ディープラーニングフレームワーク Caffe の分散環境への適用

一瀬　絢衣† 　竹房あつ子†† 　中田　秀基†† 　小口　正人†

† お茶の水女子大学　〒 112-8610 東京都文京区大塚 2-1-1
†† 産業技術総合研究所　〒 305-8560 茨城県つくば市梅園 1-1-1
E-mail: †{g1220504,oguchi}@is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp

あらまし　近年インターネット上の情報量の増大やクラウドコンピューティングの普及により，ライフログの取得やそのデータの蓄積が容易になった．その結果監視カメラなどのセンサを用いたライフログの利用も普及してきている．ここで，一般家庭にサーバやストレージを設置して解析までの処理を行うことは困難であるが，センサデータをそのまま映像を送信してクラウドで処理する場合，プライバシや各種センサとクラウド間のネットワーク帯域の問題が生じてしまう．本研究では，クラウドへ送るデータのデータ量の削減とプライバシの保護を目的とし，ディープラーニングのフレームワークである Caffe を分散環境へ適用させることを考える．本稿ではディープラーニング処理シーケンスを分割してクライアント側とクラウド側でパイプライン的に分散処理を行い，分割箇所やパラメータを変化させ処理時間を比較した．

キーワード　Caffe，ディープラーニング

Examination for the destributed processing of deep learning framework Caffe

Ayae ICHINOSE†, Atsuko TAKEFUSA††, Hidemoto NAKADA††, and Masato OGUCHI†

† Ochanomizu University　2-1-1 Otsuka, Bunkyouku Tokyo 112-8610 JAPAN
†† National Institute of Advanced Industrial Science and Technology (AIST) 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568 JAPAN
E-mail: †{g1220504,oguchi}@is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp

1. はじめに

近年インターネット上の情報量の増大やクラウドコンピューティングの普及により，ライフログの取得やそのデータの蓄積が容易になった．その結果監視カメラなどのセンサを用いたライフログの利用も普及してきている．ネットワークカメラと呼ばれる，サーバ機能を持つカメラなども比較的安価で手に入るようになり，防犯・監視用途や家庭でペットや子供の様子を遠隔地から見るという用途などで広まっている．ここで，一般家庭にサーバやストレージを設置して解析までの処理を行うことは困難であるため，クラウドへ送信して処理を行うのが一般的である．しかし，センサデータをクラウドへそのまま映像を送信する場合，連続的に大容量データを転送する必要があるためネットワーク帯域の問題が生じてしまう．また，一般消費者のライフログには個人の生活や行動に関する情報も含まれるため，プライバシの問題も生じてしまう．実際に，保育所や幼稚園，マンションなどに Web カメラを設置して子供の様子を自宅にあるコンピュータやスマートフォンで見るという例も増えているが，設定ミスなどの要因によりインターネット上で外部から見ることが可能となってしまうという危険があり問題となっている．

本研究では，クラウドへ送るデータのデータ量の削減とプライバシの保護を目的とし，ディープラーニングのフレームワークである Caffe を分散環境へ適用させることを考える．本稿ではディープラーニング処理シーケンスを分割してクライアント側とクラウド側でパイプライン的に分散処理を行い，分割箇所やパラメータを変化させ処理時間を比較した．

2. ディープラーニング

ディープラーニングとは，人間の脳の神経回路がもつ仕組み

を模した情報処理システムであるニューラルネットワークの中で，識別を行う中間層を多層化したものを用いた機械学習を指す．中間層が複数になっていることにより何段階かで認識を繰り返し，色や形状，全体像など複数の特徴を抽出してより正確な識別が可能となっている．現在，画像や音声の認識に広く使われている．

そのフレームワークである Caffe は，Convolutional Architecture for Fast Feature Embedding の略であり，Berkeley Vision and Learning Center が中心となって開発を進めている．ディープラーニングのフレームワークである [1]．Caffe は特定の機能を持つモジュールを組み合わせて構成されており，各モジュールが相互に通信することでシステム全体としての動作を決定している．これにより，新しいデータフォーマットやネットワーク層への拡張が容易にできる．また，C++で実装されており，既存の C++システムとインターフェースへの統合が可能であるという特徴がある．さらに，GPU に対応していることから高速な処理が可能である．また学習済みネットワークモデルが提供されているため誰でも簡単に実験を行うことができるという特徴がある．

Caffe は，ディープラーニングの中でも畳込みニューラルネットと呼ばれる構造になっている．畳込みニューラルネットは主に画像認識に応用されるネットワークであり，通常畳込みとプーリングという画像処理の基本的な演算を行う層がペアで複数回繰り返されたあと，隣接層間のユニットが全結合した全結合層が配置される構造になっている．畳込み層では入力画像に対しフィルタを適用し，フィルタをずらしながら各重なりで両者の積和計算を行うことによってフィルタが表す特徴的な濃淡構造を画像から抽出する．プーリング層では画像上で正方領域をとり，この中に含まれる画素値を使って一つの画素値を求め，畳込み層で抽出された特徴の位置感度を若干低下させる．

3. 分散ディープラーニングフレームワーク

本研究では，図 1 で示すフレームワークを提案，実装した．畳込みニューラルネットワーク処理シーケンスに新たな層を定義してネットワークを分離し，クライアント側，クラウド側を用いた分散処理を実現する．

クライアント側の Sink layer では伝搬されてきたデータをホスト名とポート番号でされたクラウド側の計算機へデータを流し，Source layer でデータを受け取って処理を継続する．ここで Source layer から Sink layer へ ACK が送られ，Sink layer は ACK を受け取ってから次のデータを流すようになっている．

分散処理を行うことにより映像や画像そのものでなく特徴量をクラウドに送るためプライバシが確保され，またデータ量を小さくしてから送ることによりネットワーク帯域の制限による性能劣化も軽減できる．

4. 実　　験

4.1 データセット

実験では，CIFAR-10 と STL-10 の 2 種類のデータセットを用いた．

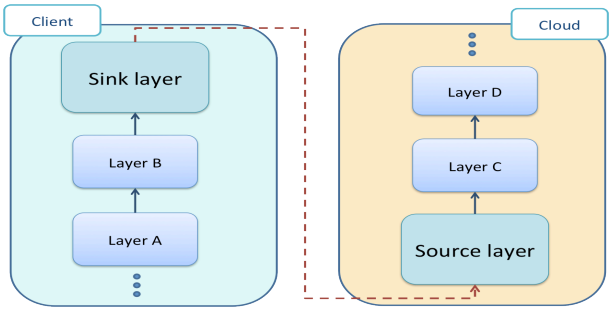

図 1　分散ディープラーニングフレームワーク

CIFAR-10 とは 32 × 32 画素の画像が 10 種類のクラスに分類されているデータセットであり，Caffe で学習済みネットワークモデルが提供されているデータセットの一つである [2]．このネットワークモデルにおいて各層で定義されているパラメータは表 1 のようになっている．

Caffe においてデータはバッチサイズ，チャネル数，画像のサイズの 4 次元配列で格納されており，チャネル数は直前の畳込み層におけるフィルタ数と一致する．提供されているネットワークモデルのデフォルトの値では，(100 × 3 × 32 × 32)byte のデータ量からフィルタ数 32 の conv1 層を通り (100 × 32 × 32 × 32)byte へ，ストライドを 2 に定義している pool1 層を通り (100 × 32 × 16 × 16)byte へと変化しており，pool3 層までは最初のデータ量よりも大きくなっていることがわかる (図 2)．

STL-10 は 64 × 64 画素の画像が 10 種類のクラスに分類されているデータセットである [3]．本研究では CIFAR-10 形式に変換し，CIFAR-10 のネットワークモデルを使って識別を行った．

表 1　パラメータ

n : num_output	フィルタ数
p : pad	パディングの幅
k : kernel_size	フィルタの大きさ
s : stride	フィルタの適用位置の間隔

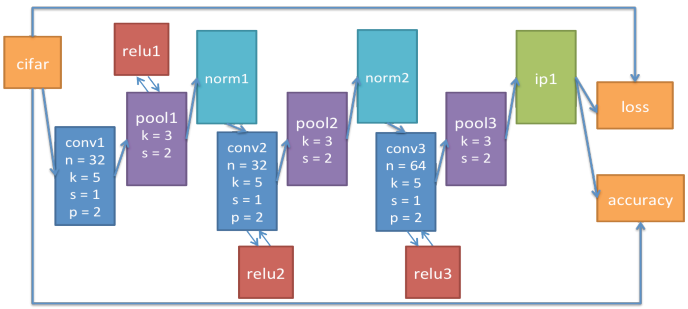

図 2　cifar10 識別用ネットワークモデル

4.2 分散方法

本稿では分散方法として，以下の 2 つを検討した．

- 分散方法 1

pool1 層，norm1 層間でネットワークを分離する．(図 3) conv1 層のフィルタ数を変化させることにより通信時のデータ量を削減させることを考える．conv2 層，conv3 層のフィルタ数はそ

れぞれデフォルト値の 32，64 に固定し，conv1 層のフィルタ数を 1 からデフォルト値の 32 まで変化させた．フィルタ数と通信時のデータ量の対応は表 2 に示す．

- 分散方法 2

pool2 層，norm2 層間でネットワークを分離する．(図 4) conv2 層のフィルタ数を変化させることにより通信時のデータ量を削減させることを考える．conv1 層，conv3 層のフィルタ数はそれぞれデフォルト値の 32，64 に固定し，conv1 層のフィルタ数を 1 からデフォルト値の 32 まで変化させた．フィルタ数と通信時のデータ量の対応は表 2 に示す．

表 2　conv1 層と conv2 層のフィルタ数と通信時のデータ量 (KB)

フィルタ数	1	4	8	12	16	20	24	28	32
conv1	25.6	102.4	204.8	307.2	409.6	512.0	614.4	716.8	819.2
conv2	6.4	25.6	51.2	76.8	102.4	128.0	153.6	179.2	204.8

4.3　異なるフィルタ数における識別率の評価

分散方法 1，2 について，データ量を小さくすると識別率が低くなってしまうことが考えられるため，フィルタ数を変化させた際の識別率の変化を調査した．実験環境は表 3 に示す．

表 3　実 験 環 境

OS	Ubuntu 14.04LTS
CPU	Intel(R) Xeon(R) CPU W5590 @3.33GHz (8 コア) × 2 ソケット
Memory	8Gbyte
GPGPU	NVIDIA GeForce GTX 980

分散方法 1 の結果を図 5，分散方法 2 の結果を図 6 に示す．CIFAR-10，STL-10 ともに少ないデータ量で識別率が収束していることが確認でき，識別率はデフォルト値では CIFAR-10 は 0.7811，STL-10 は 0.6187 であるのに対し，分散方法 1 では，フィルタ数が 4 の場合 CIFAR-10 は 0.7335，STL-10 は 0.5748 を計測した．フィルタ数 4 では画像をそのまま送信する場合と比較して 3 分の 1 のデータ量となっている．分散方法 2 では，フィルタ数が 4 の場合 CIFAR-10 は 0.7335，STL-10 は 0.5664 を計測した．フィルタ数 4 では識別率を大幅に下げないまま通信時のデータ量はさらに小さく 12 分の 1 となっていおり，フィルタ数を削減することにより通信時のデータ量を削減しても高い識別率を保てるということがわかった．

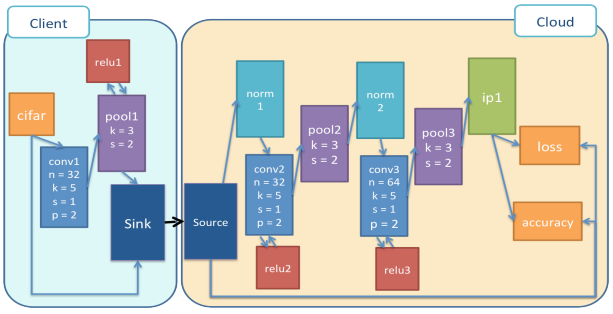

図 3　分散方法 1

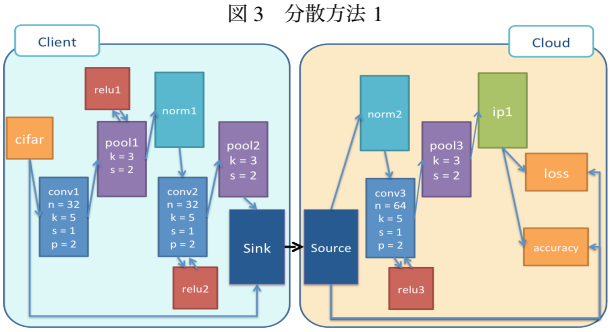

図 4　分散方法 2

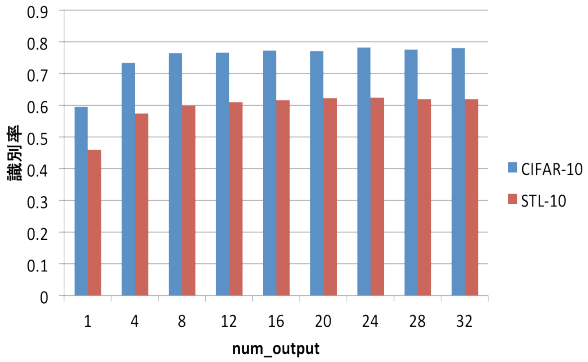

図 5　conv1 層のフィルタ数を変化させた場合の識別率

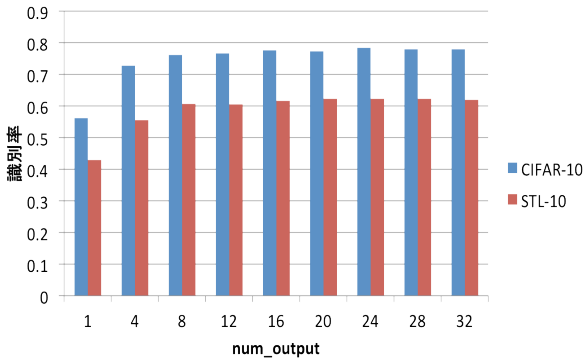

図 6　conv2 層のフィルタ数を変化させた場合の識別率

4.4　クライアント側，クラウド側で分散処理を行った場合の処理時間の評価

提案手法の評価として，クライアント側，クラウド側として 2 つの端末を利用し，畳込み層のフィルタ数を変化させて 1 バッチの処理時間を計測した．クライアント側で全ての処理を行う場合，提案手法により処理を分散させる場合，データをそのまま送信してクラウドで処理を行う場合の 3 つの処理時間を比較した．処理を分散させる場合では，クライアント側とクラウド側で処理が同期して同時に動いているため，通信時間や待ち時間を含んだクライアント側の処理時間を結果に用いた．いずれの実験においても，クライアント側では処理をすべて CPU のみで行い，クラウド側では GPU を利用した．実験環境は表 3 と同じであり，2 つの端末間のネットワーク帯域は 1Gbps である．また，通信時間のみを 100 倍にすることにより，実際のネットワーク帯域が狭い環境を模擬した調査も行った．ここで通信時間とはクライアント側からクラウド側へデータを渡し，クラウド側からの ACK を読み出すまでの時間を示す．

分散方法 1 を想定し計測した結果を図 7，その通信時間を 100 倍にしたものを図 8，分散方法 2 を想定し計測した結果を図 9，その通信時間を 100 倍にしたものを図 10 に示す．

分散方法 1 においては，分散処理 (Distribution) ではクライアント側での処理が少ないため，処理時間はクライアント側ですべての処理を行う場合 (Client) と比べて大きく削減された．計測された時間は通信時間よりも画像の処理時間が主であるため，クラウドで全ての処理行う場合 (Cloud) が最も速いが，ネットワーク帯域を考慮し通信時間を 100 倍にすると，通信時のデータ量が最初のデータ量よりも小さくなるフィルタ数 12 未満においては，分散処理の方が速くなった，また，図 8 ではクライアント側で全ての処理を行う場合が最も速くなるが，実際の環境ではクライアント側の端末にはリソースも限りがあり処理能力も低い可能性が高いため，処理時間は長くなることが予測される．

分散方法 2 で，分散方法 1 と比較するとクライアント側での処理が多いこと，conv1 層のフィルタ数が 32 に固定されているため conv1 層から conv2 層までの間のデータ量が大きく処理時間も長くなることから処理時間は長くなった．分散方法 1 と同じくクラウドで全ての処理を行う場合が最も速いという結果であっが，分散方法 2 では通信時のデータ量がより小さくなるため，通信時間を 100 倍にするとクラウドで全ての処理をする場合よりも短い時間が計測された．分散方法 1 と同じように，クライアントで全ての処理を行う場合の方が速くなったが，想定する環境ではクライアント側での処理には時間がかかることが考えられるため，分散処理は有効であると考えられる．

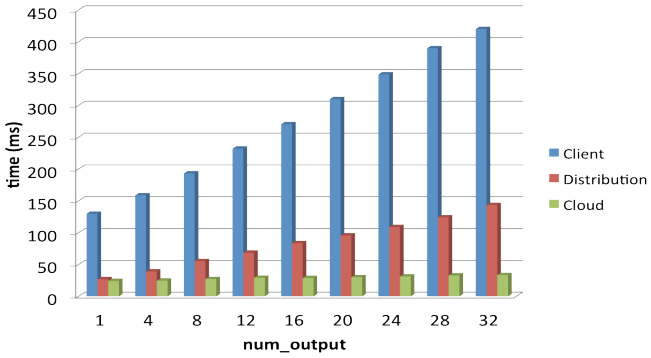

図 7　conv1 層のフィルタ数を変化させた場合の分散方法 1 の処理時間

5. 関 連 研 究

近年ニューラルネットワークは正確にパターンを分類し識別するのに広く用いられている [4] [5]．しかし，そのフレームワークの多くは一つの計算機で GPU のを活用して実行することに焦点が置かれいてる．関連研究として挙げられる DIANNE ミドルウェアフレームワークでは，通常のニューラルネットが入力層，出力層と一つ以上の隠れ層から構成されるのに対し，

各々のニューラルネットワークの層がモジュールで構成される [6].

このモジュール的アプローチにより複数の異なるデバイスに分散してニューラルネットワークの構成要素を実行することを

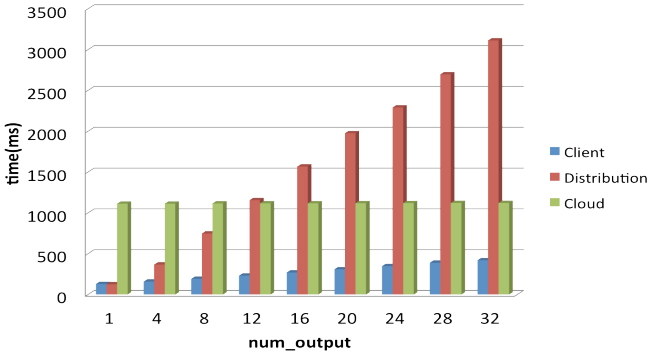

図 8　conv1 層のフィルタ数を変化させた場合の分散方法 1 の処理時間 (通信時間 100 倍)

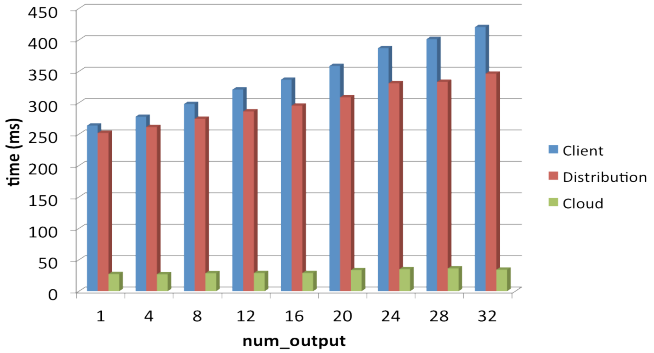

図 9　conv2 層のフィルタ数を変化させた場合の分散方法 2 の処理時間

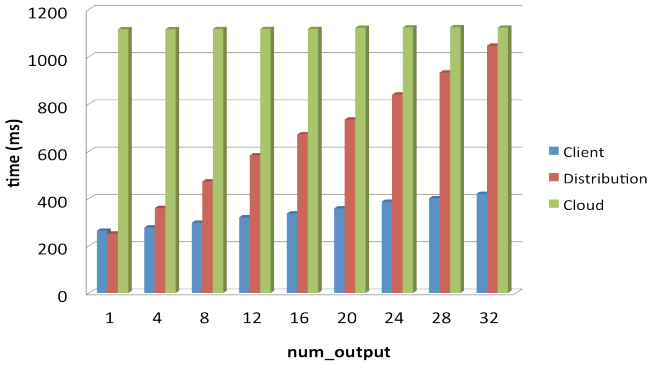

図 10　conv2 層のフィルタ数を変化させた場合の分散方法 2 の処理時間 (通信時間 100 倍)

可能にしている．しかし DIANNE ミドルウェアフレームワークでは通信時のデータ量や分散させた際の処理時間については言及されておらず，全体として処理時間が長くなってしまうことも考えられる．本研究では一般消費者のライフログの処理における効率のよいフレームワークを考える．

6. まとめと今後の課題

本研究ではディープラーニングのフレームワークである Caffe を分散環境へ適用させることにより，プライバシやネットワーク帯域を考慮したセンサデータ解析処理を検討した．畳込み層におけるフィルタ数を変化させることにより，識別率を大幅に下げることなくクライアントからクラウドへ送る際のデータ量を削減できることが確認できた．そしてクライアントとクラウド間で処理を分散させた場合の処理時間を計測し，通信データ

量を削減することによりネットワーク帯域が狭い環境で提案手法が有効であることが確認できた．

今後の課題としては，今回通信時間を伸ばすことによりネットワーク帯域を絞った環境を考慮したが厳密には模擬しきれていない．よって，実際に帯域を絞った環境で評価を行うことや，クライアント側の端末の処理能力が低い環境で評価を行うこと，データサイズの大きなデータセットを用いた実験などを検討している．

謝　　辞

この成果の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構（ＮＥＤＯ）の委託業務の結果得られたものです．

文　　献

[1] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S. and Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding, *CoRR*, Vol. abs/1408.5093 (2014).

[2] Alex, K., Nair, V. and Hinton, G.: The CIFAR-10 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`(accessed December 27, 2015).

[3] Coates, A., Lee, H. and Ng, A. Y.: An Analysis of Single Layer Networks in Unsupervised Feature Learning. `https://cs.stanford.edu/~acoates/stl10/`(accessed December 27, 2015).

[4] Kriahevsky, A., Sutskever, I. and Hinton, G.: ImageNet classification with deep convolutional neural networks, *NIPS* (2012).

[5] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and Lecun, Y.: Integrated recognition, localization and detection using convolutional networks, *ICLR* (2014).

[6] Coninck, E. D., Verbelen, T., Vankeirsbilck, B., Bohez, S., Leroux, S. and Simoens, P.: DIANNE:Distributed Artificial Neural Networks for the Internet of Things (2015).