NAIST-IS-MT1651091

修士論文

エッジコンピューティングにおける
共有ニューラルネットワークを用いた
高効率な深層学習の推論実行モデル

福岡 久和

2018年2月1日

奈良先端科学技術大学院大学
情報科学研究科

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

福岡 久和

審査委員：

中島 康彦 教授	（主指導教員）
井上 美智子 教授	（副指導教員）
中田 尚 准教授	（副指導教員）
Tran Thi Hong 助教	（副指導教員）
張任遠 助教	（副指導教員）

エッジコンピューティングにおける 共有ニューラルネットワークを用いた 高効率な深層学習の推論実行モデル*

福岡 久和

内容梗概

飛躍的な進歩を続けるニューラルネットワークは、実アプリケーションとして組み込まれるようになった。推論処理に必要な計算量は大きいため、データ取得場所付近ではなく、データセンタに設置された潤沢な計算基盤をもつサーバで処理される。しかし、サーバやネットワークスイッチの膨大な消費電力やデータの一極集中によるトラフィック輻輳が問題となる。そこで、DNN を分割し、前半部分をエッジデバイスを用いて処理を行い、出力される中間データである特徴量を圧縮する分散推論モデルと、共有ニューラルネットワークを用いたマルチ推論モデルの 2 つ実行モデルを提案した。分散推論モデルでは、特徴量圧縮によるクラス識別問題の認識率低下を調べ、推論にかかる時間の見積もりを行った。結果、AlexNet を pool5 を境界に分割した場合、非圧縮の認識率 75.6%と比較して 3.4%の低下に抑え、従来の推論処理と比較して実行時間を 14.3%の増加により、トラフィック輻輳やサーバへの負荷集中を解消できた。また、マルチ推論モデルでは、共有ニューラルネットワークを用いることにより、1000、100 クラスの識別問題において、100 クラスの認識率は平均 12.3%の低下に抑えたまま、計算量を 50.3%だけ削減できた。

*奈良先端科学技術大学院大学 情報科学研究科 修士論文, NAIST-IS-MT1651091, 2018 年 2 月 1 日.

キーワード

深層ニューラルネットワーク、エッジコンピューティング、共通ニューラルネットワーク

An Efficient Execution Model for Inference Model of DNN using Shared-CNN in Edge Computing*

Hisakazu Fukuoka

Abstract

A neural network that continues to make drastic progress has come to be incorporated as an actual application. Since the amount of calculation of the inference process is enormous, it is processed by a server with abundant calculation resource, which is far from data source. However, traffic congestion due to concentration of servers, and power consumption of both servers and network switches data are problems. Therefore, the author constructs an efficient execution model forinference of distributed DNN using devices from the edge device to the base station, and propose feature map compression, which is intermediate data in DNN. In this paper, the author investigates the degradation of the recognition rate of the classification task by the feature map compression in the proposed divided DNN and estimate the time required for the inference, As a result, when AlexNet was divided after pool 5, the recognition rate is decreased by 3.4 % compared with that of uncompressed model. Compared to inference in the server, execution time was suppressed to 14.3% increase, and transfer to the server was eliminated. In the multi-inference model, by using the shared neural network, the computational cost can be reduced by 50.3%, while keeping the recognition rate of 100 classes on average by 12.3% in the 1000 and 100 classification task.

*Master's Thesis, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1651091, February 1, 2018.

Keywords:

Deep Neural Network, Edge-computing, Inference, Shared-CNN

目 次

図 目 次

表 目 次

1. はじめに

人々の日常生活に浸透し始めた人工知能であるが、実用段階まで性能を向上できた所以は計算量の大きいニューラルネットワークモデルの学習を実現するために、演算性能の高い計算機の利用や機械学習ソフトウェア群が整備され、高精度なニューラルネットワークの開発が可能になったからである。製品に組み込まれる検証済みモデルは、推論処理によって結果を出力する。従来では、1. 末端デバイスに接続されたセンサから取得した入力情報を圧縮し、2. ネットワーク通信路を介してサーバへ転送、3. サーバは入力情報を展開、4. 演算処理より結果を得る、という手順で推論処理が実行される。入力情報は数ホップを経て転送され、各種ワークロードに対応したサーバで演算されるが、それぞれネットワークスイッチ、サーバの消費電力が増大する問題となっている。同時に、末端デバイスからサーバが設置されているクラウドへデータ転送は一極集中であるため、トラフィック輻輳にも繋がる。そのため、上述のような DNN の推論処理によって生まれる社会問題を解決する手法が研究されている。

解決策の 1 つに、末端デバイスへの搭載を対象とする専用のアクセラレータが [9, 19, 29] が研究されている。他にも、推論処理であればデータの取りうる表現の幅を落としても精度の低下は小さいため、2 値論理を用いた推論処理 [20] や固定小数点を使用した手法 [17] が提案されている。

これらの努力より、効率的な推論処理が可能となる。クラウドへ演算を委託していたサーバ・クライアントモデルを脱し、単一デバイスだけで演算が可能となる。一方、現状では DNN 推論向けのチップは対象とするニューラルネットワークが限定されており、日々新しく提案されているアルゴリズムやソフトウェアによる効率化手法に追いつかない。また、専用チップの開発コストを考えるとスマートフォンやタブレットなどの高価な電子端末なへの搭載はできるが、エッジコンピューティングにおける安価なデバイスへ搭載は見送られることが考えられる。そのため、既存デバイスを組み合わせたシステムによる分散推論処理の実装が必要になる。

Deep Neural Network(DNN) の分散実装の 1 つである BranchyNet[24] では、クラス識別を行う単一のモデルを対象とした推論実行モデルが提案がされた。しか

し、文献 [14] では単一モデルだけでなく音声から文章への変換や、画像より物体を検出するモデルなど異なるタスクを行う各モデルを 1 つに統合したモデルが提案されている。近い将来、統合されたモデルが普及し、増加を続けるニューラルネットワークアプリケーションのワークロードに耐えるためには、ニューロンの発火をシミューレートする積和演算の効率化ももちろん必要ではあるが、計算量そのものを削減するような仕組みや分散実装における問題を調査することも必要である。

本稿では、計算機の消費電力やトラフィック輻輳を解決するために、エッジコンピューティングにおける高効率な深層学習の分散推論実行モデルを構築し、特徴量の圧縮、共有ニューラルネットワークを利用した分散実装を提案する。目的は、末端デバイスから基地局までのデバイスを用い、従来のクラウド依存の推論処理モデルから脱却することである。

本稿の構成を次に示す。第 2 章では、本論文の記述必要な用語の説明を行う。第 3 章に関連研究として DNN の分散実装を取り上げる。第 4 章では提案する複数モデルにおける推論の分散実装について述べる。第 5 章では、第 4 章で提案する画像特徴量の圧縮、共有 CNN を用いた推論に対して、それぞれ精度の低下を調べる実験について述べる。第 6 章は第 5 章の実験結果をまとめ、提案手法を適用したユースケースの一例を提示し、推論処理に掛かる実行時間を従来手法と比較し、その良し悪しを明らかにする。第 7 章では本稿をまとめあげ、今後の展望について述べる。

2. 諸定義

本章では本論文を述べる上で必要な用語を説明する。まず、提案する Deep Neural Network の高効率な分散推論の実行モデルにおいて、中心に取り扱うアルゴリズムである深層ニューラルネットワークについて 2.1 節で説明する。また、提案する共有ニューラルネットワークは畳み込みニューラルネットワークがベースとなるため、畳み込みニューラルネットワークの演算や役割について 2.2 節で説明する。さらに共有ニューラルネットワークでは、事前学習された重みパラメータや、事前学習されたニューラルネットを利用し、判別機だけの学習させるために転移学習を用いるため、事前学習や転移学習について 2.3 節で述べる。最後に、提案する DNN の分散実装は、サーバ群が設置されているクラウドやデータセンタではなく、データが取得された場所付近のエッジ側で行うため、エッジ側で計算を行うためのフレームワークやその考え方について、2.3 節で説明する。

2.1 深層ニューラルネットワーク

ニューラルネットワークは人間の脳を模倣し、計算機上で扱えるアルゴリズムの一種である。入力のパターンを覚え、その記憶したパターンに対応する答えを出力する。パターンはニューラルネットワークが持つ重みパラメータによって記憶

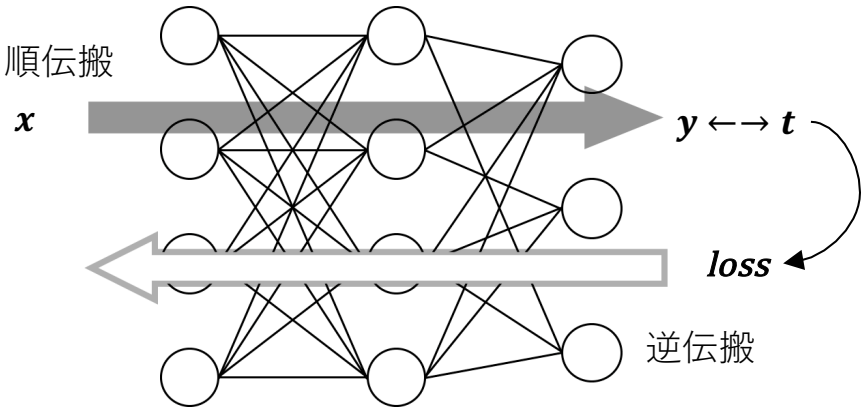

図 1: ニューラルネットワークの例

される。この入力に対する答えを覚えさせる処理は学習と呼ばれる。学習の手順を図 1 に示すニューラルネットワークの例を用いて説明する。例で紹介するニューラルネットワークでは、左から入力層、中間層、出力層と呼ばれるニューロンの集まりから構成されている。中間層は隠れ層とも呼ばれ、ニューラルネットワークの構成やニューロンの集まり層 (レイヤ) と呼ばれ、ニューラルネットワークの構成をニューラルネットワークのモデル、または単にモデルと呼ばれる。各ニューロンは前後の層のニューロン全てと接続しており、このような層を全結合層と呼ぶ。入力 x が与えられると、入力層から順に各層のもつ重みパラメータと各層の入力から各層の出力が算出され、最終的には出力層の結果 y が算出される。この入力 x から出力 y を導く演算を順伝搬という。そして、x に対応する正解データ t と出力 y の差分を誤差 $loss$ と呼び、出力層からこの誤差を各層が順伝搬において行った計算の微分から、勾配を求める。この誤差から各重みパラメータの勾配を求める計算を逆伝搬という。この逆伝搬から求めた勾配を用い、各層の持つ重みパラメータを更新することにより、ニューラルネットワークは入力パターンに対する正解データを記憶でき、学習を繰り返すことによりパターンの認識能力が向上する。しかし、モデルの学習には膨大な計算量を必要とする。そのため、計算基盤が整っていない時代では研究が活発ではなかったが、GPU といった演算性能の高い計算機の利用と機械学習ソフトウェア群が整備され、ニューラルネットワークの研究が進んだ。ニューラルネットワークは図中に示す中間層の数を増加させることにより、パターン認識の能力が向上する。ニューラルネットワークの層をより深くしたニューラルネットワークを深層ニューラルネットワーク (Deep Neural Network: DNN) と呼ぶ。しかし、層を深くすればその数だけ、学習や推論にかかる計算量やニューラルネットワークの重みパラメータが増大する。現在では、演算性能の高い計算機を利用することで、DNN の学習が数時間から数日で終えることができるため、ニューラルネットワークの研究が飛躍的に進んだ。これにより、パターン認識能力が向上し、音声から文章への変換、画像から写っている対象の検出などといった複雑な問題にも対処できるようになり、実アプリケーションに組み込まれている。

図 2 に示す画像を入力とする DNN は、前半に画像から特徴量を抽出する畳み

込み層、後半はその特徴量からパターンを認識する全結合層から構築される。入力画像が与えられると、その画像パターンに対して記憶した分類クラスの添字を出力する。例では、*Tiger* が写った画像が DNN の入力であり、演算結果は *Tiger* を示す添字 7 が出力されている。学習を終えたモデルはアプリケーションの一部に組み込まれ、順伝搬より結果を得る推論だけを行う。推論は学習と比較し、順伝搬だけだが、モバイル端末のようなサーバと比較して演算性能の低いデバイスでは畳み込みニューラルネットワークの計算量が大きいため、ほとんどの計算をサーバで肩代わりする利用が多い。しかし、サーバやネットワークルータなどの消費電力や、サーバへのデータ転送はトラフィック輻輳を引き起こし問題となっている。そのため、社会実装では、電力効率の良い専用回路や複数の計算機を利用した分散実装が研究されている。

2.2 畳み込みニューラルネットワーク

画像を入力とする DNN では、認識を役割をもつ全結合層への入力は画像の特徴量になる。そのため、その前段には特徴量抽出を行うニューラルネットワークが設置される。その画像の特徴量を抽出するニューラルネットワークが畳み込みニューラルネットワーク、すなわち CNN である。図 3 に示すように、CNN は入力が与えられた時、各畳み込み層が持つ重みパラメータと入力からある一定範囲、すなわち図中のカーネルと示された枠から取り出した値それぞれの掛け算の総和を求める演算、すなわち畳み込み演算より 1 つの出力が求められ、この畳込み

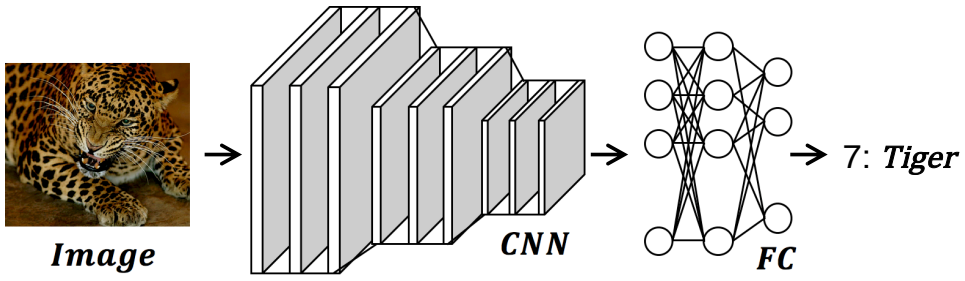

図 2: 画像を入力とする深層ニューラルネットワーク

演算を入力全てに対して適用させることにより、畳み込み層が出力する特徴量を得る。出力された特徴量は数層に重ねられた畳み込み層を通過することより、画像からより空間的な特徴が抽出される。しかし、特徴量の中には情報の少ない値もあるため、全ての特徴量を伝搬させると、その全てに対して畳み込み演算やパターン認識が行われる。しかし、小さな値を持つ特徴量は後続の層の出力へ与える影響は少ない。よって、全ての特徴量に対して演算を行うことは非効率であるため、出力された特徴量から代表値だけを伝搬させる役割を持つプーリング層がCNN の間にいくつか挿入されてる。このプーリング層により、ある一定サイズのフィルターを適用することにより代表値だけを伝搬できるため、後続のニューラルネットワークへの入力を小さくできる。CNN の計算量は、重みパラメータのサイズや入力画像、畳み込み演算層の入力範囲であるカーネルの大きさより決定される。重みパラメータの数が増えれば、もちろん重みパラメータのサイズも大きくなるが何よりも計算量の増大が大きい。

2.3 転移学習

ニューラルネットワークは層を深く重ねることにより、パラメータが増加し、パターン認識能力を向上できる。一方、学習時におけるパラメータ更新に必要な勾配は、微分から求められる。求められた勾配が 0 に近ければ、その勾配を逆伝

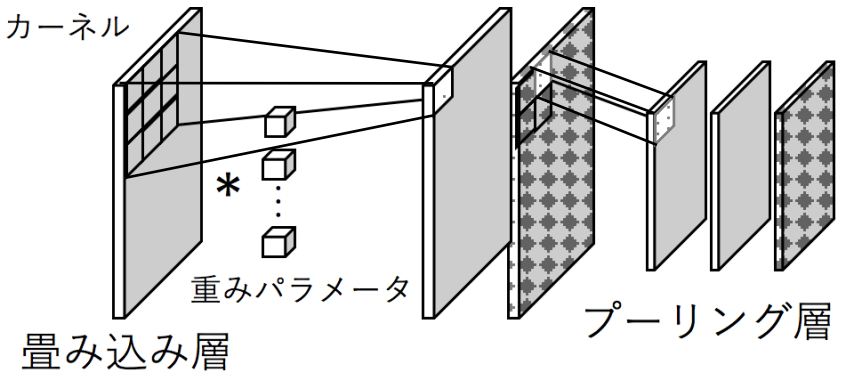

図 3: 畳み込みニューラルネットワーク

表 1: モデルの学習方法別の認識率の比較 [8]

scratch	pre-train	pre-train with fine-tuning
40.7	45.5	54.1

搬させてもパラメータの更新できなくなる。この勾配が 0 に近くづくことを勾配消失と呼び、勾配消失が発生するとモデルの学習ができない。そのため、勾配消失を防ぐために、AutoEncoder[12] 等のアルゴリズムによって学習時のパラメータの初期値を求めることを事前学習という。事前学習より求めた学習済みモデルを初期値とし、対象のデータセットを学習させることによって、与えられる初期値による学習のばらつきを軽減することができる。現在では、この問題に対してモデル中の中間層を変更するなど事前学習を利用しない解決策が発見されており、勾配消失の問題に対して事前学習を行うことは少なくなっている。現在では、対象タスクの学習用データセットがモデルを学習させるのに十分でない場合に事前学習が行われることがある。初期値を計算するために事前学習に異なるタスクのデータセットを用いた教師あり学習を行う手法を転移学習と呼んでいる。転移学習では、あるモデルが学習をするデータセットの内容やラベル数などの情報を包括したドメインという考え方がある。事前学習で学習したデータセットのラベルや内容が対象タスクのデータセットと異なる場合でもドメイン間の共通の特徴をモデルが学習しているため、異なるタスクのモデルの初期値やモデルの一部と入れ替えても性能を発揮することで知られている。図 4 は転移学習の流れである。豊富なデータセットを持つドメイン A で対象とするモデルの学習が行われる。学習を終えたモデルのパラメータを、実際に解くドメイン B のモデルの初期とする。そして、ドメイン B のデータセットより、モデルの再学習を行う。これにより、ドメイン B のデータセットの数が少なくても、十分な精度が得られるようになる。転移学習によって得られたモデルの一部を利用し、対象タスクのデータセットに対してモデルの残りのみを一から学習させることを fine-tuning と呼ばれている。タスクにより必要な出力数は異なる場合には、出力層やその前に位置する隠れ層が新たに設計し直される。既存研究ではモデルに DNN を採用し、

VOC2007[10] のデータセットを対象に、*fromscratch*、 *pre-train* 、*pre-train with fine-tuning* の三つの手法を用いてモデルを学習し、そのクラス識別の精度を比べている。*fromscratch* では、初期値をランダムにとり、対象のデータセットを利用した学習を行う。*pre-train* では、初期値をランダムにとり、ImageNet のみを使用した学習を行う。*pre-train with fine-tuning* では、事前学習として ImageNet のデータセットを学習した後、その学習で取得した値を初期値として、VOC2007 のデータセットを使用した学習を行っている。表 1 に示す結果では、*pre-train with fine-tuning* は、*fromscratch* の 40.7 %や *pre-train* の 45.5 %と比べて、54.1 %という最も良いパフォーマンスを示している。これらの結果から、DCNN は事前学習と転移学習の組み合わせによって効率的な学習と高い精度を両立することができる。

2.4 エッジ・コンピューティング

現在、ニューラルネットワークを処理する場所として、もっとも注目されている情報処理環境の一つとしては、クラウドコンピューティングがあげられる。

潤沢な計算基盤やネットワーク資源を擁し、かつその利用方法において、サーバーを各利用者が保持せず、その資源利用量に応じた料金の支払いが主流となっている。このような形式の恩恵として、初期投資費用の大幅な低下と、効率的な

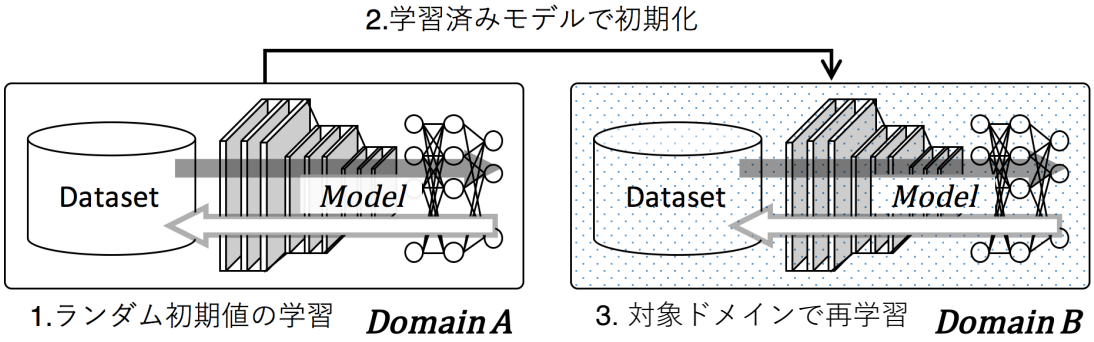

図 4: 転移学習

資源利用を行えば、ランニングコストの適正化を計れるようになった。次に、利用者が物理的資源を有しないことにより、従来の計算資源の固定的な構成から、要求仕様に合わせた動的な構成を組み上げることが可能になった。動的構成時、各クラウドコンピューティングベンダーが提供する、各種サービスを利用することが一般的であり、そのサービスはウェブ経由で提供される。これにより、全て論理的手続きのみで計算資源構成が可能となる。この結果、計算資源インフラストラクチャーが、論理的手続き、つまりコードで表現可能になったことを表す、'Infrastructure as Code' という言葉が発生することとなった。

上記のような、先進的サーバー利用形態を選択可能になるクラウドコンピューティングが市場に支持されることは自然な遷移であると考える。クラウドコンピューティングのような計算資源集中型は、歴史的には過去にも存在した。汎用コンピュータ創成期には、パーソナルコンピュータの能力は現在と比べ物にならない程低く、当時としては潤沢な計算資源場所へのアクセスを行い、実際の計算処理作業はそこで行われた。このように、コンピュータの、特に各個人が計算処理を行いたい場合の手法としては、歴史的には、計算資源集中型から出発している。この歴史の教えるところでは、計算資源集中型の次に遷移する状態としては、インターネットの誕生によって実現された資源分散型となる。各末端機器やそれに近い場所での計算資源の潤沢化が進み、データベースの集中化はあるが、計算資源の集中化は回避されるようになっていった。これらのネットワークを構成する各末端（エッジ）が協調動作することによって、現代に見られるような大規模ネットワークが実現している。

この歴史をなぞるように、大手ネットワーク機器ベンダー Cisco が、クラウドコンピューティング集中型からの脱却を目的とした、フォグコンピューティングというコンピューティングパラダイムを 2014 年に提唱した [2]。クラウドコンピューティングは、その物理的計算資源の配置は世界の各拠点であり、利用者からは、国単位のレベルでの距離をまたぐ必要がある。この物理的距離が大きくなるということは、ネットワークアプリケーションの実行時間に直結する。この時間そのものが問題になる場合、いくらクラウドコンピューティングが潤沢な計算資源を備えていても、問題の解決は不可能となる。また、計算資源集中によって、ネッ

トワークへの負荷が増大するケースが考えられ、より問題の解決を困難にする。それに対し、利用者に近接しているネットワーク周辺でのコンピューティングを示しているのが、このフォグコンピューティングである。物理的距離の近接度合いはそのままネットワーク遅延問題を解決する。フォグコンピューティングは、あらゆる計算資源を有するデバイスとの連携を想定している。もちろんクラウドもこの中に含まれる。このフォグコンピューティングの連携を実現する目的で、OpenFog コンソーシアムが創設された [4]。この創設メンバーには、ARM, Cisco, Dell, Intel, Microsoft, プリンストン大学エッジ研究所がある。

我々は、フォグコンピューティングに代表されるような、近接ネットワーク上における計算資源の利用をターゲットとする。 この近接ネットワークの範囲についてであるが、OpenFog コンソーシアムでは、エッジとフォグの環境を分けている節、つまり末端とそれが繋がる上位のネットワークを分けているが、我々は、クラウド以下の層、つまり末端から基地局やホームネットワークの層までをエッジとみなし、そのネットワーク間で目的のニューラルネットワーク処理を行わせることを、計算基盤の想定範囲とする。

3. 関連研究

分散処理を用いた推論処理の計算量を削減する手法として、文献 [16, 24] が挙げられる。

文献 [24] で提案されている BranchyNet では、クラス識別の問題を解くモデルに対して、複数の DNN を構築し、基準となる精度が満たされているならば、後段の DNN を使用せず、注目している DNN で計算された値をモデルの出力とし、全体の精度を保証する手法である。図 5 は、BranchyNet の構成図であり、入力から DNN の出力得られる出力層は複数あり、は exit と表記される。この例では 3 つの出力層がある。もし exit1 の出力だけで十分な精度が得られるならば、exit2

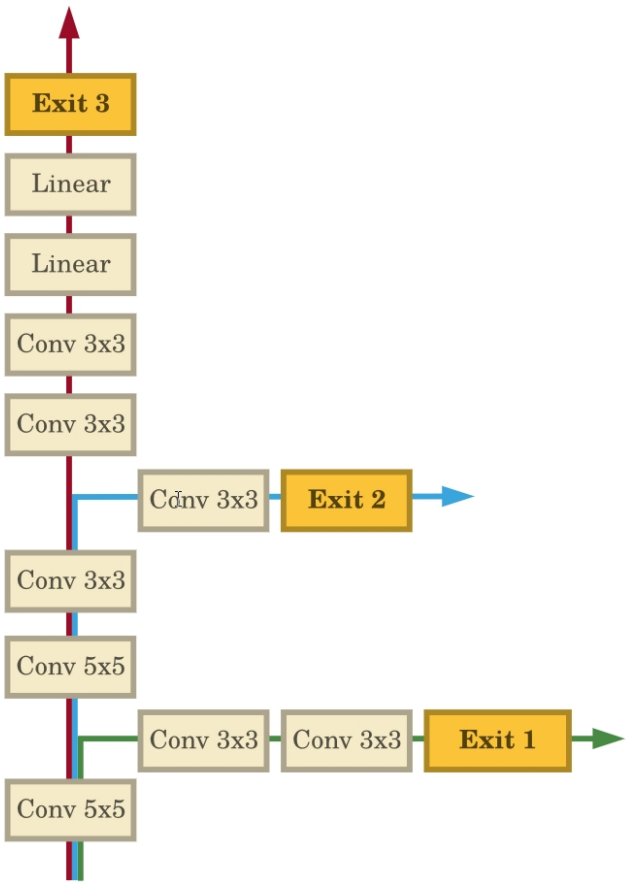

図 5: BranchyNet

を用いるDNNをを使用する必要がないため、残りの使用しないDNNだけ計算量が削減でき、また実行時間も短くなる。 一方、exit1では精度が足りないならば、続くexit2の結果を求める。exit2の結果でも満足できないならば、最終のDNNであるexit3へ転送し、搭載モデルの能力を全て発揮した精度を得ことが可能である。搭載したモデルや入力画像により、識別が容易であるかどうかは異なるため、BranchyNetは、識別する各クラス毎に精度を管理し、基準となる精度に応じて、精度が順分であるならば、控えているDNNを使わず、一方で求めた認識率が低いのであればより高い精度を求めるために後段のパスを用いるかどうかを判断する。その結果、全体の認識精度を基準値に保ったまま、識別が容易な画像に対して計算量や実行時間の削減できる。しかし、単一のDNNだけを対象とした分散手法であるため、複数モデルに対しては削減できる計算量が少なく効率が悪い。

文献[16]では、図6で示す2つの異なるタスクでCNNを共有するニューラルネットワークモデルが提案されている。提案されたモデルは1 つのCNNと、検出と回帰を行う全結合ニューラルネットワークの2つより構成される。CNNが出力する特徴量は後続2つの全結合ニューラルネットワークの入力となる。つまり、2つの全結合ニューラルネットワークの入力は共通している。提案されたモデルでは、異なる2 つのタスクを解くが、どちらも全結合ニューラルネットワークで構成されており、その入力が画像の特徴量であることに注目し、1つのCNNを共有している。 従来であればタスク毎にCNNと全結合ニューラルネットワークを組み合わせたモデルが必要になるが、提案されたモデルでは、共有するCNNの分だけ計算量や重みパラメータだけのメモリ使用量が削減可能になる。一方で、削減可能な計算量やメモリ使用量の議論はない。

4. 提案手法

本章では、エッジコンピューティングにおける複数のニューラルネットワークモデルの推論実行を想定し、DNN の分散実装を提案する。従来における DNN の推論処理は、図 7 に示すように次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

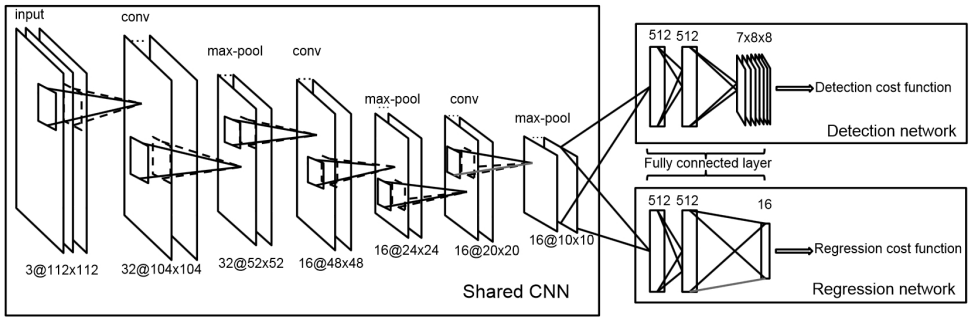

図 6: 共有 CNN を用いた異なるタスクを解くモデル

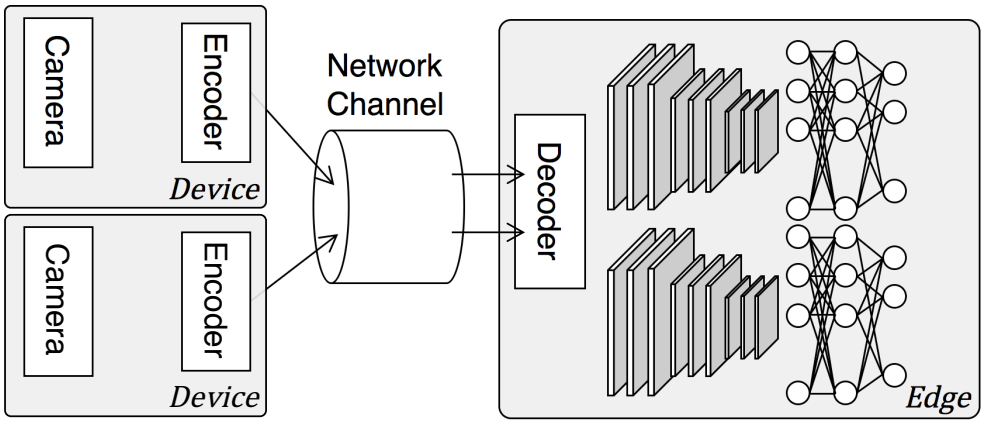

図 7: サーバを用いた DNN の推論処理

4. 処理結果だけを末端デバイスへ転送する

この一連の推論処理では、末端デバイスの演算は圧縮のみであり、DNN の推論処理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 [6]。さらに、文献 [14] では、音声から文字列へ変換、画像から物体の検出など異なるタスクのモデルを 1 つへ統合した、マルチタスクに対応したモデルが提案されている。ニューラルネットワークが進歩し、前述のようなマルチタスクに対応したモデルが普及すれば、従来型のサーバへ演算を委託する実行モデルは破綻する。この 2 つの問題に対象するために、DNN の分散推論処理の実行モデルを提案する。

提案する実行モデルは画像を入力とする、異なる DNN を分散実装し推論を行う。提案する実行モデルの概略を図 8 に示す。*Edge* はカメラセンサ *Camera* が搭載された末端デバイスである。各デバイスが画像を取得すると、そのデバイスに搭載された汎化性能のある共有ニューラルネットワーク CNN_{shared} より特徴量をが抽出される。出力された特徴量は各デバイスが持つハードウェアエンコーダ *Encoder* により圧縮され、通信路 *Network Channel* を通して基地局などの電源が確保できる場所に設置されたエッジサーバである *Server* へ転される。末端デバイス同様に転送された特徴量は、ハードウェア実装された専用回路 *Decoder* によっ

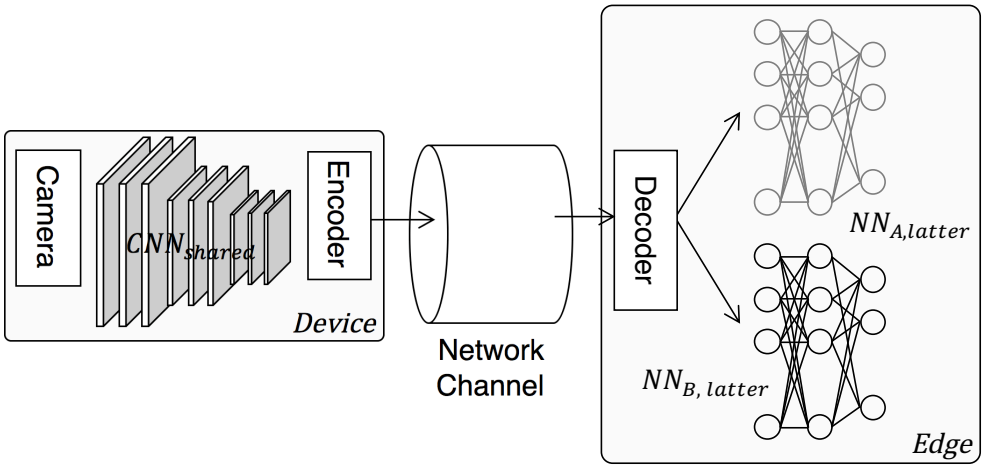

図 8: 共有 CNN を利用した複数モデルにおける推論実行

て展開後、各タスク処理を行うモデル後段である $NN_{A,latter}$ や $NN_{B,latter}$ の入力となり、モデル毎に各デバイスから得られた特徴量に対して結果を得る。

提案モデルを末端デバイスから基地局までのデバイス上に構築することにより、推論処理がエッジデバイス内で完結できる。そのため、サーバへの画像の転送がなくなり、サーバ側を膨大な処理から開放すると共に、サーバやスイッチの消費電力やトラフィック輻輳の問題を解決できる。また、共有ニューラルネットワークの搭載により、タスク別に用意された特徴量を抽出するニューラルネットワークが 1 つになり、計算量の削減やメモリ使用量の削減が可能となる。

以降では、提案する実行モデルにおける、画像特徴量の圧縮と共有ニューラルネットワークの詳細を順に述べる。

4.1 転送データ量を抑える特徴量データ圧縮

エッジ側ではサーバほど十分な計算資源を持たないデバイスを想定し、あらかじめ DNN を一定層の集まりで分割し、その層の集まりをデバイス毎に分散して演算する分散実装を想定する。図 9 に示す DNN モデルの 1 つである AlexNet を例に負荷分散を考える。図中に示された conv, pool, fc はそれぞれ、畳み込み層、プーリング層、全結合層を示し、その文字に続く数字は各層の添字である。input, output はそれぞれ入力と出力を示す。

例として、この AlexNet のモデルを 2 つのデバイスで分散実装することを考え

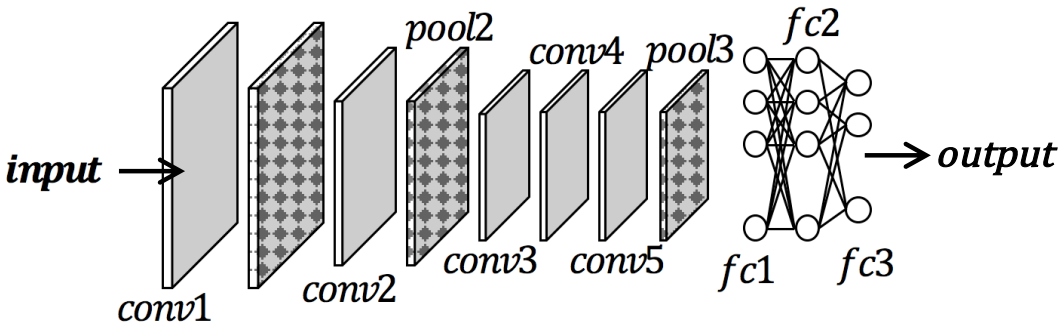

図 9: AlexNet のニューラルネットワークの構成

る。DNN を 2 つに分割すると、2 つのデバイス間でネットワークを介した通信が必要になる。転送サイズが大きければ、それだけネットワーク通信路を圧迫するため、転送サイズは小さい方が良い。図 10 は AlexNet における畳み込み層とプーリング層、全結合層の各層の出力数である。縦軸が各層の出力サイズ [KByte] であり、横軸は各層の名前である。input のデータサイズは、文献 [15] で記述されている入力画像 (RGB 各 8bit、高さと幅が 224×224) をもとに算出した。転送サイズの小さくなことだけを考えれば、fc2 で分割すれば良いが、共有ニューラルネットワークに全結合層が含まれる。共有ニューラルネットワークをを考慮すれば、パターン認識の役割を持つ全結合層を含むべきでない。そのため、分割境界は fc 以前の層が適切である。CNN の何処かで分割することになるが、図 10 の各層の出力数を見れば conv1 や conv2 の出力数が多く、通信路を圧迫するおそれがある。そのため、代表値だけを通過させるプーリング層の直後を分割境界にすることが好ましい。

一方、転送サイズだけでなく、分散 DNN における各デバイスにおいて実行される計算量も考慮する必要がある。図 11 は AlexNet における、畳み込み演算で使われる、乗算の結果を順次加算する演算、すなわち積和演算の計算量である。縦軸は積和演算の数 [M Operations]、横軸は各層の名前である。積和演算を用いる

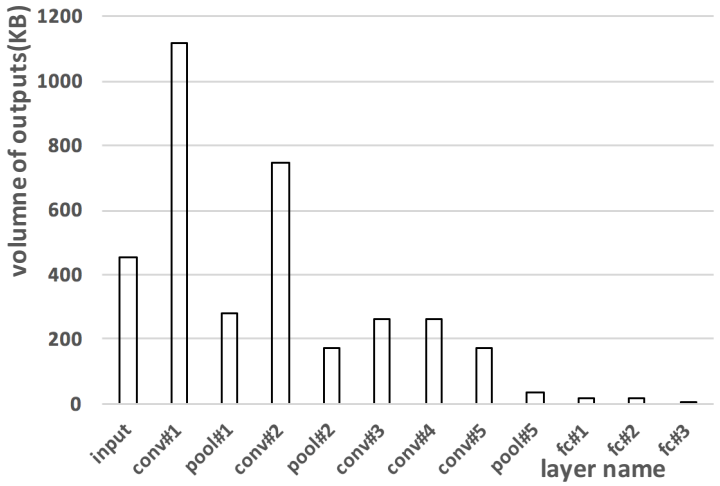

図 10: AlexNet における各レイヤのデータ出力数

ニューラルネットワークは畳み込み層と全結合層だけであるため、プーリング層は図にない。畳み込み層の計算量は大きく、モデルの後段側に位置する pool5 で分割すれば、conv1 から conv5 までの演算を 1 台のデバイス処理することになる。そのため、計算量に負荷に偏りができ、非力なエッジデバイスには過負荷となってしまう。一方、モデル全体に対して共有できるニューラルネットワークが占める割合が大きくなるため、複数モデルを考慮すれば共有できるニューラルネットワークが大きくなるため、その分だけ計算量を大きく削減できる。しかし、CNN は層が深くなるにつれ、単に画像の特徴量だけを出力するだけでなく、学習したデータセットに依存する特徴量を出力するとされている。そのため、共有するニューラルネットワークの割合が大きければ、各タスクの精度への影響も大きくなると考えられため、分割する境界は慎重に選ぶ必要がある。

Internet of Things が普及する現在において、実際の画像認識の応用を考えた場合、一般物体認識やクラス識別アプリケーションへの入力は 1 枚の独立した画像ではなく、カメラから連続的に得られる動画と考えることが自然である。コンテストで用いられるデータセットは、認識率を競うことが目的であり、多種多様な画像に対して認識や識別を行うため、それぞれ撮影条件が異なっている。単体の静止画を扱う応用として、投稿された画像を解析し、その画像を元にユーザ好

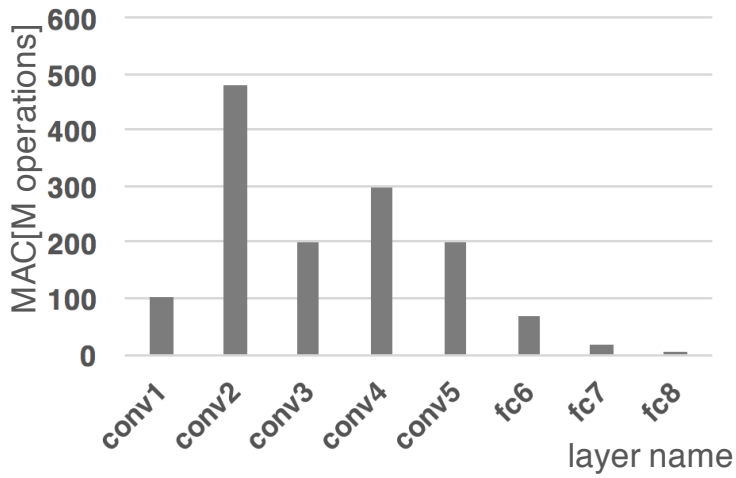

図 11: AlexNet における各レイヤの計算量

みの衣服を見つけ出してくれる PASHALY[5] や、手書き文字や伝票、領収書をスキャンし自動でテキスト化する AI inside[7] といったアプリケーションが存在するが、計算負荷を増大させるアプリケーションは、監視カメラや物体検査カメラから取得した動画を入力とするアプリケーションが多数を占めると想定される。

DNN の推論処理は、エッジ上の単一デバイスだけでの演算は重く、分割実装のためにはデバイス間のネットワーク通信が必須であり、そのオーバヘッドが問題となる。従来型実装であっても、センサから取得したデータをサーバへの送信する必要があるため、RGB 画像を圧縮したデータサイズよりも極端にデータサイズが増加すると通信コストの増大により性能のボトルネックになってしまう。一方、動画像データはデータ量が多いため生データをそのまま送信することは非効率的なため、圧縮して転送することが一般的である。これは従来型実装でも同様である。提案手法では元動画を圧縮するのではなく、中間データである画像の特徴量を圧縮する必要があるため、この圧縮率が元動画とどの程度違うのか調査することが重要である。図 9 よりプーリング層を境界として分割するのであれば、十分に実現可能性があるのではないかと考えた。中間データは原画像よりも縦横のサイズは縮小される一方で、深さ方向を表すチャンネルが増えるため圧縮の実装も注意深く検討する必要がある。

4.2 効率的な演算を図るニューラルネットワークの共有

提案する共有ニューラルネットワークついて述べる。文献 [16] では、異なる 2 つのタスクにおいて 1 つの CNN を共有し、後段にタスク別の判別機を設けている。モデル全体の学習はそれぞれの判別機から得られた誤差を共有 CNN まで逆伝搬させている。そのため、共有 CNN は 2 つの誤差を用いて学習されるため、その共有 CNN は 2 つのタスクに特化していると言える。提案手法では特徴量抽出機として汎化性能の高い重みパラメータを持ったニューラルネットワークを共有する。図 12 に示す例では、異なる 2 つのタスクを解くニューラルネットワークである。各タスクのデータセットはそれぞれ *Dataset A* と *Dataset B* である。黒の実線と破線の矢印はそれ *Dataset A*、*Dataset B* に属する入力データの流れを示す。共有ニューラルネットワークである CNN_{shared} は事前学習により得られたパ

ラメータを初期値する。図中の灰色で塗りつぶされた矢印は順伝搬を示し、灰色で囲まれた白抜きの矢印は逆伝搬を指す。そのため、共有ニューラルネットワークまで逆伝搬せず、順伝搬だけを行うため、重みパラメータは更新しない。一方、各タスク別の判別機である $NN_{A,latter}$ と $NN_{B,latter}$ で示すニューラルネットワークは逆伝搬まで行い、パラメータを更新させる。

複数の DNN モデル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、全体の計算量 $C(models)$ は式 1 になる。

$$C(models) = \sum_{k=1}^{n} C(model_k) \tag{1}$$

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計算量 $C(model)$ 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_{shared} で置き換えたならば、全体の計算量は式 3 になる。

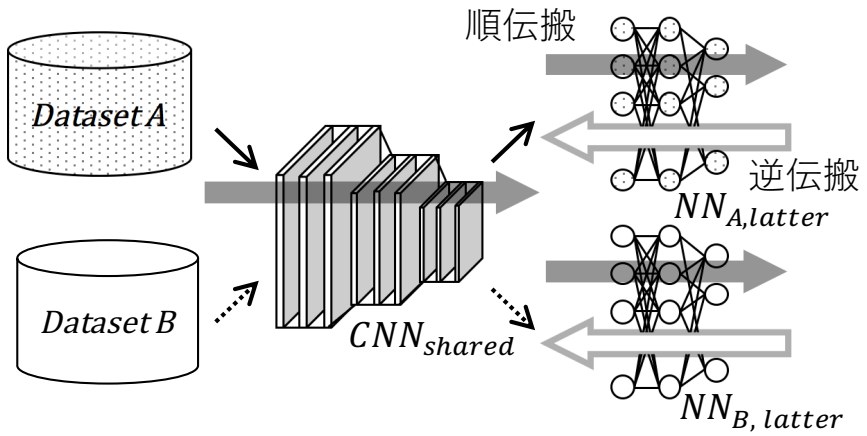

図 12: ニューラルネットワークの共有

$$C(model) = C(model_{former}) + C(model_{latter}) \tag{2}$$

$$C(models) = C(CNN_{shared}) + \sum_{k=1}^{n} C(model_{k,latter}) \tag{3}$$

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。

$$diff = C(CNN_{shared}) - \sum_{k=1}^{n} C(model_{k,latter}) \tag{4}$$

5. 実験

本章では、提案した分散 DNN の推論実行モデルにおいて、ネットワーク通信を考慮した特徴量圧縮と、共有ニューラルネットワーク手法の 2 つを独立して実験を行った。それぞれの実験について、続く 5.1 節と 5.2 節で述べる。

5.1 特徴量データ圧縮

DNN の中間データである特徴量に対して既存技術の圧縮手法が効果的かどうか調べ、認識率との関係を調査する。

DNN モデルには ILSVRC2017 のクラス識別用データセット [21] から 200*epoch* だけ学習した AlexNet[15] を用いる。学習には機械学習フレームワークである chainer[25] の Version.1.23 を用い実験した。重みパラメータやバイアス、入出力のデータタイプは単精度浮動小数点数である。

モデルを前段ニューラルネットワークと後段ニューラルネットワークの 2 つに分割し、分割する境界はプーリング層いずれかの直後とする。

例として、15 ページの図 9 において、*pool5* の出力を境界としたモデル分割を考える。前段は *input* から *pool5* まで、後段は *fc6* から *output* までの構成となり、*pool5* の出力が圧縮され、展開された出力が *fc6* の入力となる。圧縮において、本論文では HEVC エンコード [23] を想定し、入力サイズと入力形式を満たすようにプーリング層の出力を変換する。*pool5* の出力はウィンドウサイズが 6×6、奥行

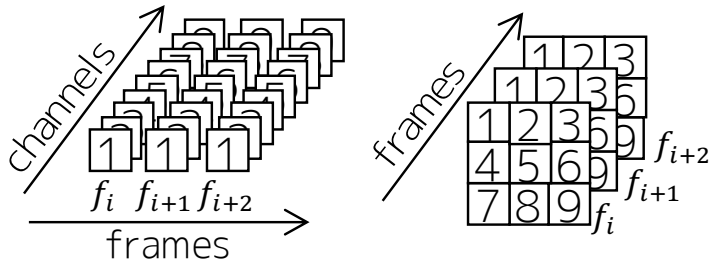

図 13: channel 出力の 1 枚フレーム画像への変換

きを示す channel の数が 256 である。1 フレームの出力を channel 別に独立した動画と考えると、6×6 の動画が 256 種類と捉えることができる。channel を時間軸と捉えると、6×6 の画像が 256 フレーム並んでいる集合の連続であると考えることができる。しかし、このどちらの方法であっても画像サイズが HEVC の符号化単位である Cording Tree Unit(CTU) の最小サイズ [28] である 16×16 を下回るため、HEVC 形式でのエンコードには不適切である。そこで、図 13 のように動画 1 フレームの出力をタイルのように敷き詰め、1 フレームが CTU の最小サイズを上回るように変換を行う。図中の番号が割り振られている正方形は 1 つの出力ウィンドウを示す。プーリング層の出力は図の左に示すように、channel は奥行、frames は時間を指す。そして、1 フレームの出力、すなわち 高さ $\times$ 幅 $\times$ 奥行き で示される出力において、高さ $\times$ 幅 の出力ウィンドウを左上から右下へなぞるようにタイル状に敷き詰め、1 枚のフレームに変換し、その各フレームを時間軸に並べた動画を HEVC の入力とする。*pool5* の場合、出力ウィンドウをタイル化した画像サイズは 96×96 となる。中間データである特徴量は畳み込み演算により入力画像の色情報がなくなっており、HEVC の入力は YUV 形式のみを受け付ける。そのため、特徴量を輝度情報とみなし、YUV400 形式として扱う。そして、プーリング層出力は浮動小数点数であるため、これを HEVC 規格のビット深度に対応する範囲に正規化した後、量子化する。YUV 形式へ変換する際はまず、特徴量を $[0, 1]$ の範囲で正規化し、ビット深度に応じた表現階調に写像する。ビット深度が 8 であれば、表現階調は 255 となる。正規化する上で必要な最小値と最大値は、レイヤ分割前のモデルよりデータセット毎に事前に測定した値を用いたが、これらの値を適応的に求める手法は今後の課題である。入力サイズは、CTU の最小サイズ、すなわち 16 の倍数でなければならないため、余りが生じるならばパディングする。プーリング層の出力を量子化した値を YUV 形式で表現した画像は、ほとんど特徴量のない値で占められている。図 14 は channel サイズが 27×27 である *pool1* の出力をタイル状に敷き詰め白黒反転した画像であり、反転後の最大値は黒色、最小値は白色である。図をみると、ほとんどが最小値を示す白色で埋まっており、特徴量の値が大きい黒色を示す部分は少ないことが分かる。似た色の数が多いほど圧縮が効くため、余白を埋める場合は白色を示す最小

値でパディングする。上述の手続きより、入力形式と入力サイズを満たしたプーリング層の出力を HEVC エンコードし、その出力をデコード、タイル状の動画出力をウィンドウサイズや channel 数などより復元し、*pool5* の次層である *fc6* に入力する。パディングした領域は HEVC 規格を満たすためだけに確保したため、復元時にその領域は破棄する。プーリング出力の圧縮 · 展開後には HM Software[3] の HEVC ソフトウェアを用い、基本的に圧縮設定は用意されたデフォルトを用いる。以上の手順より、デバイス間のデータ転送を想定した DNN の分割推論処理をシミュレーションする。

実験は入力の劣化具合に直接作用する量子化パラメータ (*Quantization Parameter:qp*) とモデルの分割境界を変更し、認識率と転送するデータ容量の関係を比較する。AlexNet の入力は 1000 クラス分類の正解ラベルとして登録されているアフリカゾウの群れがフレームの左側から右側へと移動する動画を用いた。フレーム数は 622 枚である。

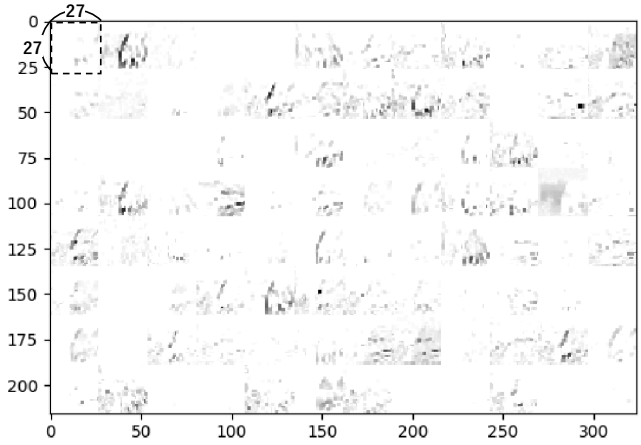

図 14: 出力をビット深度 8bit で量子化した画像

5.2 ニューラルネットワークモデルの共有

本節では、同タスクでの共通ニューラルネットワークを用いたDNNモデルの精度を調査する実験について述べる。対象タスクはクラス識別とし、分類クラス数100と1000のモデルを2つ用意する。2つのモデルを19ページの図12に示すように接続する。100クラス、1000クラス分類を行うモデルNN_AとNN_Bがそれぞれ与えられたとき、片方のNNからCNNを取り除く。もう片方のNNのCNNは共通のCNNであるCNN_{shared}とする。そして、共有CNNの以降のニューラルネットワークをそれぞれ$NN_{A,latter}$、$NN_{B,latter}$とする。NN_Aにおける推論は次の手順で処理される。NN_Aの学習データセット$Dataset\ A$から取り与えられた入力画像はCNN_{shared}で演算され、その出力は後段の判別機をの役割を持つ$NN_{A,latter}$入力となる。そして$NN_{A,latter}$の出力がNN_Aの推論結果となる。

学習用データセットはILSVRC2017[21]を用いる。1000クラス分類のモデルは学習済みモデルを用いる。共通CNNは学習済みモデルのCNNを用い、追加学習やパラメータ調整といった処理は加えない。100クラス分類のモデルは各モデルの文献を参考に初期値を定め学習を行う。データセットは1000クラスのモデル同様ILSVRCを用いるが、クラス分類が小さいため、100クラス学習のためにサブセットのデータセットは次の手順で作成する。

ILSVRCの元となるImageNetが木構造で構成されていることに着目しサブセットを作成する。ImageNetは図15のようにWordNetと呼ばれる英語の概念辞書に登録されている同義語の集合、synsetをノードとする木構造となっている。図中の*hypernym*のあるsynsetの上位語、*hyponym*はあるsynsetの下位語を示す。synsetの接続関係は次の例のように記述されている。*true cat*を取り上げると、その上位語は*Feline*であり、下位語には*wildcat*をもつ。また、*true cat*は同族語として*big cat*を持つ。*true cat*を上位語へ変換、同時に*big cat*も上位語への変換すると、両方とも*Feline*へ変換される。サブセット作成ではsynsetを上位語のsynsetに変換し、1000クラスの元のsynsetが別のsynsetと同一の先祖ノードを持つならば、ある1つを代表synsetとして残りのsynsetは削除する。代表synsetの数が目標数の$m$classに達するまで上位語への変換を繰り返す。上位語変換では上位語が1つとは限らず、複数個持つsynsetが存在する。このようなsynsetは複

数個の上位語に対して変換を行う。作成するサブセットは元データセット比較して、全学習用画像の数は少なくなる。しかし、1 クラスあたりの枚数は減少しないため、100 クラス分類の学習に影響はない。

学習した 100 クラスに分類するモデルの評価方法を考える。100 クラスは 1000 クラスのサブセットであるため、1000 クラスは 100 クラスを分類できる能力を持つ。そこで、1000 クラスの 100 クラス相当の分類精度を知る必要がある。サブセットのある 1 つのクラスは 1000 クラスのある 1 つ以上のクラスに対応する。図 15 を例に示すと、1000 クラスに属する *true cat* と *big cat* は 100 クラスでは *Feline* へと変換されるとする。そのため、100 クラスモデルでは、入力に *big cat* の画像が与えられたとき、*Feline* と推論されれば正解となる。そこで、1000 クラスでは *true cat* と推論されても、本来は不正解となるが、*true cat* の先祖ノードには *big cat* の先祖ノードと同じ *Feline* に存在するため、*true cat* も正解とする。

上述のように、1000 クラスを用いて 100 クラス相当の認識率を測定する場合、入力画像に 1 つ以上の正解ラベルが対応し、referenced pair とする。 referenced pair が 2 つ以上の正解ラベルを持つならば、100 クラスの入力画像は複数の正解ラベルから選ばれた 1 つの正解ラベルに対応する画像セットになる。複数の正解ラベルの画像セットすべてを 100 クラスのある 1 つの正解ラベルとする場合、各クラスごとに reference pair の正解ラベルの数が異なるため、上位語へ変換手順によってはサブセットに偏りができ 100 クラスモデルの学習に影響が及ぶが、本

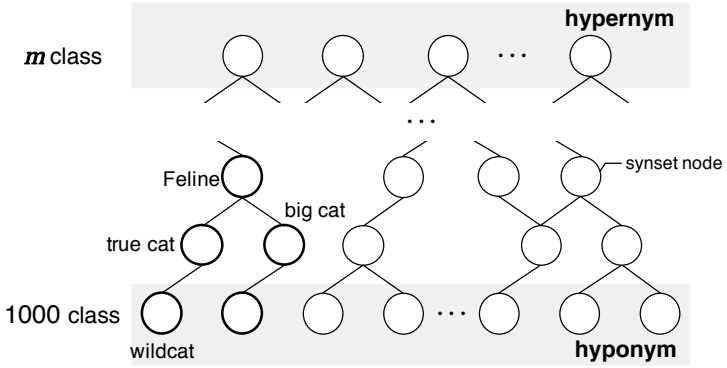

図 15: ImageNet: 同義語 synset をノードとする木

表 2: 各モデルの 100 クラス分類モデルの学習環境

	VGG16	ResNet50	GoogLeNet
最適化関数	AdaDelta[26]	MomentumSGD[27]	MomentumSGD
学習率	–	0.01	0.005
WeightDecay[18]	5×10^{-4}	1×10^{-4}	5×10^{-4}
バッチサイズ	18	18	12
epoch 数	100	40	30

実験では考慮せずサブセットは無作為に 100 クラス選択する。

作成したデータセットを用いて 100 クラスに画像を識別するニューラルネットワークの学習を行った。使用した機械学習フレームワークは 5.1 節と同じ chainer であるが、Version 3.2 を使用した。実験に使用したモデルは VGG16、ResNet50、GoogLeNet の 3 つであり、学習済みモデルはそれぞれ [22, 11, 1] を用いた。上述の 3 つの代表的なモデルは本来 1000 クラス識別用に設計されているが、本実験では基本的に出力層のニューロン数を 100 に削減したモデルを用いる。VGG16 に関しては、全結合層の直前に学習する加速させる役割を果たす Batch-normalization 層 [13] を挿入した。また、出力層より前の 2 つの全結合層のニューロン数を元実装の 2048 であったが、識別クラス数が 100 に減少し、1000 クラス分類に必要なパターン認識能力は不要であると判断し、数を 512 に減らした。詳細な各モデルの学習時の環境を表 2 にまとめる。100 クラスモデルの学習では、CNN は 1000 クラスのモデルと共有するため、CNN の学習は行わない。そのため、CNN 以降のニューラルネットワークだけ学習する、転移学習を行う。そのため、使用したモデル本来の学習条件と異なる。最適化関数は重みパラメータを更新するための関数であり、VGG16 では AdaDelta、ResNet50 と GoogLeNet は MomentumSGD を用いた。AdaDelta は学習率を持たないため、VGG16 の学習率は記載しない。WeightDecay は元の文献を参考に値を決めた。バッチサイズは、使用した GPU、1 台の GTX1080Ti で実行できよう値を変更した。epoch は学習時におけるデータセットを 1 順する単位であり、学習時に使用した値を記載した。

6. 実験結果

本章では、5 章で述べた実験を行った結果を示す。実験結果は 5 章の節に対応して記述する。

6.1 特徴量データ圧縮と認識率

学習したデータセット内にはゾウに分類されるラベルが *african elephant*、*indian elephant*, *tusker* の 3 つあり、各正解ラベルの認識率を調査した結果、top5 における認識率は *african elephant*、*indian elephant* 、*tusker* の順に 75.6%、60.8%、69.6%となった。最も高い認識率が得られたラベルは *african elephant* であったため、以降を動画の正解データを *african elephant* とする。ここで、得られた top5 の認識率は 75.6%と文献 [15] で示された認識率 84.7%と比較して 10%程低下している。今回の実験でも、文献 [15] で記述されたように、スケールダウン、切り出しを行いゾウが画面内に収まるように加工した。しかし、実験で用いた動画はゾウが右から左へと歩く姿を定点カメラにより撮影された映像であるため、図 16 に示す使用した動画の中には、左図のように認識対象であるゾウがフレーム内に収まる画像もあれば、右図のように反対に左からフレームインするゾウと右の画面

(a) inframe の例　　(b) outframe の例

図 16: 定点カメラにより撮影されたゾウ

外へフレームアウトするゾウが映し出されたフレームもあり、ゾウが見切れる場合があるため、識別が困難となり、全体として認識率が低下している。

得られた正解データより、プーリング層を境界に量子化、HEVC による圧縮した場合における認識率は図 17 になった。分割層は、その出力力が量子化や圧縮

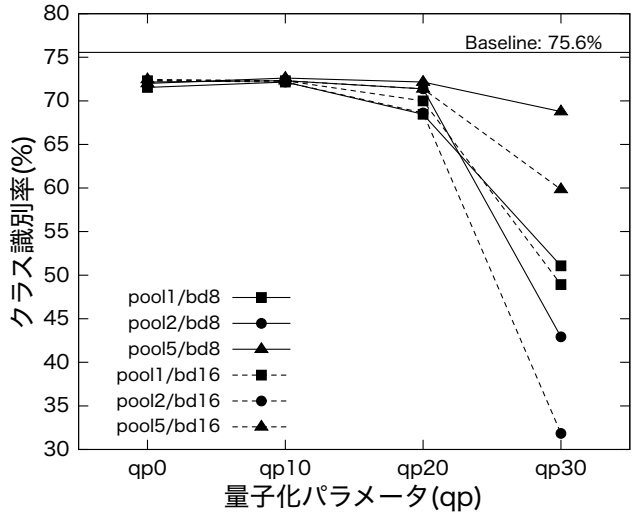

図 17: 量子化パラメータに対する認識率 (Top5)

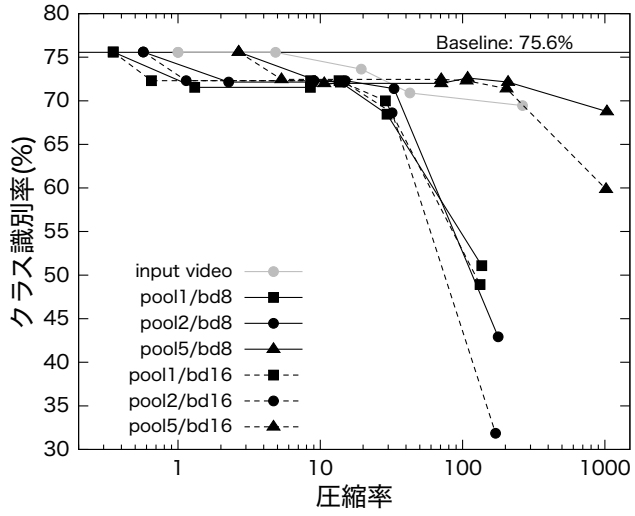

図 18: 圧縮比率に対する認識率 (Top5)

表 3: 各プーリング層出力の圧縮率

量子化	pool#1		pool#2		pool#5	
パラメータ	8bit	16bit	8bit	16bit	8bit	16bit
qp0	6.51	13.09	3.90	7.95	6.64	13.28
qp10	10.59	20.73	6.59	12.96	10.18	20.00
qp20	22.53	43.89	14.51	28.14	19.57	38.22
qp30	104.38	203.12	78.33	150.14	96.92	191.39

表 4: 各プーリング層出力の YUV 形式ファイルの容量

8bit[MByte]			16bit[MByte]		
pool1	pool2	pool5	pool1	pool2	pool5
46.81	26.91	5.73	93.63	53.82	11.46

処理される層を指す。図中上部に引かれた黒色の点線で示すベースラインである直線は量子化、HEVC による圧縮の両方を適用していない、すなわち変更を加えていない基準の認識率である。HEVC を適用する前に、分割層の出力は YUV400 形式 (グレースケール) に正規化する。量子化はビット深度が 8bit と 16bit の両方を用い、図中の *bd8* や *bd16* がそれぞれに対応する。qX は YUV 出力を HEVC でエンコードした結果である。qX の X は量子化パラメータ qp を指す。X が 0 の場合は損失のない可逆圧縮であり、識別精度の劣化は 8bit または 16bit の量子化だけによるものである。そして X の値が大きほど圧縮率が高くなる。図 17 において、10 まであれば認識率は低下しないが、以降の 20 や 30 になるにつれ低下しており、その振れ幅はより多くの精度をもつ 16bit の方が差が大きいことが伺える。

次に圧縮率と動画のデータサイズ、圧縮率に対する認識率を調査した結果、表 3、4 で示すようになった。YUV に量子化する前のサイズ、単精度浮動小数点数と比較した場合における圧縮率である。量子化の時点では、8bit と比較して 16bit の方が 2 倍の情報量を保持しているが、これに圧縮を適用すると、8bit と比較し

て 16bit の方が約 2 倍の圧縮率となり、8bit と 16bit の間に転送サイズの差はなくなっている。認識率を縦軸、表 3 の圧縮率を横軸にしたグラフを図 18 に示した。このグラフでは、約 100 までの圧縮率であれば量子化後の認識率と比較して低下が少ないことが言える。

本章では、提案手法の特徴量圧縮を適用した場合における推論に掛かる実行時間を見積もりを行い、従来手法と比較しその良し悪しを明らかにする。

想定する従来手法、提案手法における、推論処理の実行環境を図 19 より説明する。図中の *Edge Device*, *Edge Server*, *Sever* はそれぞれ、カメラといった画像センサを搭載する末端デバイス、電源が確保できる基地局に設置された小規模計算機、クラウドある各種ワークロードに耐えるようなサーバである。*Edge Network* は末端デバイスが接続される帯域の狭いネットワークであり、*Backbone Network* は基地局間を接続する帯域の広いネットワークを想定する。ここで扱う推論処理は 5.1 節と同様に、AlexNet モデルを使用した画像識別を行う。入力動画やその動画のフレーム数も同じ値を用いる。推論結果を転送するかは本考察では考慮しない。そのため、推論処理の完了は入力画像に対する識別されたクラスが出力された時点とし、その終了時点にかかるまでのレイテンシを推論に掛かる実行時間とする。また、提案手法を用いた認識精度の低下は図 18 で示す値を用いる。

従来手法であれば、図 19 の左に位置するエッジデバイスはカメラより画像を取得し、圧縮の前処理を行う。圧縮済みデータはエッジネットワーク、バックボーンネットワークを介して、サーバへ転送される。サーバは取得データを展開後、推論処理を実行する。一方、提案する分散 DNN を採用したならば、エッジデバイスでは、CNN で構成されるモデル前段部分が処理され，出力される特徴量を

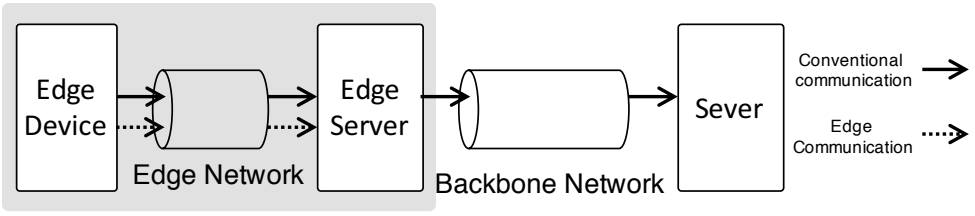

図 19: 従来手法と提案手法の推論処理の様子

表 5: 想定する環境下において使用するデバイスとネットワーク環境

(a) デバイス

Device	Performance[MFLOPS]
Mali G71	173.4
Jetson TX2	332.8
GTX1080	4113.9

(b) 通信路

Network	Throughput[KByte/sec]
10Gbit Ether	125000
Bluetooth Low Energy (BLE)	750

圧縮し、エッジサーバへ転送する。エッジサーバは圧縮済み特徴量を受け取り展開後、モデルの残り、後段部分を演算し、推論処理を実行する。

本章では、上述の推論処理の手続きは、表 5 で示すデバイスやネットワーク環境を想定し実行されるものとする。表の左側は、使用するデバイス名とその理論値性能である。理論値性能は畳み込み層や全結合層で用いられる積和演算の 1 秒あたりの演算回数、すなわち [FLOPS] を用いる。表の右側は、ネットワーク環境と、そのネットワークにおけるスループットの理論値である。単位は [KByte/sec] を用いる。図 19 の従来手法と提案手法における推論処理の実行環境に表 5 に示すデバイスやネットワークを当てはめると次のようになる。エッジデバイスではモバイル GPU の Mali G71、エッジサーバではサーバと比較して電源は確保されているが、大きな電力は確保できないことを想定し十数 W で動作する Jetson TX2 を用いる。サーバにはデスクトップ向け GPU である GTX1080Ti を使用する。ネットワーク環境は、バックボーンネットワークでは莫大な数の通信機器が接続された想定でも十分なスループットが確保できるように、本来であれば表で示した 10Gbit Ether を複数束ねたような帯域が確保されることが実際に即している。しかし、本章で取り上げる 256×256 サイズの動画像であればレイテンシの増加は小さいため、バックボーンネットワークは 10Gbit Ether での接続を仮定する。エッジネットワークは、低消費電力向けの Bluetooth Low Energy(BLE) を用いた。

従来手法と提案手法の推論処理において、表 5 に示すデバイスやネットワークを想定した環境での推論にかかる時間を表 6 に示す。表中の値は各処理や転送に

表 6: 各処理にかかる遅延時間とその合計時間

	モデル前段	エッジネットワーク転送	バックボーンネットワーク転送	モデル後段	合計時間
従来手法	0	4.202	0.003	0.207	4.412
提案手法	4.952	0.391	0	0.164	5.147
					[sec]

掛かるレイテンシを表し、単位は [sec] を用いた。一連の推論処理の中には、カメラから受け取った画素値 (RGB) を YUV へ変換や特徴量の正規化、量子化のような画像の前処理が含まれるが無視できる値であるため、表に載せない。圧縮や展開はハードウェアエンコーダで処理されることを想定する。従来手法と提案手法の両方で圧縮、展開が行われるが、画像の前処理と同様に実行時間は無視できる値ため表に含めない。両手法において、圧縮には 5.1 節と同様に HEVC を用い、量子パラメータは非圧縮の認識率 75.6%より 5%を下回らないような量子パラメータを採用する。そのため、量子パラメータは従来手法と提案手法それぞれ、10 と 20 を用いる。表中の従来手法のモデル前段と、提案手法のバックボーンネットワーク転送は、それぞれの推論処理に含まれないため 0[sec] になる。提案手法における分割境界は、pool5 を採用する。

以上の仮定や実験結果より、従来手法の転送サイズは 3151.44[KByte] となりエッジネットワークのレイテンシが 4.202[sec] となった。サーバの計算時間は、GPU の計算能力が高いため、0.207[sec] で演算処理が終わる。よって、従来手法における推論にかかる時間は合計 4.412[sec] になる。一方、提案手法では、モデル後段に近い層で分割し、またプーリング層を経て代表的な特徴量だけが残っているためこともあり、転送サイズは 292.96[KByte] になった。そのため、バックボーンネットワークと比較してネットワーク帯域の狭いエッジネットワークの通信路を用いても 0.391[sec] で転送できた。しかし、計算量の多い CNN とモバイル GPU の組み合わせだと、モデル前段の処理に 4.592[sec] 掛かる。よって、提案手法において推論時にかかる時間はモデル後段の処理に掛かる時間を合わせて合計 5.147[sec] となる。

結果として、特徴量の圧縮を用いた分散推論では、AlexNet を pool5 を境界に分割した場合、非圧縮の認識率 75.6%と比較して 3.4%の低下に抑え、従来の推論処理と比較して実行時間の増加を 14.3%に抑えることができた。これにより、バックボーンネットワークへの転送もなくすことができ、トラフィックの輻輳とサーバへの計算負荷の集中を解消できた。

6.2 ニューラルネットワークの共有と認識率

本節では、5.2 節の実験の評価を行う。100 クラス、1000 クラス分類の認識率を評価する。表 7 は、各モデルにおけるクラス分類の識別率である。表中の 1000class と 100class は 1 画像に対して 1 つの正解ラベルがある入力を与えたときの認識率である。reference-class100 は、100 クラスの認識率が妥当かどうかを 1000 クラスの推論結果より算出したものである。サブセットのデータセットは、元となる 1000 クラスの正解ラベルを上位語へ変換したラベルである。変換後のラベルは下位語の synset である 1000 クラスの正解ラベルを複数持つことが考えられる。この関係を利用し、変換後の上位語が同じであれば識別できたとする。例えば、ページ 25 の図 15 を参考にすると、入力画像 *true cat* が与えられ、識別結果が *big cat* であるならば、*true cat* と *big cat* は共通の親 *Feline* を持つため、正解となる。この手続きより、1000 クラスの認識率を読み替え referenced-100class の認識率を算出した。

100class の認識率は referenced-100class と比較して平均 12.31%低い。各文献で提案されたモデルは試行錯誤して学習した結果であり、十分な調整が加えられている。精度の良いモデルを得るためには、深層学習のより専門性の高い知識が必要なる。100 クラス分類を行うモデルは提案された構造を変更し作成したモデルであり、学習には変化した構造を考慮しなければならない。前述の 2 つより、100class の認識率は referenced-100class と比較して低下したが、学習ができていることを示す認識率が得られたと言える。よって、複数モデルにおいて、共有ニューラルネットワークを用いた推論処理が実現可能であることを示すことができた。

提案する共有ニューラルネットワークを用いれば、全体のメモリ使用量や計算量を削減できる。本実験で取り上げた各モデルにおいて、ある層を分割境界とす

表 7: 各モデルにおける 1000 クラス、100 クラス識別の認識率

Model	Accuracy(Top1)		
	1000class	referenced-100class	100class
VGG16	74.88%	81.76%	73.94%
Resnet50	78.26%	84.42%	65.30%
GoogLeNet	69.84%	77.10%	67.12%

表 8: 各モデルの分割時における前半、後半モデルのメモリ量と計算量

Model	分割境界	メモリ量 [MByte]			積和演算 [M Operations]		
		前段	後段	合計	前段	後段	合計
VGG16	conv3_5	58.9	494.5	553.4	15,346.6	123.6	15,470.3
Resnet50	res5/a	76.1	26.0	102.1	5,222.5	42.2	5,264.6
GoogLeNet	inception_5b	23.5	4.1	27.6	5,406.0	1.0	5,407.0

る場合において、メモリ使用量と計算量を表 8 に示す。表中の分割境界は共有される最後の層であり、その層以降は各モデルで独立したパラメータを持つ。メモリ量は、前段ニューラルネットワーク、後段ニューラルネットワークとその 2 つをあわせた元のモデルそれぞれに対して、重みパラメータの総数から使用されるメモリ量を算出したものである。重みパラメータのデータ型は単精度浮動小数点数とし算出した。積和演算の総和もメモリ量と同様に、前段、後段とその 2 つをあわせたモデルそれぞれに対して、重みパラメータのサイズや畳み込み層や全結合層への入力サイズなどから実行される積和演算の総数を求めた。分割境界の選出は、基本的に共有されるニューラルネットワークが最大となるように、識別の役割をもつ全結合層の 1 つ前とする。しかし、ResNet50 では、後段を全結合層だけにすると学習が進まなかったため、全結合層の直前に位置する畳み込み演算ブロック 1 つだけ、共有せず後段ニューラルネットワークに割り当て、ランダム初期値から学習した。

計算量に関しては、どのモデルにおいても前段は計算量の大きい畳み込み層で

構成されており、前段 CNN の計算量はモデル全体の内、平均して全体の 99%を占める。そのため、CNN が共有可能であれば、複数モデルを用いたならば、計算量を大きく削減できる。メモリ使用量に関しては、全結合層は畳み込み層と比較してパラメータ数が多いが、共有されるニューラルネットワークのほとんどは CNN である。しかし、ResNet50 や GoogLeNet の全結合層はシナプスの結合数が少なく、モデル全体の内、前段ニューラルネットワークの占める割合高いため、前段のメモリ量はモデル全体の内、それぞれ 74.5%、85.1%と高い。そのため、この 2 つのモデルであればメモリ量を大きく削減できる。一方、VGG16 のメモリ量を基準に考えると、ResNet50 と GoogLeNet の元々のメモリ使用量は少ないため、共有ニューラルネットワークによる削減できるメモリ使用量の効果は小さい。VGG16 に関しては、3 つの全結合層を後段に持つため、後段ニューラルネットワークの重みパラメータはモデル全体の内、89.4%となる。そのため、共有ニューラルネットワークを利用しても、削減可能なメモリ使用量は 10.6%と少ない。

結論として、100、1000 クラスの識別モデルにおいて、共有ニューラルネットワークを用いれば、100 クラスの認識率を平均 12.3%に抑えたまま、計算量を 50.3%だけ削減できた。

6.3 特徴量圧縮と共有ニューラルネットワーク

7. おわりに

本稿では、エッジコンピューティングにおける分散ニューラルネットワークモデルの推論実行において、画像特徴量の圧縮と共有ニューラルネットワークモデルを利用した実行モデルを提案した。そして、2 つの提案に対してそれぞれ実験を行った。

特徴量圧縮の実験では、AlexNet モデルにおいて、pool5 の出力を HEVC で圧縮することより、認識率を 3.4%に抑え、入力画像を圧縮した場合と比較して 10.8 倍だけ転送サイズを小さくできた。結果、特徴量圧縮によるクラス識別問題の認識率低下を調べ、推論にかかる時間の見積もりを行った。結果、AlexNet を pool5 を境界に分割した場合、非圧縮の認識率 75.6%と比較して 3.4%の低下に抑え、従来の推論処理と比較して実行時間を 15%の増加により、トラフィック輻輳やサーバへの負荷集中を解消できた。共有ニューラルネットワークモデルの実験では、2 つのクラス識別問題を用意し、GoogLeNet や ResNet50、VGG16 のモデルにおいて、認識率を調査した。共有ニューラルネットワークを用いることにより、 1000、100 クラスの識別問題において、100 クラスの認識率は平均 12.3%の低下に抑えたまま、計算量を 50.3%だけ削減できた。

今後の展望として、関節点推定や一般物体検出を扱うモデルを組み合わせた、共有ネットワークモデルの評価が挙げられる。また、本論文では提案手法を別々に評価したが、より現実的なアプリーケーションでの評価は必要であるため、特徴量圧縮とニューラルネットワークモデルの共有の両方を組み合わせた評価が挙げられる。

謝辞

研究活動から日常生活まで，多大なる御指導を頂いた本学の中島康彦教授に深く感謝の意を表します．本稿をご精読いただき，発表の場においても貴重なご意見を頂いた本学の井上美智子教授に深く感謝致します．日頃の研究活動におけるご指導はもちろん，学会での発表をはじめ様々なことに挑戦するチャンスを与え，また自主性を重んじ成長させてくださった本学の中田尚准教授に心から感謝申し上げます．また研究活動や本稿の執筆をはじめ多くのご助言を頂いた本学のTRAN Thi Hong 助教授に心より御礼申し上げます．報告の場や論文執筆の際，様々な視点から助言をくださり，視野を広げて下さった，また修士 1 年の短い間，多大なるご指導を頂いた 現北海道大学の高前田伸也准教授に深く感謝致します．研究活動から日常の生活に至るまで 2 年間をともに過ごし，また賑やかに支えてくれた同輩の山野龍祐氏，そして三谷 剛正氏， 平賀 由利亜氏， 木村睦氏， 一倉 孝宏氏， 上竹 規之氏， 菊谷 雄真氏，山根 弘樹氏， 吉田 怜矢氏 をはじめとするコンピューティング・アーキテクチャ研究室の皆様には感謝の念に堪えません．最後に，大学院への入学を喜んで認めてくださり，どんな時も暖かく見守ってくれた家族に感謝いたします。

参考文献

[1] Bair/bvlc googlenet model. https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet.

[2] Fog computing and the internet of things: Extend the cloud to where the things are. http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[3] High efficiency video coding(hevc) — jct-vc. https://hevc.hhi.fraunhofer.de.

[4] Openfog. https://www.openfogconsortium.org.

[5] イメージ画像で探せるaiショッピングアプリ、『pashaly（パシャリィ）』を刷新. http://www.scigineer.co.jp/news/20170814/2416/.

[6] 将来のネットワークインフラに関する研究会（第 4 回）配付資料. http://www.soumu.go.jp/main_sosiki/kenkyu/nwinfra/02kiban05_04000270.html.

[7] 人工知能搭載ocr「ai inside」とメディア型ecモール「kabukiペディア」が業務提携. https://inside.ai/news/14.html.

[8] Pulkit Agrawal, Ross B. Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. *CoRR*, abs/1407.1610, 2014.

[9] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ACM Sigplan Notices*, volume 49, pages 269–284. ACM, 2014.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007 (voc2007)

results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[14] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Sijin LI, Zhi-Qiang Liu, and Antoni B. Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[17] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. *CoRR*, abs/1511.06393, 2015.

[18] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[19] Tony Nowatzki, Vinay Gangadhar, Karthikeyan Sankaralingam, and Greg Wright. Domain specialization is generally unnecessary for accelerators. *IEEE Micro*, 37(3):40–50, 2017.

[20] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

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

[23] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[24] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *CoRR*, abs/1709.01686, 2017.

[25] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, volume 5, pages 1–6, 2015.

[26] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[27] Jian Zhang, Ioannis Mitliagkas, and Christopher Ré. Yellowfin and the art of momentum tuning. *CoRR*, abs/1706.03471, 2017.

[28] 大久保 榮 鈴木 輝彦 高村誠之 中條 健. *H.265/HEVC* **教科書**. インプレスジャパン, 2013.

[29] 中島康彦. Emaxv における複数バースト転送と複数ベクトル演算のオーバラップ手法. *CPSY*, 15:71–76, 2016.

業績

1. 福岡久和, 山野龍佑, 高前田伸也, 中島康彦: ”高位合成ツールを用いたFPGA並列コンピューティングの可能性検討”, 信学技報, CPSY2016-26, pp.181-186, Aug. (2016)

2. 一倉孝宏, 山野龍佑, 福岡久和, 中島康彦: ”DCNN に最適な CGRA の探索と予備評価”, 信学技報, vol.116, no.416, CPSY2016-114, pp.49-54, Jan. (2017)

3. 平賀由利亜, 三谷剛正, 福岡久和, 中田　尚, 中島康彦: ”エッジコンピューティングによる分散ニューラルネットワークの構想”, 信学技報, vol.117, no.44, CPSY2017-11, pp.62-67, May. (2017)

4. 福岡久和, 三谷剛正, 平賀由利亜, 中田尚, 中島康彦: ”動画圧縮技術を利用した分散機械学習における情報伝達効率化”, 信学技報, vol.117, no.153, CPSY2017-30, pp.151-155, Jul. (2017)

5. 福岡久和, 平賀由利亜, 三谷剛正, 中田尚, 中島康彦: ”分散 CNN における圧縮と集約による情報転送の最適化”, 信学技報, vol.117, no.314, CPSY2017-59, pp.51-54, Nov. (2017)

6. Takamasa Mitani, Hisakazu Fukuoka, Yuria Hiraga, Takashi Nakada and Yasuhiko Nakashima: ”Compression and aggregation for optimizing information transmission in distributed CNN”, CANDAR’17, REGULAR PAPER, pp.112-118, Nov. (2017)