一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2018-113,DC2018-95（2019-03）

一対多のデータフローにおける共有CNNを用いたマルチタスク分割推論実行

平賀由利亜† 中田 尚† 中島 康彦†

† 奈良先端科学技術大学院大学 情報科学研究科 〒630–0192 奈良県生駒市高山町 8916–5
E-mail: {hiraga.yuria.hu6, nakada, nakashim}@is.naist.jp

あらまし 本稿では深層学習アプリケーションにおけるネットワークの輻輳問題とクラウドへの計算負荷の集中を軽減するため、マルチタスクにおける共有 CNN を用いた深層学習の推論の分割と中間データを圧縮する手法を提案し、RGB 画像に共通の前処理をしたものを入力とした共有可能なタスクと全体計算量、転送データサイズの削減割合について調査を行う。ILSVRC2012 の物体認識タスクと PASCAL VOC の物体検出タスクに対して中間データの圧縮に BPG 圧縮を利用した場合、物体認識タスクでは圧縮前のモデルの精度と比較すると Top1 では VGG16 は 2.11%、MobileNetV2 は 2.59%の精度の劣化を示し、物体検出タスクでは SSD300 と比較をすると VGG16 は 19.93%、MobileNetV2 は 14.91%の精度を劣化を示した。同時に、VGG16 を使用した場合に全体計算量を 42.30%、データ転送量の 36.19%の削減を行った。MobileNetV2 では全体計算量を 10.83%削減し、データ転送量の 68.70%の削減を行った。

キーワード Convolutional Neural Network, Edge Computing, Fine-tuning, Multi-task Neural Network

1. はじめに

インターネットの普及や計算機、センサーテクノロジーの発展により、様々なサービスが我々の生活に浸透している。特にインターネットにつながるモノである IoT(Internet of Things) デバイスの数は増え続け、2016 年時点の 173 億個から 2020 年には 300 億個まで増加することが予想される [1]。IoT デバイスを利用したシステムでは、（1）IoT デバイスに接続されたセンサーから取得した入力情報を圧縮し、（2）ネットワーク通信路、基地局を介してサーバーへ転送、（3）サーバーは入力情報を展開、（4）演算処理より結果を得る、という手順で処理が実行される。サーバーがクラウドに置いてあると仮定すると、増え続ける IoT デバイスからのデータ転送に対してエッジデバイス-クラウド間の通信網が一極集中であるため、トラフィック輻輳が発生する可能性がある。解決策の一つがエッジコンピューティングである。エッジコンピューティングとは端末の近くにサーバーを分散配置するネットワーク技法であり、ユーザーや端末の近くでデータ処理を行うことで上位システムへの演算やネットワークへの負荷を軽減する手法である。

私は実用的な深層学習アプリケーションとして、一つのエッジデバイスで取得したセンサーデータをネットワークを経由して複数のクラウドに転送し、それぞれ別のタスクの深層学習の推論を行うモデルを考える。画像のようなデータサイズの大きいセンサーデータを送る場合、転送先の数だけセンサーデータを送る必要があり、ネットワークの輻輳問題や行うタスク数に限りが出る。ネットワークの輻輳問題に対して、エッジデバイス側で深層学習の一部の処理を行い、出力される特徴量が元のデータサイズよりも小さくなった段階で転送をする手法が考えられる。しかし、エッジデバイスの計算資源では複数のタスクを同時に処理することは難しい。別のアプローチとして、Convolutional Neural Network(CNN) を共有する共有 CNN を用いたマルチタスクの深層学習モデルが考えられる。深層学習では、前半層は汎用的な特徴を学習し、後半層はタスクに特化した特徴を学習していると考えている [2]。深層学習を用いたマルチタスク学習では、深層学習モデルの前半層をタスク間で共有し、後半層はタスクごとに分岐するモデル構成となっており、マルチタスク学習のモデル構成を推論に応用することで、タスクあたりの計算量を削減し、共有層の処理は単一なのでエッジデバイス側で行うことができる。本稿では深層学習アプリケーションにおけるネットワークの輻輳問題とクラウドへの計算負荷の集中を軽減するため、マルチタスクにおける共有 CNN を用いた深層学習の推論の分割を提案し、提案するモデルにおける RGB 画像を入力とした場合の共有可能なタスクと、中間データの転送を行う際の圧縮手法について調査を行う。提案する分散深層学習モデルは、あらかじめ学習を行いパラメータを固定した共有 CNN を用いてタスクごとに再学習を行った複数の深層学習モデルから構成されている。エッジで共有 CNN による推論を行い、元画像よりも要素数が減少した段階で中間データを圧縮し、クラウドに転送することで、アプリケーション全体のネットワークトラフィック量を削減する。

2. 関連研究

エッジコンピューティングを利用した深層学習の推論の分割では、全体の処理を前半と後半に分割する。処理の前半部をエッジデバイス上で実行し、実行結果である中間データをサーバーへ転送、後半の処理を行う手法を示す。Y. Kang [3] らの研究では 3 種類のタスク、8 つのモデルを対象に、各層で分割したときのレイテンシとエッジデバイスでの処理、データ転送、

クラウドでの処理で消費したエネルギーについて計測を行い、最も良い分割点について考察がされている。研究内で使用されているエッジデバイスとクラウドの構成では、画像認識のタスクに対する VGG16 [4] を利用した推論において、レイテンシでは pool5 での分割点が最も良く、エネルギー消費量ではクラウド側で全ての処理を行うことが最も良いという結果が出ている。使用する機器、タスクやモデルによって結果は異なるが、使用環境による最適な性能を引き出す手段の一つに推論の分割という選択肢がある。現実的なアプリケーションとして、推論の分割でも転送するデータサイズを考慮すれば、中間データの転送時には圧縮を行うことが考えられる。動画像を利用したサーバークライアントモデルでは、エッジデバイスで動画像を取得して圧縮を行う。その後、クラウドへ転送を行い、圧縮されたデータを解凍し、処理を行う。H. Choi らの研究 [5] では、VGG16 を用いた物体認識、YOLOv2 [6] を用いた物体検出に対して推論をエッジデバイス側、クラウド側の二箇所に分散する。エッジデバイス側で生成された中間データをクラウドに転送する際に動画圧縮手法である H265/HEVC [7] を用いた圧縮を行なっている。これらの手法では入力に使用する画像とタスクが一対一であることが前提である。圧縮条件や学習条件をタスクごとに調節することができる一方で、エッジデバイスで取得したデータを複数のタスクに適用することは考慮されておらず、複数のタスクをこれらの手法で実現するためには、計算資源の乏しいエッジデバイス側で並列に推論を実行するか、順番に推論を繰り返す必要がある。タスク数が多い場合や、リアルタイム性を求めるアプリケーションが多い場合に、アプリケーションの要求を満たすことが難しくなる。また、不特定多数の第三者に配信しそれぞれのタスクに対して推論をするようなことは想定されていないため、推論の詳細が明かされていない場合にこれらの手法を採用することはできない。

S. Kornblith らの研究 [8] では、ILSVRC のような学習が困難な大規模データセットで良い結果を出したモデルは、ハードパラメーターシェアリングによって他のタスクに転移可能な特徴量抽出器として機能することを仮定している。研究では、ILSVRC2012 で使用された IMAGENET のデータセットを用いて学習をした 13 個の深層学習モデルを特徴量抽出器として利用し、得られた特徴量をロジスティック回帰への入力として、検証のために用意した 12 個のタスクについて学習を行なっている。この実験では、ILSVRC2015 で最も良い精度を出した Resnet-152 [9] が最も良い結果を出している。また、異なる種類のタスクに対して汎用的なモデルの提案もされている。MobileNet の研究 [10] [11] では、スマートフォンのような末端端末でも効率良く動作するようにモデルに必要なパラメーター数や演算回数を減らす試みがされている。MobileNetV2 の研究 [11] では、高効率な演算だけでなく、深層学習で後段の層に伝搬する度に情報の損失が起こる問題に対して反転残差構造を挿入することによって活性化関数による情報の損失を軽減させている。順伝搬時の情報の損失を軽減させることで、画像認識、物体検出、セマンティックセグメンテーションの 3 つのタスクに対してパラメーターが十分に持つモデルと比べて遜色ない結果を示している。しかしながら、深層学習モデルを特徴量抽出器として利用した実験では単にタスクごとに転移学習を行なったのみであり、特徴量抽出器として利用する共有 CNN の構成について画像認識以外のタスクへの影響は考慮されていない。また、推論の分割を目的としているわけではないため、共有 CNN と後段のタスク別深層学習モデルのデータの受け渡しに圧縮処理等は行われていない。MobileNetV2 の研究では、同一のモデル構造で物体認識、物体検出、領域検出の複数タスクに対応することを検討しているが、入力画像の大きさが異なることや、物体検出では feature map の転送も行われるため、共有 CNN を用いた推論の分割に直接利用することはできない。

3. 提 案 手 法

入力データに対して複数のタスクを処理する場合、図 1 のようにタスク数分のモデルを動作させることが考えられる。IoT デバイスとクラウドを利用したサーバークライアントモデルを考えた場合に、IoT デバイスで取得したデータをクラウドに転送し、計算リソースが潤沢にある環境で処理が行われる。一方で、エッジコンピューティングにより全ての処理をエッジ側で処理するためには、計算リソースとタスクの処理に必要な深層学習モデルの大きさを考慮する必要がある。リアルタイムに動作させる必要のあるタスクがある場合には、必要なタスク数分のモデルを並列で動作させる必要があるが、計算リソースの乏しいエッジ側では並列で動作させることのできるタスク数には限りがあるため、並列実行可能なタスク数に制限が出る。計算リソースに対して深層学習モデルの大きさを削減する試みには、深層学習モデルの蒸留や使用するデータ単位の縮小が研究されている。しかしながら、入力データに対して並列に動作させる必要のあるタスクが増えるに従い、要求される計算リソースも増え続けるため解決策にはならない。

これらの問題に対して、ハードパラメーターシェアリングを利用したマルチタスク学習のモデル構造を推論に応用し、エッジ-クラウド間で計算を分散することによってエッジコンピューティングにおける計算リソースが不足する問題に対処することができる。エッジ-クラウドを協調した複数タスクに対する深層学習モデルの推論実行を想定し、マルチタスク DNN 向け推論の分割を提案する。従来における DNN の推論は、以下の手順で処理される。

（1） 末端デバイスは接続されたセンサからデータを取得し、圧縮する

（2） 圧縮済みデータはネットワーク通信路を介しサーバへ転送される

（3） サーバは受け取ったデータを展開後、DNN の演算を行う

（4） 処理結果だけを末端デバイスへ転送する

この一連の推論では、末端デバイスの演算は圧縮のみであり、DNN の推論はサーバ側で行っている。そのため、サーバへの計算負荷が集中し、サーバへ至る通信路の輻輳も問題となっている。提案する実行モデルは画像を入力とする、タスク別の DNN を分割実装し推論を行う。提案する実行モデルの概略を

図 2 に示す。提案するモデルでは *Hard-shared* までの処理をエッジで行い、その出力を圧縮しクラウドへと転送する。転送先の A、B、C では転送されたデータを展開し、それぞれのタスクに対する後段の処理を行う。共有 CNN を利用することで、タスク別に用意された特徴量を抽出する CNN が 1 つになる。(タスク数 $\times$*Hard-shared*) の計算量削減と (対象のタスク $-$*Hard-shared*) のメモリ使用量が削減され、より多くの計算資源の乏しいエッジ向けハードウェアでの実行も可能となる。また、提案するモデルでは複数タスクに対するモデル全体の学習を行わなわず、タスク別に *Hard-shared* された共有 CNN を用いた転移学習を行うため、タスクの追加に対するネットワーク全体の再学習等を行う必要がなく、不特定多数のユーザーに対して特徴量の転送をすることが可能である。

提案する共有 CNN を用いたマルチタスク推論のモデルでは、共有 CNN 部をハードパラメーターシェアリングによって共有するため、あらかじめ共有 CNN 部を含んだモデルを学習させる必要がある。本研究では、共有 CNN 部の学習モデルに VGG16、MobileNetV2 の二つのモデルを採用し、物体認識に使用する ILSVRC2012 のデータセットを利用した学習を行い、学習したパラメータを他のタスクにハードパラメーターシェアリングする。共有 CNN を使用したモデルでは、タスクによらず共通の入力画像を利用するため、画像の大きさを揃える必要がある。本研究では、共有 CNN を学習する際の ILSVRC で用いられることの多い (224,224,3) のテンソルになるように画像を縮小する。入力画像の大きさを物体認識のタスクに都合良く定めたため、物体検出のタスクではモデル構造の変更が求められる。共有 CNN に利用するモデルの分割点であるが、分割点の出力がエッジからクラウドへ転送されるデータとなる。中間データは入力画像を転送するよりデータサイズが小さく、高効率に転送することを目標とするため、本研究では、入力画像よりデータサイズが小さく、また共有 CNN から出力される単一のテンソルのみ転送する。VGG16 の各レイヤーにおける出力要素数を図 3 に示す。入力画像より要素数が小さくなるのは pool4 の出力以降であるため、pool4 以降の出力を中間データとして伝搬させることが望ましい。MobileNetV2 の描くレイヤーにおける出力要素数を図 4 に示す。反転残差構造では、構造内でチャネルの拡大を行うため一時的にデータ量が増える。また、箇所によって反転残差構造への入力を保存する必要があるため、反転残差構造ごとの出力を伝搬させることを考え、図 4 では Bottleneck と示した。入力画像より要素数が小さくなるのは Bottleneck2 の出力以降であるため、Bottleneck2 以降の出力を中間データとして伝搬させることが望ましい。単一のテンソルのみを転送する場合、物体検出のモデルである SSD は中間の feature map を最終層へ伝搬させる必要があるため、転送するテンソルが最終層に伝搬される最大の feature map となる。共有 CNN に VGG16 を用いる場合、SSD のネットワーク構造は変更せずに伝搬する feature map を Conv4_3 の出力から pool4 の出力に変更する。共有 CNN に MobileNetV2 を用いる場合、Bottleneck14 の出力を SSD の Conv5_1 に伝搬するようにネットワーク構造を変更し、伝搬する feature map を Conv4_3 の出力から Bottleneck7 に変更する。

共有 CNN を利用したマルチタスク推論処理における全体の計算量を見ると、タスク毎に個別のモデルを設けた場合、全体の MAC 計算量 $MAC(models)$ は式 1 になる。続いて、共有ニューラルネットワークを用いる場合を述べる。まず、あるモデルの分割時におけるモデルの MAC 計算量 $MAC(model)$ は式 2 で表せられる。$model_{former}$ と $model_{latter}$ は、それぞれモデルをある境界で分割した場合において、そのモデルの前段と後段にあたるニューラルネットワークである。特徴量抽出機の役割を持つ前段 $model_{former}$ を共有畳み込みニューラルネットワーク CNN_{Hard_shared} で置き換えたならば、全体の MAC

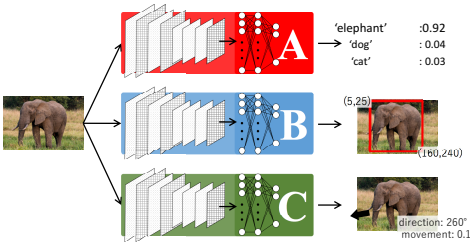

図 1: 一つの入力データに対する複数タスクの推論実行

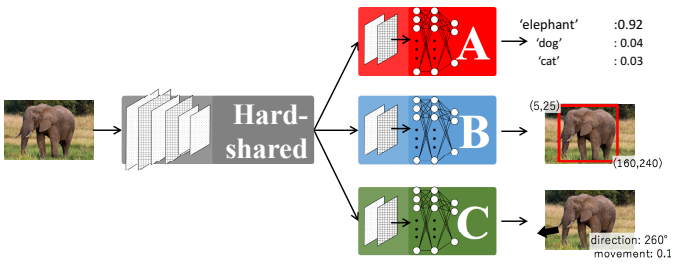

図 2: 共有 CNN を利用した複数タスクにおける推論実行

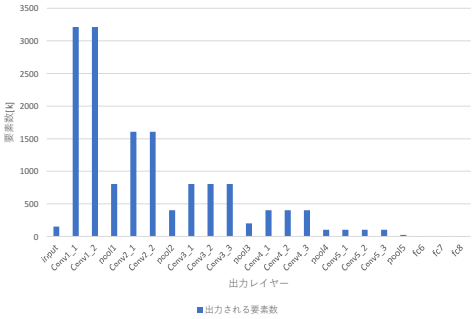

図 3: 1000 クラス分類のための VGG16 の出力要素数

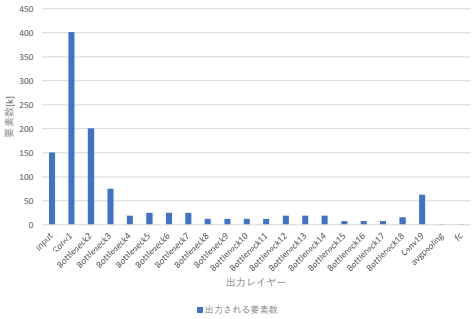

図 4: 1000 クラス分類のための MobileNetV2 の出力要素数

計算量は式 3 になる。

$$MAC(models) = \sum_{k=1}^{n} MAC(model_k) \quad (1)$$

$$MAC(model) = MAC(model_{former}) + MAC(model_{latter}) \quad (2)$$

$$MAC(models) = MAC(CNN_{Hard\_shared}) + \sum_{k=1}^{n} MAC(model_{k,latter}) \quad (3)$$

よって、前段のニューラルネットワークを共有 CNN で置換したモデル全体では式 4 だけ計算量削減が可能になる。一方、モデルの分割境界により、削減できる計算量と各モデルの精度はトレードオフの関係にある。なぜなら、CNN 本来の役割の特徴量の抽出だが、CNN の層が深くなるにつれパラメータは学習したデータセットに特化するためである。そのため、分割境界は分割時における各タスクの精度が、共有ニューラルネットワークを利用しないモデルと比較してどれだけ低下しているかは調査する必要がある。

$$diff = MAC(CNN_{Hard\_shared}) - \sum_{k=1}^{n} MAC(model_{k,latter}) \quad (4)$$

エッジとクラウドでの分割推論実行では中間データの転送を行うが、通常のクライアントサーバーアプリケーションと同様に転送するデータを圧縮することが考えられる。画像を入力としたときの VGG16 の pool4 の出力と MobileNetV2 の Bottleneck7 の出力を可視化したものを図 5 に示す。VGG16 の構造では活性化関数に ReLU を利用しているため、出力される中間データは 0 が多く含まれ、圧縮方法によっては圧縮率を高める可能性がある。一方で MobileNetV2 では反転残差構造を採用しているため、VGG16 の中間データよりもスパースな構造ではないため、画像圧縮を行う上では VGG16 の中間データよりも圧縮率が高くなる可能性がある。深層学習の中間出力である特徴量マップに最適化された圧縮手法は現在無いため元データ量が多く、圧縮率向上について多く研究されている動画像の圧縮手法を利用する。動画像の圧縮には可逆圧縮と非可逆圧縮がある。名称のとおり、可逆圧縮は圧縮後にデータの欠損はないが、非可逆圧縮では圧縮手法や圧縮時に指定するパラメーターごとにデータの欠損具合と圧縮率が変化する。本研究では非可逆圧縮のアルゴリズムとして H.265/HEVC [7] を主に使用する。HEVC とは動画圧縮に使用されるコーデックであり、画像圧縮に応用をしたのが BPG [12] である。別の画像の非可逆圧縮のアルゴリズムとして WebP [13] を利用する。

4. 実　　験

異なるタスクに対して一部に共通の CNN を用いた深層学習モデルを学習させた場合の精度を調査する実験を行う。タスクには物体認識と物体検出の二つを対象とする。物体認識は ILSVRC2012 で利用された 1000 クラス分類の画像のデータセットを利用し、物体検出には PASCAL VOC の 2007 年と 2012 年の画像のデータセットを組み合わせたものを利用する。共有 CNN に利用するモデルは物体認識のタスクにおいてモデル構造を考える上で転機となったモデルである VGG16、MobileNetV2 を採用する。共有に使用するモデルは ILSVRC2012 で利用されたデータセットで学習したものを利用した。共有する箇所は、VGG16 は入力から 4 つ目の Max pooling の処理後までを共有し、出力された (512,14,14) のテンソルをタスク別モデルに伝搬させる。MobileNetV2 は入力から 6 つ目の反転残差構造までを共有し、出力された (32,28,28) のテンソルをタスク別モデルに伝搬させる。モデル別の共通の前処理は、VGG16 のときは式 5 で示すように入力データを (224,224,3) に縮小したあと、平均値を引き、255 で割ることによって 0-1 量子化を行なった。MobileNetV2 のときは式 6 で示すように入力データを (224,224,3) に縮小したあと、128 で割り、1 を引いた。共有 CNN の学習時には、初期値を学習済みモデルにより決定する。VGG16 は Caffe の学習済みモデルを利用し、MobileNetV2 は Tensorflow の学習済みモデルのパラメーターを初期値として利用した。物体認識では fine-tuning を行い、物体検出では転移学習を行なっているため、学習時のパラメーターは表 1 に示すとおり、最適化関数には MomentumSGD を利用した。

表 1: 各モデルの学習環境

共有 CNN	VGG16		MobileNetV2	
タスク別モデル	VGG16	SSD224	MobileNetV2	SSD224
最適化関数	MomentumSGD	MomentumSGD	MomentumSGD	MomentumSGD
学習率	0.01	0.001	0.01	0.001
WeightDecay	0.1	0.1	0.1	0.1
バッチサイズ	128	32	128	32

$$preprocessedData = \frac{resize(input) - mean}{255.} \quad (5)$$

$$preprocessedData = \frac{resize(input)}{128.} - 1. \quad (6)$$

次に共有 CNN の出力を圧縮する手法について述べる。本実験で行うエッジークラウド間の中間データの圧縮展開フローを以下に示す。

（1） （エッジ）中間データのテンソルを RGB 画像のフォーマットのテンソルへと変換する

（2） （エッジ）データ単位が unsigned int8 になるように Min-Max 量子化を行う

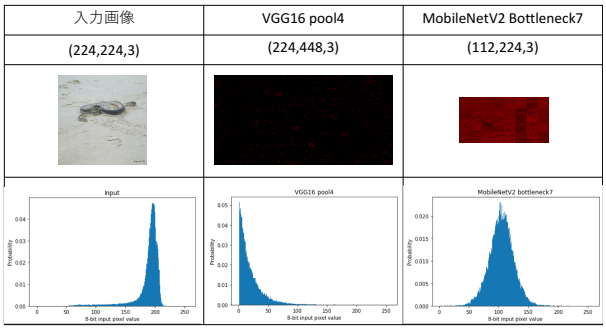

図 5: 入力画像のピクセル値のヒストグラム

表 2: 中間データのテンソルの変換

モデル	変換前 (Height, Width, Channel)	変換後 (Height, Width, Channel)
VGG16	(512,14,14)	(224,448,3)
MobileNetV2	(32,28,28)	(112,224,3)

（3） （エッジ）量子化されたデータの画像圧縮を行う

（4） 圧縮されたデータと Min-Max 量子化で使用された最大値、最小値を転送する

（5） （クラウド）圧縮されたデータを展開する

（6） （クラウド）最小値、最大値をもとに Min-Max 逆量子化を行う

（7） （クラウド）始めに中間データのテンソルを変換した手法の逆変換を行う

深層学習の推論の途中で生成されるテンソルは画像のように (Height, Width, 3 Channel) のテンソルで表されていないため、画像圧縮をするために変換を行う必要がある。本研究では、表 2 に示すように中間データごとの大きさの画像を用意し、中間データのチャネルごとの二次元のテンソルを画像の Red Channel にタイリングを行い、Green Channel と Blue Channel を 0 でパディングをした。深層学習の推論では浮動小数点 32bit が使用されることが多く、本実験でも使用している。深層学習の推論中の中間データのような特徴量を圧縮する専門の手法はないため、本研究では画像圧縮に用いられる非可逆圧縮の BPG、WebP を用いた圧縮を行う。画像圧縮を行う際にはデータ単位を unsigned int8 または unsigned int16 への量子化を行う必要があり、本実験では式 7 で示す Min-Max 量子化と式 8 で示す Min-Max 逆量子化を採用している。Min-Max 逆量子化をクラウドで使用するためには圧縮されたデータの最大値、最小値が必要となるため、転送の際には圧縮されたデータと共に最大値、最小値も転送する必要がある。

$$\tilde{V} = round\left(\frac{V - min(V)}{max(V) - min(V)} \cdot (2^n - 1)\right) \tag{7}$$

$$\hat{V} = min + \tilde{V} \cdot \frac{max - min}{(2^n - 1)} \tag{8}$$

5. 実験結果

共有 CNN を利用することにより、提案するモデル全体の計算量は VGG16 では 42.31%、MobileNetV2 では 9.77%の削減を行われる。次に推論の中間データの圧縮を評価するために、入力データとのデータサイズの比較を行う。ILSVRC2012 と PASCAL VOC2007 のテストデータは JPEG で提供されているため比較するために (224,224,3) に縮小をする。中間データは共有 CNN の出力に量子化とタイリングを行なったあとのデータを PNG 圧縮したものを示す。入力画像、量子化後の中間データのデータサイズを表 3 に示す。入力画像と PNG 圧縮した中間データのデータサイズを比較すると、PNG 圧縮した中間データは VGG16、MobileNetV2 ともに入力画像よりも大きいことがわかる。モデルの精度の劣化について述べるが、中間データは PNG 生成過程で量子化を行うため、精度の変化が予想される。量子化後の認識率を表 5 に示す。量子化前の認識率である表 4 と比較をすると、全てのタスク、モデルに組み合わせにおいて最大 0.02%の精度の劣化が見られた。次に ILSVRC2012 のデータセットに対して PNG 圧縮された中間データを BPG 圧縮した後の圧縮率、認識率を表 6 に示し、WebP 圧縮した後の圧縮率、認識率を表 6 に示す。次に PASCAL VOC2007 のデータセットに対して PNG 圧縮された中間データを BPG 圧縮した後の圧縮率、認識率を表 8 に示し、WebP 圧縮した後の圧縮率、認識率を表 9 に示す。非可逆圧縮に BPG を使用した場合の精度と WebP を使用した場合の精度を比較する。BPG を qp35 で使用したときに、物体認識の VGG16 の中間データは 63.81%、MobileNetV2 の中間データは 31.30%の圧縮率を記録している。WebP の qp80 では VGG16 の中間データでは 89.22%、MobileNetV2 の中間データでは 46.55%を示す。物体検出では、BPG が qp35 のとき VGG16 の中間データは 32.66%、16.63%の圧縮率を示した。一方で WebP が qp80 のとき VGG16 の中間データは 46.34%、24.72%の圧縮率を示している。物体認識、物体検出ともに同程度の圧縮率の場合に BPG を利用した非可逆圧縮を行うほうが VGG16、MobileNetV2 ともに精度が良い。

これらの結果から、CNN の中間データの特徴量を非可逆圧縮するとき画像の非可逆圧縮でより良い結果を出したアルゴリズムは中間データの非可逆圧縮においても良い圧縮ができて

表 3: 各層の出力データサイズ [Mbytes]

データセット	input[JPEG]	VGG16-conv4-3[PNG]	MobileNetV2-conv7[PNG]
ILSVRC2012	485.44	1742.48	1016.01
PASCAL VOC2007	89.47	158.05	100.31

表 4: 共有モデルの認識結果

データセット	ILSVRC2012		PASCAL VOC2007
共有モデル	Top-1	Top-5	mAP
VGG16	70.50	89.22	57.03
MobileNetV2	67.86	88.05	61.04

表 5: 共有モデルの Min-Max 量子化後の認識結果

データセット	ILSVRC2012		PASCAL VOC2007
共有モデル	Top-1	Top-5	mAP
VGG16	70.52	89.22	57.07
MobileNetV2	67.84	88.05	61.07

表 6: 物体認識の中間データに対する BPG 圧縮

量子化	VGG16			MobileNetV2		
パラメータ	圧縮率	Top1	Top5	圧縮率	Top1	Top5
qp10	485.23	70.52	89.24	177.80	67.85	88.05
qp20	298.11	70.48	89.20	117.62	67.77	87.99
qp30	131.60	69.98	89.04	61.00	67.16	87.63
qp35	63.81	68.39	88.18	31.30	65.27	86.43
qp40	19.92	57.93	80.94	9.80	49.99	74.41
qp50	1.30	0.99	2.63	1.08	0.73	2.00

表 7: 物体認識の中間データ対する WebP 圧縮

量子化	VGG16			MobileNetV2		
パラメータ	圧縮率	Top1	Top5	圧縮率	Top1	Top5
qp10	7.71	10.37	22.07	4.46	1.294	3.40
qp20	12.84	26.57	48.47	7.47	3.96	9.74
qp30	19.55	40.61	65.60	11.09	11.48	24.86
qp40	26.53	48.75	73.61	15.01	22.60	43.48
qp50	34.91	54.51	78.36	19.62	33.62	57.72
qp60	44.74	58.37	81.25	24.77	41.03	66.39
qp70	57.04	61.09	83.23	31.22	46.40	71.84
qp80	89.22	64.38	85.42	46.55	51.78	76.59
qp90	189.87	66.52	86.90	84.43	54.87	79.05

表 8: 物体検出の中間データに対する BPG 圧縮

量子化	VGG16		MobileNetV2	
パラメータ	圧縮率	mAP	圧縮率	mAP
qp10	233.82	57.16	95.21	61.09
qp20	144.58	57.14	62.90	61.07
qp30	65.17	56.29	32.54	60.59
qp35	32.66	54.37	16.63	59.39
qp40	11.12	42.54	5.19	50.62
qp50	0.73	1.39	0.61	2.37

表 9: 物体検出の中間データに対する WebP 圧縮

量子化	VGG16		MobileNetV2	
パラメータ	圧縮率	mAP	圧縮率	mAP
qp10	4.53	6.80	2.42	3.95
qp20	7.29	12.69	4.01	11.80
qp30	11.08	21.99	5.92	22.32
qp40	14.91	29.77	7.99	33.39
qp50	19.21	36.16	10.43	41.62
qp60	24.37	40.64	13.15	46.35
qp70	30.25	43.76	16.57	50.08
qp80	46.34	47.17	24.72	53.53
qp90	95.82	50.75	44.94	55.12

いることがわかる。非可逆圧縮を行なったときの精度の劣化だが、ILSVRC2012 のテストデータセットを用いた 1000 クラスの物体認識タスクに対して BPG 圧縮を用いた推論の分割と中間データの非可逆圧縮では、qp35 の場合に VGG16 は Top1 で 68.39%、Top5 で 88.18%の認識率を示し、非圧縮の結果と比較して Top1 で 2.11%の精度の劣化を確認した。MobileNetV2 は Top1 で 65.27%、Top5 で 86.43%の認識率を示し、非圧縮の結果と比べて、Top1 で 2.59%の精度の劣化を確認した。BPG 圧縮を用いた推論の分割と中間データの非可逆圧縮では、qp35 の場合に VGG16 は 54.37%の mAP を示し、MobileNetV2 は 59.39%の mAP を示した。非圧縮の結果と比較すると、VGG16 は 2.70%の劣化、MobileNetV2 は 1.68%の劣化を示した。

6. おわりに

本稿では、一対多のエッジコンピューティングにおける共有 CNN を用いたマルチタスクに対する深層学習モデルの推論の分割を提案した。分割推論時に中間データの圧縮には動画像に用いられる非可逆圧縮を提案し、中間データの劣化程度によるモデルの認識精度の変化についてそれぞれ実験を行った。MobileNetV2 を利用した場合に 9.77%の全体計算量の削減をした。ILSVRC2012 のテストデータセットを用いた 1000 クラスの物体認識タスクに対して BPG 圧縮を用いた推論の分割と中間データの非可逆圧縮では、qp35 の場合に Top1 で 65.27%、Top5 で 86.43%の認識率を示し、非圧縮の精度と比較すると Top1 について 2.59%の精度の劣化を確認した。圧縮率では、縮小済みの元画像に対して 31.3%の圧縮率を確認した。PASCAL VOC2007 のテストデータセットを用いた 20 クラスの物体検出タスクに対して BPG 圧縮を用いた推論の分割と中間データの非可逆圧縮では、qp35 の場合に 59.39%の mAP を示した。実験条件は異なるが、SSD の研究と比較すると 14.91%の精度の劣化を確認し, 圧縮率では圧縮済みの元画像に対して 16.63%の圧縮率を確認した。これらの結果から共有 CNN を用いたネットワークの分割と中間データの非可逆圧縮により、クラウドへの計算負荷の集中、転送データサイズの削減を精度のトレードオフとともに達成した。今後の展望として、本実験で示した物体認識で用いられる画像の前処理、モデル構造ではなく、物体検出や関節点推定等の他のタスクの精度に影響しない統一的な前処理たモデル構造の提案が挙げられる。

文　献

[1] 総務省，“情報通信白書”.

[2] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” CoRR, abs/1411.1792, 2014.

[3] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T.N. Mudge, J. Mars, and L. Tang, “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,” ASPLOS, 615–629, ACM, 2017.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, abs/1409.1556, 2014.

[5] H. Choi and I.V. Baji, “Near-lossless deep feature compression for collaborative intelligence,” IEEE MMSP’18, Vancouver, BC, 2018.

[6] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” CoRR, abs/1612.08242, 2016.

[7] G.J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (hevc) standard,” IEEE Transactions on circuits and systems for video technology, vol.22, no.12, pp.1649–1668, 2012.

[8] S. Kornblith, J. Shlens, and Q.V. Le, “Do better imagenet models transfer better?,” CoRR, abs/1805.08974, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[10] 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,” CoRR, abs/1704.04861, 2017.

[11] M. Sandler, A.G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” CoRR, abs/1801.04381, 2018.

[12] “https://bellard.org/bpg/”.

[13] “https://developers.google.com/speed/webp/”.