ディープラーニングフレームワークCaffeの分散処理の評価

一瀬 絢衣[1]　竹房 あつ子[2]　中田 秀基[3]　小口 正人[1]

概要：近年各種センサの普及とクラウドコンピューティング技術の習熟により，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データをクラウドで収集して解析するアプリケーションも普及してきている．ここで，センサデータをそのまま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，フレームワークCaffe の機械学習処理をセンサ側とクラウド側でパイプライン的に分散処理を行う．その評価として，分割箇所やパラメータを変化させ処理時間を比較してきた．本稿では ImageNet を用いて実験を行い，大規模なデータセットにおいても分散処理を行うことで生データをクラウドに送ることなく遜色ない学習が可能であり，また低帯域環境ではより高速に処理できることを示す．

Evaluation of Distributed Processing of the Caffe Deep Learning Framework

Ayae ICHINOSE[1]　Atsuko TAKEFUSA[2]　Hidemoto NAKADA[3]　Masato OGUCHI[1]

1. はじめに

近年各種センサの普及とクラウドコンピューティング技術の習熟により，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データをクラウドで収集して解析するアプリケーション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．このようなサービスでは，一般家庭にサーバやストレージを設置して全ての処理を行うことは困難であるため，センサデータをそのまま送信してクラウドで処理するというのが一般的である．しかし，連続的に大容量データを転送する必要があるため，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．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ある．

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，フレームワーク Caffe の機械学習処理をセンサ側とクラウド側でパイプライン的に分散処理を行う．その評価として，分割箇所やパラメータを変化させ処理時間を比較してきた．本稿では ImageNet を用いて実験を行い，大規模なデータセットにおいても分散処理を行うことで生データをクラウドに送ることなく遜色ない学習が可能であり，また低帯域環境ではより高速に処理できることを示す．

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．

Caffe(Convolutional Architecture for Fast Feature Em-

[1] お茶の水女子大学
Ochanomizu University, Bunkyo, Tokyo 112-8610, Japan
[2] 国立情報学研究所
National Institute of Informatcs, Chiyoda, Tokyo 101-8430, Japan
[3] 産業技術総合研究所
AIST, Tsukuba, Ibaraki 305-8560, Japan

bedding) はディープラーニングのフレームワークの一つであり，Berkeley Vision and Learning Center が中心となって開発を進めている [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．コア部分は C++で実装されており，Caffe の C++ API を使って Notebook の例として挙げられる Python コードに類似した画像分類アプリケーションを実行することが可能である．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り，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．

Caffe のネットワークに利用される各層について記述する．

- Convolution
  畳込み計算を行う．フィルタ数とカーネルサイズを設定する必要があり，オプションとしてフィルタの適用位置の間隔を表すストライドや入力データの縁に加えるピクセル数を表すパディング，チャネルの分割の数を表すグループを設定することが可能である．
- Pooling
  プーリングの計算を行う．カーネルサイズの指定が必要であり，プーリングの方法や，パディング，ストライドの設定が可能である．プーリングの方法には，画素値の集合から最大値を選ぶ最大プーリングや，平均値を計算する平均プーリングなどが挙げられる．
- Local Response Normalization
  インプットの正規化を行うことにより，側方抑制を行う．
- Inner Product
  インプットをベクトルとして扱い，単一ベクトルのアウトプットを生成する．全結合層である．

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，実装している．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，クラウド側でパイプライン的な分散処理を実現する．

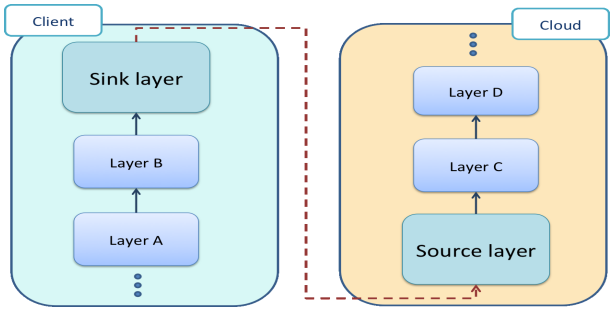

図 1 分散ディープラーニングフレームワーク

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指定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．データを受け取った際に 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る．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 ILSVRC2012 のデータセットを用いた [2]．ILSVRC は ImageNet が実施している大規模画像認識のコンペティションであり，2012 年に行われた ILSVRC2012 ではトレーニングデータとして 15 万のイメージが 1000 個のカテゴリに分類されたデータセットが提供された．トレーニングデータには，WordNet のオントロジーに従って各単語に対応する画像を収集した大規模画像データベースである ImageNet データセットの一部が利用されている．Caffe では ImageNet データセットの識別用ネットワークモデルが提供されており，その構造は図 2 のようになっている．各層で定義されているパラメータは表 1 に示す通りである．

Caffe においてデータはバッチサイズ，チャネル数，2 次元イメージサイズの 4 次元の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．テストにおけるバッチサイズでは 256 に設定されているため，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データ量は始めの (256 × 3 × 227 × 227)byte からフィルタ数 96，ストライド 4 の conv1 層を通り (256 × 96 × 55 × 55)byte へ，次にストライド 2 の pool1 層を通り (256 × 96 × 27 × 27)byte へと変化している．ここでデータ量は始めと比べて約 2 分の 1 となり，同じように pool2 層後は約 3 分の 1 となる．

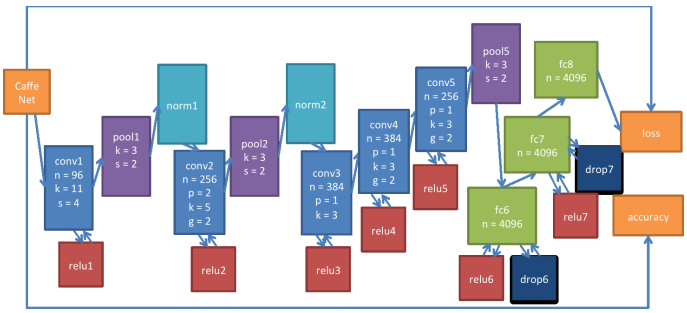

図 2　分散ディープラーニングフレームワーク

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
g：group	チャネルの分割数

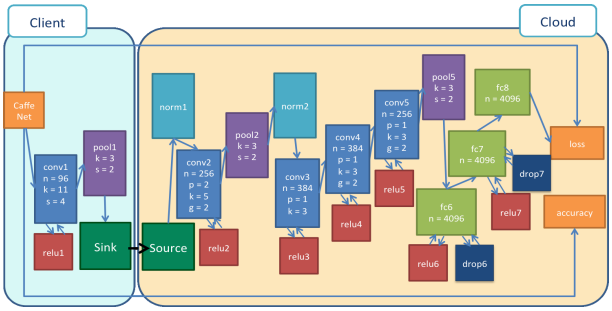

図 3　分散方法 1

4.2　分散方法

本稿で実験に利用した，通信時のデータ量を考慮した 2 つの分散方法を示す．

- 分散方法 1

  pool1 層，norm1 層間でネットワークを分離する (図 3)．データ量はデフォルト値で約 2 分の 1 となっているが，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さらに削減させることを考える．本稿ではフィルタ数をデフォルト値の 4 分の 3 である 72，2 分の 1 である 48 に設定した．データ量を削減させると識別率の低下に繋がることが考えられるが，デフォルト値の識別率が 57.4%であったのに対し，フィルタ数が 72 の場合が 56.8%，48 の場合が 56.3%であり，高い識別率を保持できていることが確認できた．

- 分散方法 2

  pool2 層，norm2 層間でネットワークを分離する (図 4)．データ量はデフォルト値で約 3 分の 1 となっているが，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さらに削減させることを考える．本稿ではフィルタ数をデフォルト値の 4 分の 3 である 192，2 分の 1 である 128 に設定した．識別率は 192 の場合が 57.0%，128 の場合が 56.4%であり，分散方法 1 と同じく高い識別率を計測した．

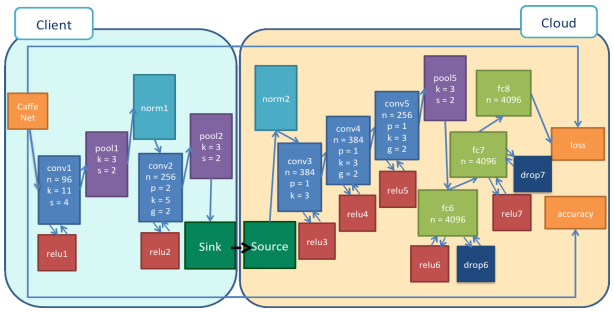

図 4　分散方法 2

表 2　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

4.3　処理時間計測による提案手法の評価

クライアント側，クラウド側として 2 つの端末を利用して 1 バッチの識別の処理時間を計測することにより，提案手法の有効性を示す．クライアント側で全ての処理を行う場合 (Client)，提案手法により処理を分散させる場合 (Distribution)，データをそのまま送信してクラウドで処理を行う場合 (Cloud) の 3 つの処理時間を比較する．また，実験ではクライアントとクラウド間のネットワーク帯域を一般的なセンサとクラウド間の通信環境を想定して，1Gbps から 10Mbps まで変化させた．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む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．実験環境を表 2 に示す．クライアント側およびクラウド側で同質のノードを用い，その間のネットワーク帯域は 1Gbps でつながれている．ネットワーク帯域制御には PSPacer を用いた [3]．

フィルタ数をそれぞれ 96，72，48 に設定した分散方法 1 の処理時間を図 5，6，7，フィルタ数をそれぞれ 256，192，128 に設定した分散 2 の処理時間を図 8，9，10 に示す．いずれの実験においても，2 つのマシン間のネットワーク帯域が 1Gbps の場合には，データをそのまま送りクラウド側で全ての処理を行う場合が最も高速に処理できた．しかし 10Mbps や 50Mbps の場合は，生のデータはデータ量が多く通信に時間がかかってしまい，全体の処理時間も長くなってしまうことが確認できた．また，分散処理における処理時間は，フィルタ数の削減によりさらに短縮可できることが確認できた．2 つのマシン間のネットワーク帯域が 10Mbps の場合は，conv2 層のフィルタ数を 128 に設定し

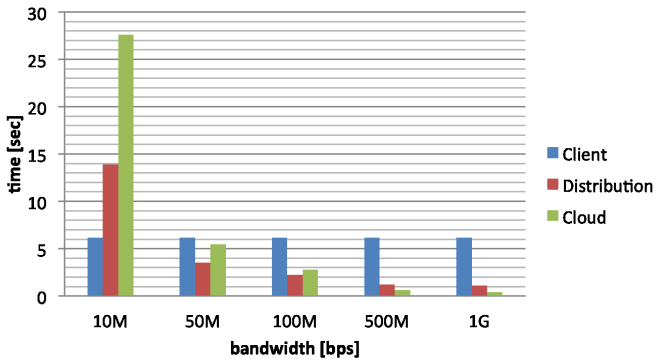

図 5 conv1 層フィルタ数をデフォルト値 96 に設定した場合の分散方法 1 の処理時間

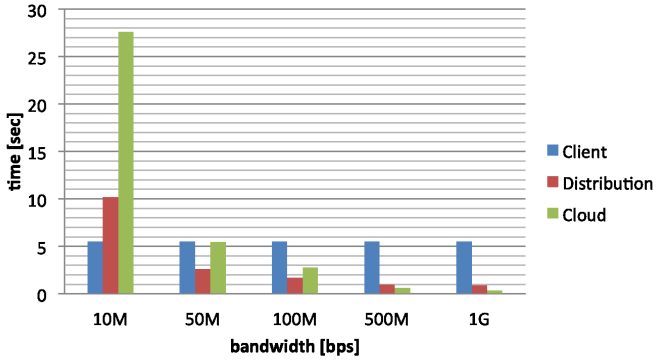

図 6 conv1 層のフィルタ数を 72 に設定した場合の分散方法 1 の処理時間

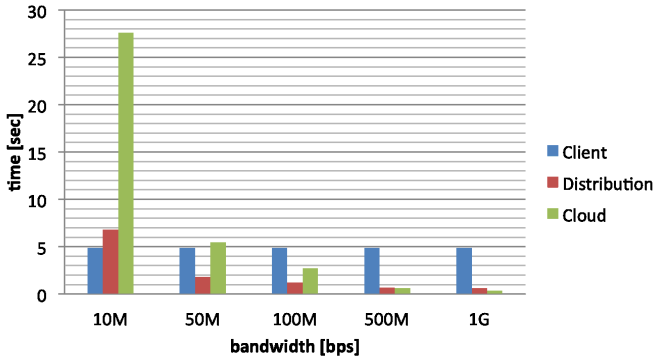

図 7 conv1 層のフィルタ数を 48 に設定した場合の分散方法 1 の処理時間

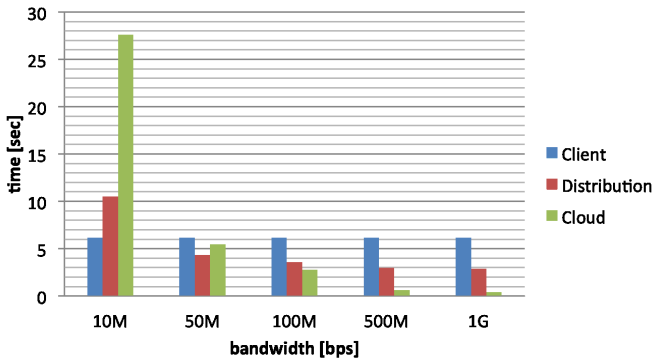

図 8 conv2 層のフィルタ数をデフォルト値 256 に設定した場合の分散方法 2 の処理時間

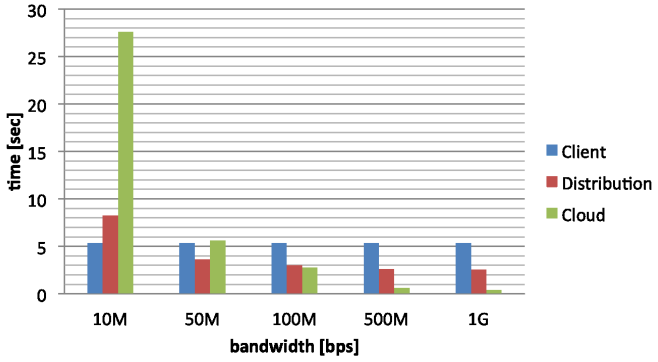

図 9 conv2 層のフィルタ数を 192 に設定した場合の分散方法 2 の処理時間

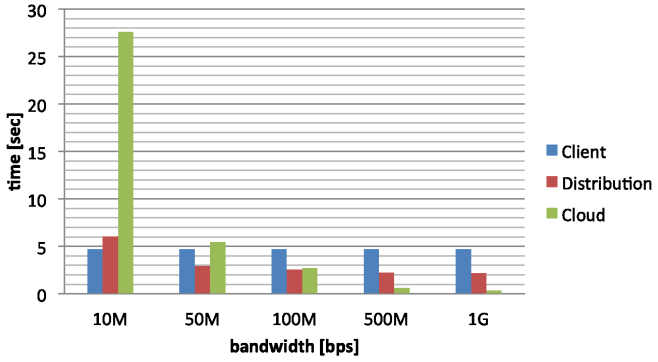

図 10 conv2 層のフィルタ数を 128 に設定した場合の分散方法 2 の処理時間

た分散方法 2 の処理時間が最も短く，50Mbps，100Mbps の場合は conv1 層のフィルタ数を 48 に設定した分散方法 1 の処理時間が最も短くなり，ネットワーク帯域やクライアント側，クラウド側のマシンの性能により，効率のよい分散方法が異なることもわかった．

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 を活用して高速に実行することに焦点が置かれいてる．関連研究として DIANNE ミドルウェアフレームワークを挙げる [6]．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，DIANNE ミドルウェアフレームワークでは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．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可能にしている．一方 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．本研究では通信時のデータ量を考慮することにより処理時間の削減を図り，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，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目的とし，ディープラーニングフレームワーク Caffe のネットワークの分離を実装した．一般家庭とクラウドの現実的なネットワーク帯域を考慮すると，センサデータをそのままクラウドに送ると時間がかかってしまい，提案手法の有効性が示された．また，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サイズを小さくできることが確認でき，通信データ量の削減によりさらに効率の良い処理が可能であることがわかった．

今後の課題として，クライアント側としてより現実的な性能の端末を用いた実験や，ライフログに近いデータセットを用いた実験を検討している．

謝辞

この成果の一部は，JSPS 科研費 JP16K00177 および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．

参考文献

[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] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)*, Vol. 115, No. 3, pp. 211–252 (2015).

[3] Takano, R., Kudoh, T., Kodama, Y. and Okazaki, F.: High-resolution Timer-based Packet Pacing Mechanism on Linux Operating System, *IEICE Transactions on Communications*, Vol. E94.B, No. 8, pp. 2199–2207 (2011).

[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).