C-003

学習済モデルのパラメータを活用した再学習手法の検討
Re-Training Method Using Parameters of Learned Models

伊原 和美† 中西 知嘉子‡
Kazumi Ihara Chikako Nakanishi

1. はじめに

1.1 背景

近年，AI 技術の進展がめざましく，エッジ AI とよばれる技術の利用が進んでいる．エッジ AI とは，携帯電話や車載端末などの安価な端末上で推論を行う技術のことである．これらの安価な端末のことをエッジ端末という．

エッジ AI は推論時に通信を必要としないため，様々な場所での動作が可能で，通信による遅延時間を抑えられるメリットをもつ．一方で，使用できる計算リソースの量が限られてしまうデメリットをもつ．そのため，エッジ端末で高性能な AI を動作させる場合，使用するリソースの量が少なくなるよう設計をしなければならない．

1.2 目的と環境

そこで本研究では，エッジ AI の精度を保ちつつ計算量を削減する手法について提案した．今回は，機械学習モデルの計算に時間がかかっている畳み込み演算部分に着目した．畳み込み演算を行っている箇所の構造を部分的に変更することで，計算の高速化が実現できないか検討した．

畳み込み演算部分を置き換えるモジュールとして，畳み込み演算よりも高速かつ精度低下を抑えた Ghost Module[1]を採用した．Ghost Module については 3 章 3 節で述べる．

また，構造を部分的に変更する手法として，学習済ネットワークの中間出力を活用し，Ghost Module 化する際に追加される層に含まれるパラメータのみを学習させる方法を採用し，再学習にかかる時間を抑えることができないか検討した．

2. 使用したツール

本研究では，機械学習の中から特によく使用される CNN(畳み込みニューラルネットワーク)を取り上げた．その中で，VGG16 を使用した．また，データセットとして，CIFAR-10 を用いた．

また，学習に用いるライブラリとして PyTorch，機械学習モデルを表現するフレームワークとして ONNX を用いた．推論ツールとして Ceras を用いた．

2.1 VGG16

VGG とは，2014 年にオックスフォード大学の Visual Geometry Group によって発表されたニューラルネットワークである[2]．畳み込み層，プーリング層，全結合層が直列に接続された単純な構造をもつ点が特徴である．

VGG16 とは，深さが 16 層の VGG を指す．VGG16 の構造を図 1 に示す．

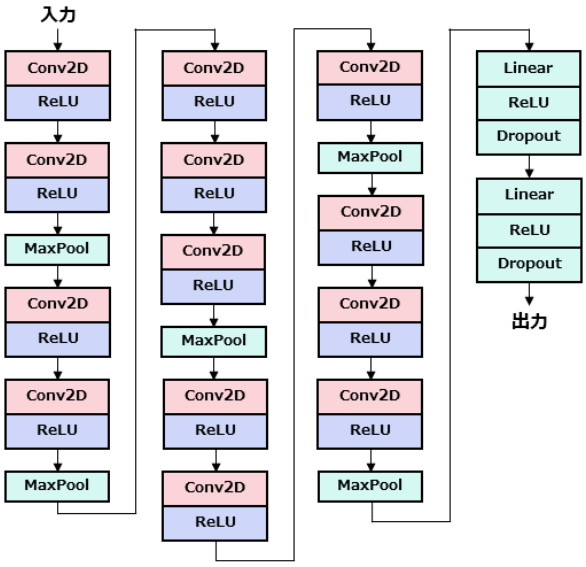

図 1　VGG16 の構造

Conv2D は畳み込み層を示す．(また，Linear は全結合層を示す．)図 1 に示すように，VGG16 には 13 層の畳み込み層が存在する．本研究では，これらの畳み込み層について Ghost Module に置き換えた際の性能の変化，学習にかかった時間を検証した．

2.2 CIFAR-10

CIFAR-10 とは，Alex Krizhevsky，Vinod Nair，Geoffrey Hinton によって公開された画像データセットである[3]．10 クラスからなり，5 万枚の訓練データ，1 万枚のテストデータで構成されている．含まれている画像は，すべて 24bit のフルカラー画像であり，縦のサイズは 32 ピクセル，横のサイズは 32 ピクセルである．

2.3 Ceras

Ceras(C++ edge rapid ai simulator)とは，本研究室で開発された，C++言語上で機械学習モデルの推論処理を実行するツールである[4]．機械学習モデルを扱う際に独自のフォー

† 大阪工業大学 情報科学研究科 情報科学専攻　Graduate School of Information Science and Technology Osaka Institute of Technology
‡ 大阪工業大学 情報科学部 情報知能学科　Department of Information and Computer Science Osaka Institute of Technology

マットへの変換が必要であるが，ONNX 形式で表現された機械学習モデルは Ceras で扱う形式への変換が可能である．この独自形式へと変換したモデルは C++標準ライブラリのみで動作するという利点がある．組み込み機器などの様々な環境で容易に動作させることが可能であるため，本研究で採用した．

3. 提案手法

本章では，まず本研究で使用する畳み込み演算，Depthwise 演算，Ghost Module について触れた後，本研究で提案する手法について述べる．

3.1 畳み込み層

畳み込み演算とは，入力されたデータに対して，カーネルを順に平行移動させながら積和演算を行った結果を出力とする演算である．畳み込み層の処理手順を図 2 に示す．

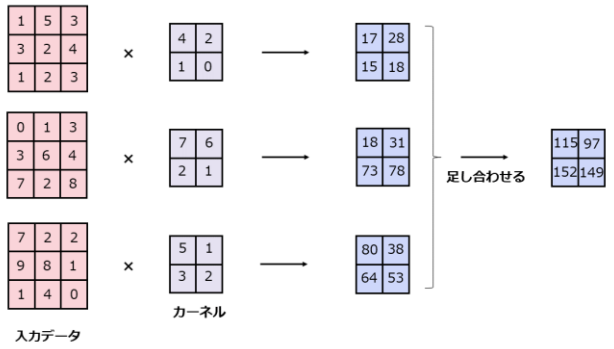

図 2　畳み込み層の処理手順

畳み込み層では，まず，入力されたデータのチャネルごとにカーネルを用意し，対応する入力データのチャネルとカーネルで畳み込み演算が行われる．そして，全てのチャネルの演算結果を足し合わせたものが最終的な出力となる．図 2 で示したのは，1 カーネルあたりの畳み込み層の処理内容である．カーネルが複数ある場合の入力データのチャネル数とカーネル枚数について図 3 に示す．

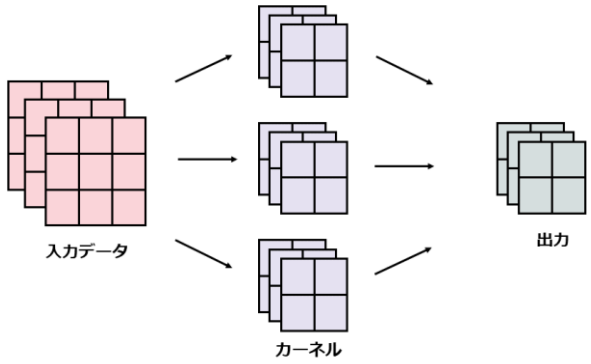

図 3　畳み込み演算

図 3 のように，出力 1 チャネル畳み込み層に必要となるカーネルの枚数は入力チャネル数×出力チャネル数となる．

3.2 Depthwise 層

次に，Depthwise 層の処理手順を図 4 に示す．

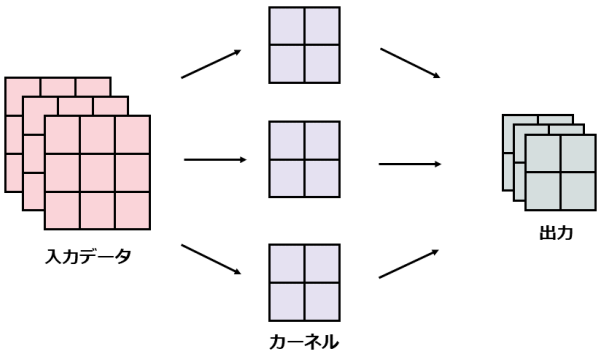

図 4　Depthwise 層の処理手順

Depthwise 層と畳み込み層ではどちらも畳み込み演算を行うが，用いるカーネルの枚数と計算手順が異なる．畳み込み層では入力データと各チャネルの畳み込み演算を行った結果を加算して 1 チャネルの出力としていたのに対し，Depthwise 層では入力データの個々のチャネルに対して，1 チャネルのカーネルを用いて畳み込み演算を行う．出力 1 チャネルに対して 1 チャネルのカーネルが 1 枚存在するのみであり，通常の畳み込み層と比較して保持するデータの量が少なく軽量であることが特徴である．

3.3 Ghost Module

Ghost Module とは，Kai Han らによって発表された，畳み込み層にある畳み込み演算を一定の割合でより軽量な演算へと置き換えたモジュール構造である．

1 例として，入力データのチャネル数 3，出力のチャネル数 4，3 チャネルのカーネルを 4 枚使用する畳み込み層を，0.5 の割合で Depthwise 演算へと置き換えた Ghost Module を挙げる．畳み込み層のカーネル数は元の畳み込み層の半分の 2 へと削減される．この畳み込み層の出力結果を用いて Depthwise 演算を行い，チャネル数 2 の出力を得る．畳み込み層の 2 チャネルの出力と，軽量な演算部分の 2 チャネルの出力を連結した 4 チャネルの出力が，Ghost Module の最終的な出力となる．最終的な出力のチャネル数は，元の畳み込み層の出力のチャネル数と一致する．

本研究で使用した Ghost Module の構造を図 5 に示す．

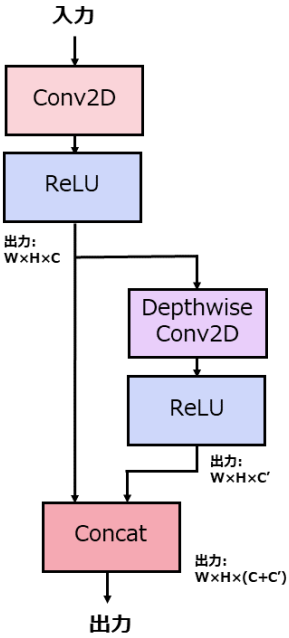

図 5　Ghost Module の構造

Concat 層は，2 つの入力を受け取り，チャネル方向に連結したものを出力とする層である．

演算の流れは次の通りである．まず，入力データの畳み込み層による演算後，出力結果が Depthwise 層へと入力される．Depthwise 層の出力と畳み込み層の出力を Concat 層により連結した結果が，Ghost Module の出力となる．Concat 層からの出力のチャネル数は，畳み込み層の出力のチャネル数と Depthwise 層からの出力のチャネル数を足したものになる．

このモジュールは，畳み込み層からの出力は互いに類似したペアが存在していることから，それらのペアは畳み込み演算よりも安価な演算によって生成可能ではないか，という考えによって作成された．実際，GhostNet の論文では，畳み込み層と遜色ない精度で実行することができ，モデルによってはむしろ精度が向上したことが述べられている．また，このモジュールの利点は，より軽量な演算へと置き換えることによって実行時間の短縮が見込める点と保持するパラメータの量が少なくなる点である．

3.4 提案手法

本研究で提案する，畳み込み層の Ghost Module への置換方法について述べる．提案手法は，①学習部分，②置換部分の 2 つから構成される．①学習部分の処理手順を図 X に示す．

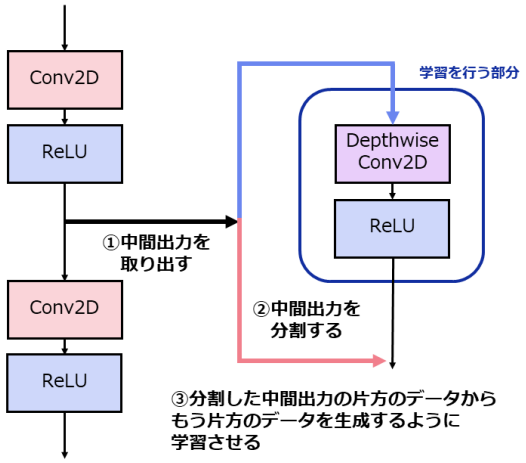

図 6　提案手法(①学習部分)

研究では置き換える構造部分の学習のみを行っている．学習部分の詳細な処理手順を次に示す．

まず，既存のモデルの中間出力を一定の比率で分割する．そして，分割した中間出力の前半部分を入力，後半部分を出力として教師データを作成し，Depthwise Conv2D 層を学習させる．この操作を，Ghost Module へ置き換える畳み込み層の数だけ繰り返し行う．学習中，モデル全体の再学習は行わず，既存のモデルのパラメータが変更されることはない．

次に，置換部分の流れを示す．

Depthwise 層を Ceras 実行形式へと変換する際に Conv2D 層の後ろへ接続し，Conv2D 層の出力を利用するように組み込む．Conv2D 層の出力と Depthwise 層の出力を連結する Concat 層を追加する．さらに，Conv2D の重みを分割し，前半部分のみを使用する．こうすることで，学習を行った際に前半部分の出力を作成したカーネルのみを使用することができる．

4. 評価方法と検証

CIFAR-10 で学習を行った VGG16 に対して，1 層～13 層の畳み込み層の中間出力を学習させ，Ghost Module へ置き換えた際の正解率の変化を図 7 に示す．

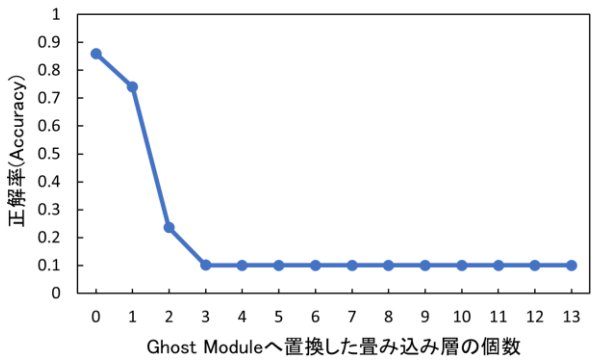

図 7　正解率の変化

図 7 に示したように，Ghost Module へ置換した畳み込み層の個数が増えるほど，精度が顕著に低下したことがわかる．

また，推論時間の変化，Ceras 上で実行する形式へと変換した際の 1 枚当たりの推論時間と容量の変化を表 1 に，学習にかかった時間を
表 2 に示す．

表 1　推論時間，容量の変化

置き換えた層の数	推論時間[s/枚]	容量[MB]
0	3.71	233.93
4	3.47	231.90
8	2.80	210.90
13	1.91	117.31

表 2　学習にかかった時間

Ghost Module の数	学習時間（提案手法）[min]	学習時間（従来手法）[min]
1	3.83	55.76
2	7.63	56.98
3	11.29	56.09
4	14.95	56.12
5	18.54	58.90
6	22.14	59.60
7	25.73	59.01
8	29.28	60.66
9	32.83	64.81
10	36.39	67.03
11	39.95	68.61
12	43.53	70.75
13	47.07	69.30

モデルファイルの容量はオーダ X に従って減少しており，推論時間もパラメータの減少度に比例して減少している．また，学習時間も，大きなモデルを学習するよりも調整すべきパラメータの数が少ないため，大幅に抑えることができた．しかし，精度の低下が著しく，このままでは実用に堪えない．

5. 結論

本研究では，機械学習モデルの畳み込み層を Ghost Module へ置換し，置換部分のみを学習させる手法を提案した．

結果，提案手法では著しい精度の低下が見られ，実用に堪えるほどの精度をもつモデルを作成することができなかった．

提案手法には，次の 2 点の問題があると考える．まず 1 点目に，差分がもたらす影響が考慮されていなかったことである．すべての Ghost Module は既に学習が行われているモデルの中間出力を利用して学習を行った．しかし，中間出力を活用して学習させた Ghost Module は，元の畳み込み層を完全に再現できるとはいえない．1 層の Ghost Module が生み出す元の出力との差は小さくとも，複数層実行されることで差が蓄積され，最終的に無視できない程の大きな差となる．したがって，精度が大きく低下してしまったと考えられる．2 点目に，置換前の畳み込み層の出力を分割する際に，類似する出力のペアについて考慮を行わなかった点である．Ghost Module の論文では，似たペアが存在することに基づき，ペアの片方を畳み込み演算によって生成し，Depthwise の演算によってペアのもう片方を補っている．

今後の課題として，上記 2 点を考慮した学習方法を考案することがあげられる．まず差分のもたらす影響を調査し，影響の度合いにより，提案手法がどのような場合ならば有効となるか検討したい．また，学習済モデルのパラメータまたは出力を活用して，類似ペアが出力されるカーネルを同定し，再学習手法の再検討に使用したい．

参考文献

[1] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, Chang Xu, “GhostNet: More Features from Cheap Operations”, CVPR 2020
[2] Karen Simonyan, Andrew Zissermanm, “Very Deep Convolutional Networks for Large-Scale ImageRecognition”, ICLR2015
[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 'Learning multiple layers of features from tiny images', Alex Krizhevsky, 2009
[4] 西岡駿, 中西知嘉子, "機械学習ライブラリの C 言語化の実現", 電子情報通信学会ソサイエティ大会(2021)