConv の計算コストおよびパラメータ数

OctConv の計算コストおよびパラメータ数について説明する．OctConv では，$H\to H$, $H\to L$, $L\to H$, $L\to L$ の 4 種類の経路でそれぞれ畳み込み計算を行う．$c_{in} = c_{out} = c$，$\alpha_{in} = \alpha_{out} = \alpha$ とすると，各経路における計算コストは，それぞれ以下の通りである．

$$FLOPS(Y^{H\to H}) = h\times w\times k^2\times (1-\alpha)^2\times c^2 \quad (3)$$

$$FLOPS(Y^{H\to L}) = \frac{h}{2}\times\frac{w}{2}\times k^2\times\alpha\times(1-\alpha)\times c^2 \quad (4)$$

$$FLOPS(Y^{L\to H}) = \frac{h}{2}\times\frac{w}{2}\times k^2\times(1-\alpha)\times\alpha\times c^2 \quad (5)$$

$$FLOPS(Y^{L\to L}) = \frac{h}{2}\times\frac{w}{2}\times k^2\times\alpha^2\times c^2 \quad (6)$$

よって，OctConv の計算コストは，(3) 式から (6) 式までの値を足し合わせ，以下の式になる．

$$FLOPS([Y^H, Y^L]) = (1-\frac{3}{4}\alpha(2-\alpha))\times h\times w\times k^2\times c^2 \quad (7)$$

一般的な畳み込みの計算コストは，$h\times w\times k^2\times c^2$ である．このとき，OctConv と一般的な畳み込みの計算コストの比率は，同じ入力チャネル数，出力チャネル数，およびカーネルサイズを用いた場合，$1-\frac{3}{4}\alpha(2-\alpha)$ となる．この比率は α の値が 1 に近づくにつれて増加するため，OctConv の計算コストを削減できる．

また，OctConv のパラメータ数は，以下の式で示され，一般的な畳み込みと同じである．

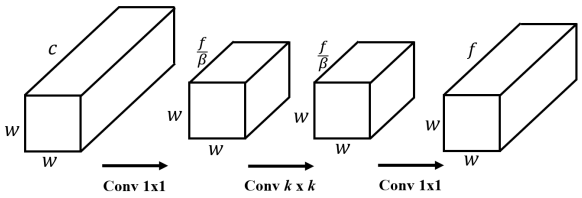

図 6 bottleneck 構造

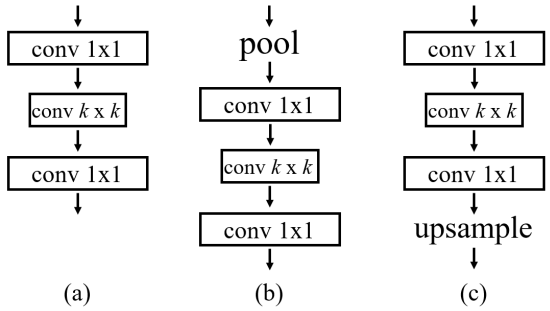

図 7 提案手法の構造

$$k^2(1-\alpha)^2c^2+k^2\alpha(1-\alpha)c^2+k^2(1-\alpha)\alpha c^2+k^2\alpha^2c^2=k^2c^2 \tag{8}$$

5. 提案手法

5.1 概要

本研究では，OctConv と Pointwise Conv を組み合わせたパラメータ数削減手法として，Pointwise Octave Convolution (Pointwise OctConv) を提案する．提案手法では，ResNet [4] の Residual Units における bottleneck 構造 [12] を導入する．また，**図 6** に示すように，畳み込み層を Pointwise Conv で挟み込むことで，1 回目の Pointwise Conv でチャネル数を減らし，2 回目の Pointwise Conv でチャネル数を増やす．これにより，ボトルネックであった $k \times k$ の畳み込み層の入力チャネル数および出力チャネル数を減少でき，パラメータ数を削減できる．

5.2 構造

提案手法では，各経路の畳み込み層を Pointwise Conv で挟み込む．$H \to H$，$L \to L$ の 2 経路では，**図 7**(a) に示すように，単純に畳み込み層を Pointwise Conv で挟み込む．また，$H \to L$ の経路では，図 7(b) に示すように，Avrage Pooling の実行後に畳み込み層を Pointwise Conv で挟み込む．さらに，$L \to H$ の経路では，図 7(c) に示すように，畳み込み層を Pointwise Conv で挟み込んだ後に UpSampling を実行する．最後に，これらの 4 経路に対してそれぞれ評価し，分類精度低下の抑制およびパラメータ数の削減の効果がもっとも大きい経路を選択する．

5.3 計算コスト

図 6 において，1 回目の Pointwise Conv における計算コストは $w^2c\frac{f}{\beta}$ となる．このとき，前後の Pointwise Conv で挟まれた通常の畳み込み層における計算コストは $k^2w^2\left(\frac{f}{\beta}\right)^2$，2 回目の Pointwise Conv における計算コストは $w^2\frac{f^2}{\beta}$ となる．ここで，β はチャネルの圧縮率である．したがって，全体の計算コストは以下の式で表される．

$$w^2c\frac{f}{\beta}+k^2w^2\left(\frac{f}{\beta}\right)^2+w^2\frac{f^2}{\beta}=\frac{w^2f}{\beta^2}(k^2f+\beta(c+f)) \tag{9}$$

このとき，$w=32$，$c=f=16$，$k=3$，$\beta=2$ とすると，計算コストは約 63.9% 減少する．

最後に，(9) 式より，OctConv における各経路の計算コストは，

$$FLOPS(Y^{H\to H})=\frac{(1-\alpha)^2}{\beta^2}\cdot hwc^2(k^2+2\beta) \tag{10}$$

$$FLOPS(Y^{H\to L})=\frac{\alpha}{4\beta^2}\cdot hwc^2(\alpha k^2+\beta) \tag{11}$$

$$FLOPS(Y^{L\to H})=\frac{(1-\alpha)}{4\beta^2}\cdot hwc^2((1-\alpha)k^2+\beta) \tag{12}$$

$$FLOPS(Y^{L\to L})=\frac{\alpha^2}{4\beta^2}\cdot hwc^2(k^2+2\beta) \tag{13}$$

となる．

5.4 パラメータ数

bottleneck 構造のパラメータ数は，以下の式で表される．

$$c\cdot\frac{f}{\beta}+k^2\cdot\frac{f}{\beta}\cdot\frac{f}{\beta}+\frac{f}{\beta}\cdot f=\frac{f}{\beta^2}(k^2f+\beta(c+f)) \tag{14}$$

5.3 節の (9) 式と同様に，$w=32, c=f=16, k=3, \beta=2$ の場合，パラメータ数は約 63.9% 減少する．

6. 評価

本章では，提案手法と既存手法の比較評価を行う．評価では，データセットとして CIFAR-10 [13] を使用する．

6.1 CNN モデルの構成

評価では，ResNet [4] を用いて評価を行う．ResNet に対して既存手法である OctConv [10] を実装した CNN モデルを Oct-ResNet，および提案手法である Pointwise OctConv を実装した CNN モデルを Pwise-Oct-ResNet とする．Oct-ResNet では，最初の畳み込み層を除くすべての畳み込み層を OctConv 層に置き換えており，最初と最後の OctConv 層を除くすべての層では $\alpha_{in}=\alpha_{out}=\alpha$ と設定する．また，最初の OctConv 層では $\alpha_{in}=0$，$\alpha_{out}=\alpha$ と設定し，最後の OctConv 層では $\alpha_{in}=\alpha$，$\alpha_{out}=0$ と設

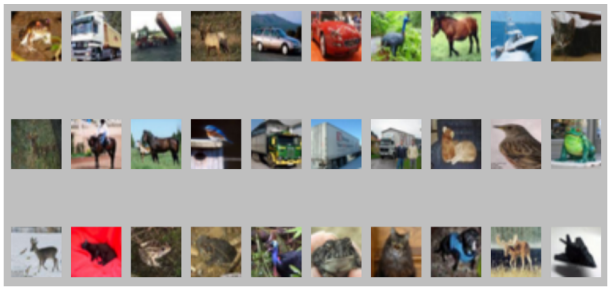

図 8　CIFAR-10 の画像例

定する．PwiseOct-ResNet では，Oct-ResNet と同様の設定で，各経路の畳み込み層を Pointwise Conv で挟み込む．評価では，$\beta = 2$ と設定する．すべてのモデルは，CNN で用いる最適化アルゴリズムを表すオプティマイザとして，モメンタムを用いる．

6.2　CIFAR-10

CIFAR-10 [13] は，一般物体認識に用いる評価用のデータセットである．画像の大きさは 32 × 32 ピクセルであり，RGB のカラー画像である．画像全体は 60,000 枚であり，50,000 枚の訓練画像と 10,000 枚のテスト画像で構成されている．CIFAR-10 において分類精度の計算で用いるクラスラベルは，airplane，automobile，bird，cat，deer，dog，frog，horse，ship，および truck の 10 種類である．CIFAR-10 におけるデータセットの画像例を**図 8** に示す．評価では，0.1 の割合で上下左右シフトおよび水平反転をさせる Data Augmentation を行う．

6.3　ResNet-56 を用いた評価

表 1 に，ResNet-56，Oct-ResNet-56，およびすべての経路に Pointwise Conv を実装した場合における Pwide-Oct-ResNet-56 の評価結果をそれぞれ示す．計算量は疑似コードを用いて算出した．

表 1 より，提案手法は高周波成分のチャネルの比率が多い $\alpha = 0.25$ の場合および低周波成分のチャネルの比率が多い $\alpha = 0.75$ の場合の両方において，Oct-ResNet-56 に比べてパラメータ数を約 63.8% 削減した一方で，分類精度は 約 3.0% 低下した．表 1 の評価では，すべての経路に Pointwise Conv を実装してパラメータ数を大きく削減したため，分類精度に影響を与えた．

次に，各 α の値における分類精度低下の抑制およびパラメータ数の削減に効果的な経路を評価する．**表 2** に，Pwise-Oct-ResNet-56 において経路別に Pointwise Conv を実装した場合の結果を示す．表 2 より，$\alpha = 0.25$ で $H \to H$ 経路に Pointwise Conv を実装した場合，分類精度の低下が大きい．また，$\alpha = 0.75$ で $L \to L$ 経路に Pointwise Conv を実装した場合も同様に，分類精度の低下が大きい．チャネルの比率が大きい同周波成分内で情報

表 1　手法ごとの評価結果 (ResNet-56)

モデル	α	精度 [%]	パラメータ数	計算量
ResNet-56	-	92.58	0.86×10^6	1.25×10^8
Oct-ResNet-56	0.25	93.13	0.86×10^6	0.85×10^8
Pwise-Oct-ResNet-56		90.31	0.31×10^6	0.36×10^8
Oct-ResNet-56	0.75	91.62	0.86×10^6	0.38×10^8
Pwise-Oct-ResNet-56		88.58	0.31×10^6	0.18×10^8

更新を行う経路を用いる場合，分類精度に与える影響が大きい．

以上より，分類精度の低下を抑制しつつパラメータ数を削減するため，$\alpha = 0.25$ の場合は $H \to H$ 経路以外，および $\alpha = 0.75$ の場合は $L \to L$ 経路以外でパラメータ削減を行う必要がある．

表 2　各経路における評価結果 (Pwide-Oct-ResNet-56)

経路	α	精度 [%]	パラメータ数
$H \to H$	0.25	91.19	0.52×10^6
$H \to L$		93.11	0.72×10^6
$L \to H$		93.38	0.82×10^6
$L \to L$		93.27	0.82×10^6
$H \to H$	0.75	91.12	0.82×10^6
$H \to L$		91.54	0.82×10^6
$L \to H$		91.33	0.72×10^6
$L \to L$		90.70	0.52×10^6

表 3 に，$H \to H$ 経路もしくは $L \to L$ 経路でパラメータ削減を行わない場合における Pwise-Oct-ResNet-56 の比較結果を示す．表 3 より，$\alpha = 0.25$ で $H \to H$ 経路によるパラメータ削減を行わない場合，Pwise-Oct-ResNet-56 は Oct-ResNet-56 に比べてパラメータ数を約 24.4% 削減するとともに，分類精度の低下は約 0.02% と小さい．また，すべての経路でパラメータ削減を行った場合における Pwise-Oct-ResNet-56 に比べて，分類精度が約 2.8% 向上した．次に，$\alpha = 0.75$ で $L \to L$ 経路によるパラメータ削減を行わない場合，Pwise-Oct-ResNet-56 はすべての経路でパラメータ削減を行った場合における Pwise-Oct-ResNet-56 に比べて分類精度が約 1.9% 向上し，Oct-ResNet-56 との分類精度の差は約 1.2% となった．

6.4　ResNet-110 を用いた評価

ResNet-56 に比べてニューラルネットワークの階層が深い ResNet-110 による評価結果を**図 4** に示す．図 4 より，$\alpha = 0.25$ で $H \to H$ 経路によるパラメータ削減を行わない場合，および $\alpha = 0.75$ で $L \to L$ 経路によるパラメータ削

表 3　$H \to H$ 経路もしくは，$L \to L$ 経路を除いた場合の評価結果

モデル	α	精度 [%]	パラメータ数	計算量
Pwise-Oct-ResNet-56（$H \to H$ 経路は除く）	0.25	93.11	0.65×10^6	0.79×10^8
Pwise-Oct-ResNet-56（$L \to L$ 経路は除く）		90.69	0.35×10^6	0.37×10^8
Pwise-Oct-ResNet-56（$H \to H$ 経路は除く）	0.75	90.02	0.35×10^6	0.23×10^8
Pwise-Oct-ResNet-56（$L \to L$ 経路は除く）		90.46	0.65×10^6	0.29×10^8

減を行わない場合の両方について，Pwise-Oct-ResNet-110 はすべての経路でパラメータ削減を行った場合の Pwise-Oct-ResNet-110 に比べて分類精度が約 2.5% 向上した．また，$\alpha = 0.25$ で $H \to H$ 経路によるパラメータ削減を行わない場合，Pwise-Oct-ResNet-110 は Oct-ResNet-110 に比べて分類精度が 0.42% 低下した．一方で，パラメータ数は約 25.4% 削減した．

表 4　手法ごとの評価結果 (ResNet-110)

モデル	α	精度 [%]	パラメータ数	計算量
ResNet-110	-	93.59	1.74×10^6	2.53×10^8
Oct-ResNet-110	0.25	93.84	1.74×10^6	1.71×10^8
Pwise-Oct-ResNet-110		90.99	0.59×10^6	0.70×10^8
Pwise-Oct-ResNet-110（$H \to H$ 経路は除く）		93.42	1.30×10^6	1.58×10^8
Oct-ResNet-110	0.75	92.18	1.74×10^6	0.76×10^8
Pwise-Oct-ResNet-110		89.04	0.59×10^6	0.34×10^8
Pwise-Oct-ResNet-110（$L \to L$ 経路は除く）		91.46	1.30×10^6	0.56×10^8

7. おわりに

本研究では，OctConv で用いるパラメータ数を削減してより軽量なモデルを作成するため，パラメータ数の削減手法である Pointwise convolution を OctConv に組み合わせた手法として，Pointwise Octave Convolution (Pointwise OctConv) を提案した．提案手法では，OctConv 層で行われる各経路の畳み込み処理の前後で Pointwise convolution をそれぞれ実行することで，パラメータ数を削減する．評価の結果，$\alpha = 0.25$ の場合，$H \to H$ 経路を除いた 3 種類の経路において提案手法を実装した ResNet-56 は，既存手法に比べて分類精度の低下を約 0.02% に抑えるとともに，パラメータ数を約 24.4% 削減した．

今後は，ResNet 以外のモデルを用いた評価，およびより大規模なデータセットを用いた評価を行う．

謝辞

本研究は，文部科学省による Society 5.0 実現化研究拠点支援事業によって行われたものである．ここに記して謝意を表す．

参考文献

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks, In Advances in Neural Information Processing Systems, pp.1097 - 1105 (2012).
[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich: Going Deeper with Convolutions, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1 - 9 (2015).
[3] K. Simonyan, and A. Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv (online), available from < https://arxiv.org/pdf/1409.1556.pdf > (accessed 2019-11-15).
[4] K. He, X. Zhang, S. Ren, and J. Sun: Deep Residual Learning for Image Recognition, arXiv (online), available from < https://arxiv.org/pdf/1512.03385.pdf > (accessed 2019-11-15).
[5] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger: Densely Connected Convolutional Networks, arXiv (online), available from < https://arxiv.org/pdf/1608.06993.pdf > (accessed 2019-11-15).
[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size, arXiv (online), available from < https://arxiv.org/pdf/1602.07360.pdf > (accessed 2019-11-15).
[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv (online), available from < https://arxiv.org/pdf/1704.04861.pdf > (accessed 2019-11-15).
[8] X. Zhang, X. Zhou, M. Lin, and J. Sun: ShuffleNet: An Extremely Efficient Convolutional Neural Networks for Mobile Devices, arXiv (online), available from < https://arxiv.org/pdf/1707.01083.pdf > (accessed 2019-11-15).
[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen: MobileNetV2: Inverted Residuals and Linear Bottlenecks, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.4510 - 4520 (2018).
[10] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng: Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution, arXiv (online), available from < https://arxiv.org/pdf/1904.05049.pdf > (accessed 2019-11-15).
[11] F. Chollet: Xception: Deep Learning with Depthwise Separable Convolutions, arXiv (online), available from <

https://arxiv.org/pdf/1610.02357.pdf > (accessed 2019-11-15).
[12] K. He, X. Zhang, S. Ren, and J. Sun: Identity Mappings in Deep Residual Networks, arXiv (online), available from < https://arxiv.org/pdf/1603.05027.pdf > (accessed 2019-11-15).
[13] The CIFAR-10 and CIFAR-100 datasets (online), available from < https://www.cs.toronto.edu/~kriz/cifar.html > (accessed 2019-11-15).