広域分散を想定した深層学習手法の比較

小国　英明†　　高橋　良希††　　首藤　一幸††

† 東京工業大学 理学部 情報科学科
†† 東京工業大学 情報理工学院 数理・計算科学系
〒 152–8552 東京都目黒区大岡山 2-12-1-W8-43

あらまし　機械学習は通常，すべての学習データが手元に揃っていることを前提とする．しかし，病院が持つ個人データなど，プライバシなどの理由から 1 ヵ所に集めることのできない場合，データを広域に分散させたまま学習を行う必要がある．本研究では，広域分散環境での深層学習を想定し，通信パターンの異なる手法を，学習効率の観点で比較する．多数のノードを用意することは困難なため，学習は GPU 上で行い，通信時間はシミュレーションを行う．全ノードの性能が同一でネットワークが均質という環境において，通信パターンに all-reduce を使用する手法の方が gossip を使用する方法より学習効率が良いことが分かった．

キーワード　深層学習，広域分散，シミュレーション

1 はじめに

スマートフォンや IoT デバイスの普及などによって機械学習の対象となるデータは多く生成されている．通常，使用する学習データは 1 ヵ所に集めて学習を行う．

しかしスマートフォンに保存されている画像や病院が持つ個人データなどは，プライバシなどの理由から集められない場合がある．例えば網膜画像からは年齢，性別，喫煙状況，血圧，主要心血管イベントなどが推定できてしまう [1]．データを集められない場合，広域に分散させたまま学習を行うことになる．

クラスタを用いた，広域ではない分散深層学習では all-reduce という通信方法を用いて学習モデルを更新する手法が主に研究されている [2], [3], [4], [5]．all-reduce は各ノードの持つ情報を同期的な通信によって集約，共有する操作である．

しかし広域の場合，同期的な通信方法である all-reduce を用いる手法が適切とは限らない．クラスタを用いる通常の分散深層学習と異なり，ネットワークの遅延が大きくなり，帯域幅が限られ，さらにマシンの性能も均質でないからである．

さらに，広域分散ではデータの分布やデータ数の偏りなども考慮する必要がある [6]．データの分布の偏りとは，ノードが持つデータのラベルの種類が偏っていることなどである．

本研究ではネットワークの遅延，帯域幅，マシンの性能が均質である場合について，gossip と all-reduce の 2 つの通信方法を用いた手法についてシミュレーションを行って比較する．各ノードのデータ数は偏らせないが，データの分布については偏りがある場合の実験も行う．

本稿の構成は以下の通りである．2 章では分散深層学習の既存手法について述べ，3 章では広域な環境で留意すべきことと本研究で用いる手法を述べる．4 章では実験を行い，ノード数の増加や通信頻度による学習効率の変化について考察する．5 章では関連研究を述べ，6 章で本研究のまとめと今後の課題を述べる．

This is an unrefereed paper.

2 分散深層学習

本研究では広域分散環境での深層学習を想定するが，まず本章でクラスタを用いる分散深層学習の手法を述べる．そして次章でその手法を広域へと拡張する．

分散深層学習は複数のマシン（ノード）が学習データを使用して計算を行い，ノード間で通信をすることで全体としての学習を進める．

モデルの最適化は確率的勾配降下法 (Stochastic Gradient Descent, SGD) に基づいて行う．SGD とは，モデルを w, 学習率を α, 目的関数を f としたとき，データをランダムに一つ選んで以下の更新を繰り返し, f を最小化するアルゴリズムである．

$$w \leftarrow w - \alpha \nabla f(w) \tag{1}$$

実際には，一度の更新にデータを複数使用するミニバッチ SGD により，モデルを最適化することがほとんどである．これはミニバッチのデータ数（ミニバッチサイズ，以降バッチサイズと表記する）を n としたとき以下の更新を繰り返すアルゴリズムである．

$$w \leftarrow w - \alpha \frac{1}{n} \sum_{k=1}^{n} \nabla f(w) \ (= w - \alpha \nabla F(w)) \tag{2}$$

以上で述べた学習理論を並列環境で実行する場合，各ノード上で分散して式 2 による更新を適用し，勾配情報を互いに通信しあいながら誤差が小さくなるモデルを獲得する．各ノードはそれぞれ別のデータとモデルを持つ．そして勾配情報を通信してモデルの勾配の平均を取るなどのマージを行うことで誤差を小さくする．

中央管理のサーバを用いない場合の通信方法には，同期的な方法と非同期的な方法が考えられる．同期的な方法は MPI の

Algorithm 1 (pull) gossip SGD (Run on client i).

Require: $w_{0,i} = w_0, t = 0$
  loop
    shuffle(X_i)
    for minibatch $x \in X_i$ **do**
      $w_{t+1,i} \leftarrow w_{t,i} - \alpha \nabla F_i(w_{t,i}; x)$
      $t \leftarrow t + 1$
      if $t \equiv 0 \bmod T$ **then**
        choose j at random
        $w_{t,i} \leftarrow$average$(w_{t,i}, w_j)$
      end if
    end for
  end loop

Algorithm 2 all-reduce SGD (Run on client i).

Require: $w_{0,i} = w_0, t = 0$
  loop
    shuffle(X_i)
    for minibatch $x \in X_i$ **do**
      $\Delta w_{t+1,i} \leftarrow -\alpha \nabla F_i(w_t; x)$
      $\Delta w_{t+1} \leftarrow$all-reduce-average$(\Delta w_{t+1,i})$
      $w_{t+1} \leftarrow w_t + \Delta w_{t+1}$
      $t \leftarrow t + 1$
    end for
  end loop

all-reduce に見られるように，あるタイミングで全ノードのモデルの勾配を集約するような方法である．非同期的な方法は gossip に見られるように，各ノードが自律的に通信先ノードを選択し，そのノードが保有するモデルの勾配と平均化するような方法である．

分散深層学習では，パラメータサーバと呼ばれる中央管理のサーバを活用する方法もあるが，本研究は完全分散な環境を想定するため，本研究では扱わない．パラメータサーバはモデルを管理するサーバである．サーバはノードにモデルを与え，ノードはそのノードが持つデータで学習する．そしてサーバはノードから学習済みモデルを受信し，その情報をもとにサーバのモデルを更新する．

2.1 Gossip を用いた分散深層学習

まず非同期的な通信パターンである gossip を用いる手法を述べる．gossip は各ノードが非同期に自律的な通信を行うことで，大域的な情報を緩く共有する通信方法である．

gossip SGD の疑似コードを Algorithm 1 に示す．各ノードはそのノードが持っているデータ X_i を使用して，SGD に基づきモデルを更新する．そして定期的にランダムに選んだ他のノード j と非同期通信を行ってモデルを受信し，モデルの平均をとる．モデルの平均は，w_i，w_j の各パラメータに対してそれぞれ平均を計算することで求められる．

このようなモデルの更新を繰り返して，ノード全体のデータを反映したモデルを各ノードで共有する．

各ノードが他のノードとモデルを共有するための通信頻度は T によって調整する．各ノードは自身のデータを用いて自身のモデルを更新することを T 回繰り返した後に通信を行う．

通信頻度は学習モデルや学習データによって変化させるべきである．通信頻度 T が小さいと，学習するデータ数に対して多くのノードと通信を行うことになるため，モデルの誤差は小さくなる．一方，T が大きいと，学習するデータ数に対して通信回数が少なくなるため，学習時間が短くなる．既存研究では $T = 1$ が用いられている [2]．

ここで述べた gossip SGD は pull-gossip と呼ばれる手法である．gossip SGD は他に push-gossip という手法もあるが，ノード数が多い場合，pull-gossip の方が学習効率が良いことが実験により示されている [2]．push-gossip SGD はそのノードが持っているデータを使用してモデルを更新する．そしてランダムに選んだノードにモデルを送信し，学習の間に受信したモデルの平均をとる．

2.2 All-reduce を用いた分散深層学習

次に同期的な通信パターンである all-reduce を用いる手法を述べる．all-reduce は各ノードの持つ情報を同期的な通信によって集約，共有する操作である．

all-reduce SGD の疑似コードを Algorithm 2 に示す．各ノードはそのノードが持っているデータ X_i を使用して，勾配を計算する．そしてノード間で all-reduce を行うことで全ノードの勾配の平均を計算，共有する．

all-reduce には butterfly, recursive halving and doubling [7] と ring [8] などの方法があり，それぞれ通信量と通信回数が異なる．butterfly は通信回数が少なく，recursive halving and doubling と ring は通信量が少ない．K, G をそれぞれノード数，勾配の容量としている．K は 2 冪とする．

butterfly all-reduce は，各ノードが $\log_2 K$ 回通信を行う．1 回の通信で勾配を互いに通信し合うため，各ノードは合計で $\log_2 K \times G$ のデータを送受信する．

recursive halving and doubling all-reduce は送信したいデータを K 個に分割して 1 回に通信するデータ量を少なくする．$2\log_2 K$ 回通信を行い，合計で $\frac{2(K-1)}{K} \times G$ のデータを送受信する．

ring all-reduce も送信したいデータを K 個に分割することで 1 回に通信するデータ量を少なくする．$2(K-1)$ 回通信を行い，合計で $\frac{2(K-1)}{K} \times G$ のデータを送受信する．

butterfly all-reduce, recursive halving and doubling all-reduce と ring all-reduce は送信するデータの容量とネットワーク環境によって使い分ける必要がある．データの容量が大きい場合は帯域幅がボトルネックとなり，小さい場合には遅延がボトルネックとなる．また，ネットワークの帯域幅が狭い場合には通信量の少ない recursive halving and doubling all-reduce や ring all-reduce の方が通信時間が短くなり，遅延が大きい場合には通信回数の少ない butterfly all-reduce の方が通信時間が短くなる．

クラスタを用いる場合，帯域幅は 10 Gbps 程度と広いことが

Algorithm 3 all-reduce2 SGD (Run on client i).

Require: $w_{0,i} = w_0, t = 0$
  loop
    shuffle(X_i)
    for minibatch $x \in X_i$ **do**
      $w_{t+1,i} \leftarrow w_{t,i} - \alpha \nabla F_i(w_{t,i}; x)$
      $t \leftarrow t + 1$
      if $t \equiv 0 \bmod T$ **then**
        $w_t \leftarrow$all-reduce-average($w_{t,i}$)
      end if
    end for
  end loop

想定され，通信時間に対して遅延の占める割合は低い．そのため，butterfly all-reduce より recursive halving and doubling all-reduce や ring all-reduce の方が通信時間が短いことが多い．

3 広域分散深層学習

本章では，広域分散環境で留意すべきことを述べ，さらに本研究で用いた手法を述べる．

広域な環境では，クラスタを用いる場合に比べて，ネットワークの遅延が大きくなり，帯域幅が限られる．gossip SGD は学習モデルを平均化するとき，1 つのノードと非同期通信を行う．一方，all-reduce SGD は複数のノードと同期通信を行う．そのため，学習にかかる時間に占める，モデルの平均化を行う通信時間の割合も異なり，帯域幅や遅延は学習効率に影響を与える．

さらに，帯域幅がノードによって異なる場合，帯域幅が狭いノードは通信時間に影響を及ぼす．帯域幅の狭いノードによる 1 回のモデルの平均化での影響は，非同期通信を行う gossip SGD の場合 1 つのノードのみであるが，同期通信を行う all-reduce SGD の場合すべてのノードに及ぶ．そのため，このような場合には gossip SGD の方が，all-reduce SGD より学習効率が良くなることも考えられる．

前章でも述べた通り，クラスタを用いる分散深層学習では，recursive halving and doubling all-reduce や ring all-reduce の方が butterfly all-reduce より通信時間が短くなることが多い．しかし広域な環境では，ネットワークの遅延も考慮して，全体の通信時間が短くなる方法を選ばなければならない．勾配の計算と通信をオーバーラップさせる手法もあるが，簡単のため本研究では扱わない．帯域幅，遅延が均質なとき，1 回の all-reduce にかかる通信時間は以下のようになる．K, G, B, L をそれぞれノード数，勾配の容量，帯域幅，遅延としている．K は 2 冪とする．

- butterfly all-reduce:

$$\left(\frac{G}{B} + L\right) \times \log_2 K$$

- recursive halving and doubling all-reduce:

$$\frac{G}{B} \times \frac{2\,(K-1)}{K} + L \times 2 \log_2 K$$

表 1 実験環境

OS	Ubuntu 16.04.2 LTS
CPU	Intel Xeon E5-2698 v4
GPU	Tesla P100-PCIE-16GB
イーサネット	10 GBASE-T

表 2 データセットの概要

データセット	学習データ数	テストデータ数
MNIST	60000	10000
CIFAR-10	50000	10000

- ring all-reduce:

$$\left(\frac{\frac{G}{K}}{B} + L\right) \times 2\,(K-1)$$

通常，all-reduce SGD は all-reduce を勾配の計算ごとに行う．しかし，広域な環境では通常の分散深層学習と比べて，学習時間に対して通信時間が大きくなる．そこで本研究では，all-reduce SGD も gossip SGD と同様に，通信を定期的に行うことにする．その疑似コードを Algorithm 3 に示す．

深層学習ではバッチサイズも学習効率に影響を与える．バッチサイズが大きすぎると精度が下がる可能性があるが，小さすぎても学習するデータ数に対する通信回数が多くなる．そのため，広域な環境ではバッチサイズをある程度大きくする必要がある．

4 実験と考察

本章では，2，3 章で述べた gossip SGD と all-reduce SGD の学習効率の比較をする．

本研究で使用した実験環境は表 1 の通りである．

4.1 データセット

データセットは MNIST [9] と CIFAR-10 [10] を用いた．MNIST は 0∼9 の手書き数字の白黒画像である．CIFAR-10 は航空機，自動車などの 10 種類のカラー画像である．表 2 にデータセットの概要を示す．

ニューラルネットワークは MNIST では，Federated Learning で用いられている CNN, CIFAR-10 では VGG16 [11] を用いた．

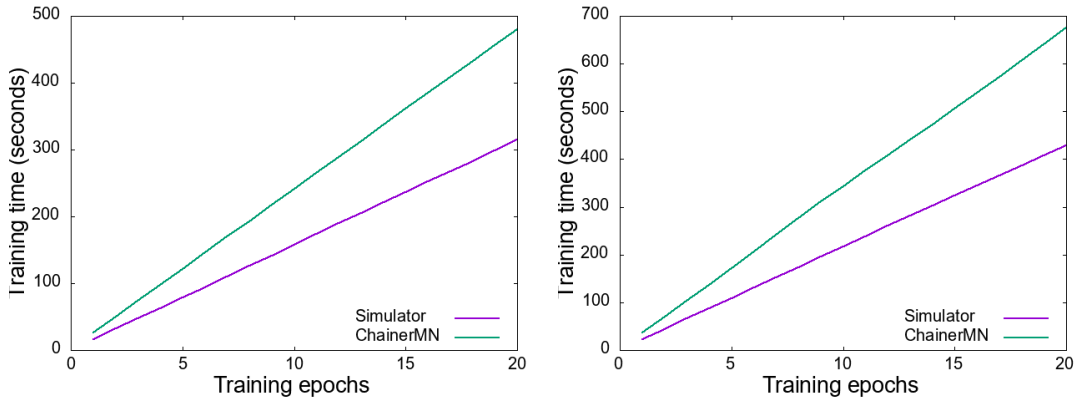
4.2 準　　備

ノード数が 16, 32, 64 の 3 つの場合について実験を行った．実機で多数のノードを用意することは困難であるため，学習は GPU 上で行い，通信時間はシミュレーションを行った．

まず，通信時間を決めるために，学習モデルの容量を調べた．all-reduce SGD を実行し，1 エポックごとに 10 回，numpy 配列を保存するときによく用いられる npz ファイルとしてモデルを圧縮して保存した．10 回ともモデルの容量は同じで，MNIST は 2.1 MiB (≒2.2 MB), CIFAR-10 は 54 MiB (≒56 MB) であった．

4.3 シミュレータ

シミュレータが適切に実装されているか確かめるために，4



(a) MNIST　　(b) CIFAR-10

図 1　シミュレータと ChainerMN の比較

ノードの場合に対して，分散深層学習フレームワークである ChainerMN [12] と比較を行った．これは広域な環境ではなく，クラスタを用いる通常の分散深層学習の実験である．

シミュレータと ChainerMN の all-reduce SGD を，エポック数ごとの学習時間をグラフにしたものが，図 1 である．帯域幅は iperf を用いて測定し 9.4 Gbps, ノード間の遅延は ping を用いて計測し 0.077 ms であった．シミュレータはこの数値を使用して実験を行った．ChainerMN は ring all-reduce を用いているため，シミュレータも ring all-reduce を用いた．

MNIST, CIFAR-10 どちらもシミュレータの学習時間の方が短くなっているが，これは ChainerMN の通信にオーバーヘッドがあるためだと考えられる．

どちらのデータセットの場合でも学習時間の予測は大きく外れてはいないため，以降の実験ではシミュレーションを用いて行う．

複数拠点が広域ネットワークで通信を行うことを想定し，全ノード均質に帯域幅を 1.0 Gbps, 遅延を 5.0 ms に設定した．all-reduce は butterfly all-reduce と recursive halving and doubling all-reduce のうち通信時間が短くなる方を用いた．精度は，gossip SGD では全ノードのテストデータに対する正答率の平均，all-reduce SGD ではテストデータに対する正答率である．

データを偏らせた実験では次のように学習データをノードに分配した．まずデータをラベルを基準にソートした．そしてソートしたデータをノード数 ×2 だけ分割して，各ノードにランダムに 2 つずつ分割したデータを分配した．このように分配すると，各ノードが持つデータのラベルは 1 ～ 4 種類となる．

4.4　MNIST

MNIST の実験は学習率を 0.1 に固定して行った．

まずは各ノードのデータの分布に偏りがない場合の実験を行った．最初に 16 ノードの実験で適切なバッチサイズと通信頻度を調べた．

バッチサイズを大きくすると，学習するデータ数に対して通信頻度が下がる．そのため，学習時間に占める通信時間の割合も小さくなる．しかし，バッチサイズを変化させた実験を行ったが学習効率はほとんど変わらなかった．通信時間は短くなったが，バッチサイズを大きくしたことによる学習の質の低下も起きていることが推測できる．

これに対し，バッチサイズを固定し，通信頻度を変化させた実験が図 2 である．バッチサイズを変化させた実験と異なり，通信頻度によって学習効率が変化している．gossip SGD では $T = 10$, all-reduce SGD では $T = 50$ までは通信頻度を減らすほど学習効率が良い．しかし，それ以上通信頻度を減らすと学習効率が悪い．

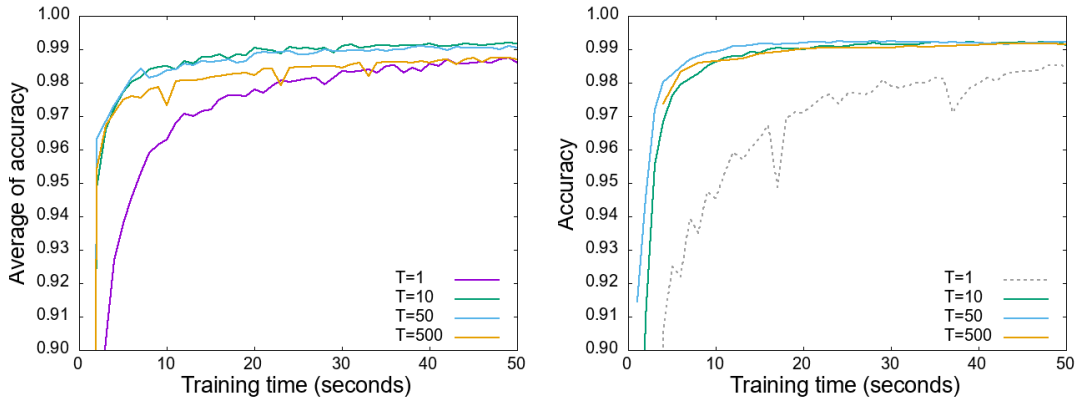
32, 64 ノードでは以上の結果をもとに，通信頻度を gossip SGD では $T = 10$, all-reduce SGD では $T = 50$ に固定した．バッチサイズは,（バッチサイズ）×（ノード数）が一定のものと，バッチサイズが一定のものに対して実験を行った．

図 3 がその実験結果である．gossip SGD, all-reduce SGD ともに 32 ノードではバッチサイズを変化させても学習効率はほとんど変化していない．一方，all-reduce SGD の 64 ノードの実験ではバッチサイズが 20 の方が学習効率がわずかに良い．all-reduce SGD は一般的に（バッチサイズ）×（ノード数）を一定にすることが多く，この実験でもそちらの方が学習効率が良いことが分かる．また，gossip SGD, all-reduce SGD ともにノード数の変化による学習効率の変化がほとんどないことも分かる．

図 4(a) は 16 ノードでの，gossip SGD と all-reduce SGD の $T = 1$ の学習効率を比較したグラフである．gossip SGD の方が all-reduce SGD の $T = 1$ よりも学習効率が良いことが分かる．一方，図 4(b) は 64 ノードでの，gossip SGD と all-reduce SGD の最も学習効率が良いものを比較したグラフである．わずかに all-reduce SGD の方が gossip SGD よりも学習効率が良いことが分かる．

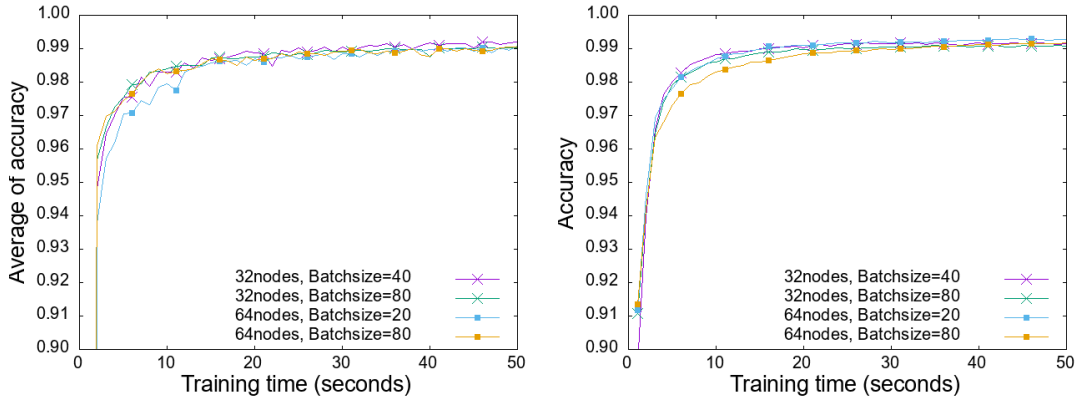
次に，データの分布に偏りがある場合の実験を行った．最初に 16 ノードの実験で適切なバッチサイズと通信頻度を調べ，その結果に基づき 32, 64 ノードの実験を行った．

図 5(a) は 16 ノードでの，gossip SGD と all-reduce SGD の $T = 1$ の学習効率を比較したグラフである．データの分布に偏



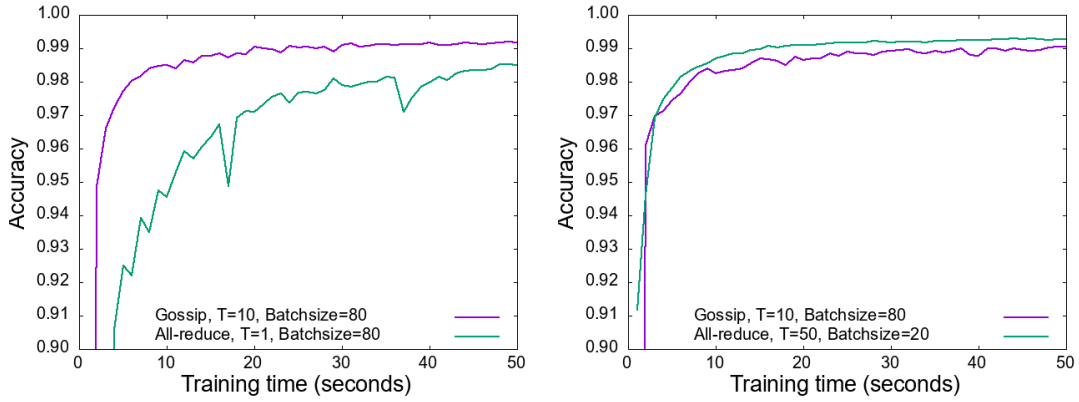
(a) Gossip,（バッチサイズ）= 80　　(b) All-reduce,（バッチサイズ）= 80

図 2　16 ノードで通信頻度を変化させたときの学習曲線 (MNIST)



(a) Gossip, $T = 10$　　(b) All-reduce, $T = 50$

図 3　32, 64 ノードでバッチサイズを変化させたときの学習曲線 (MNIST)



(a) 16 ノードの gossip と all-reduce ($T = 1$) の学習曲線　　(b) 64 ノードの gossip と all-reduce の学習曲線

図 4　gossip と all-reduce の学習曲線 (MNIST)

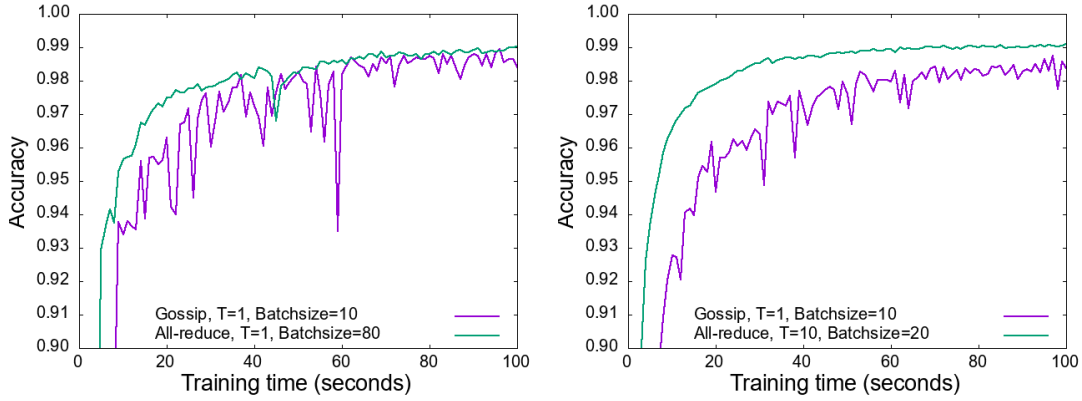
りがない場合と異なり，gossip SGD の方が all-reduce SGD の $T = 1$ よりも学習効率が悪くなっている．一方，図 5(b) は 64 ノードでの，gossip SGD と all-reduce SGD の最も学習効率が良いものを比較したグラフである．データの分布に偏りがない場合と比べると，gossip SGD に対する all-reduce SGD の学習効率の良さが大きくなっている．

4.5　CIFAR-10

CIFAR-10 の実験は学習率を 0.3 に固定して行った．

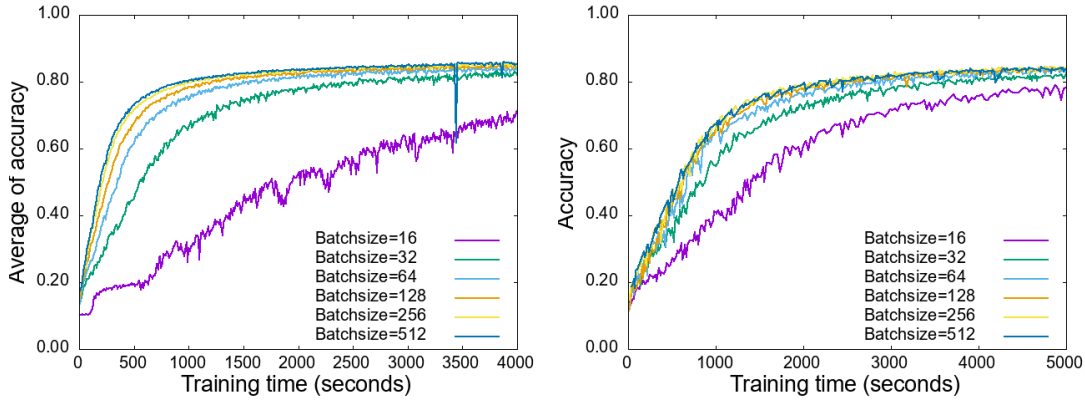
まずは各ノードのデータの分布に偏りがない場合の実験を行った．最初に 16 ノードの実験で適切なバッチサイズと通信頻度を調べた．

図 6 がバッチサイズを変化させた実験である．MNIST を使用した実験とは異なり，バッチサイズを大きくすると学習効率



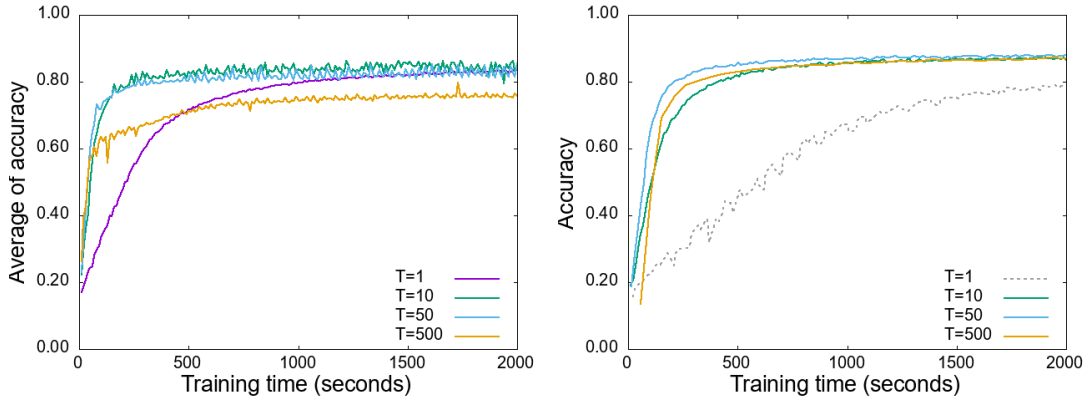
(a) 16 ノードの gossip と all-reduce ($T = 1$) の学習曲線　(b) 64 ノードの gossip と all-reduce の学習曲線

図 5　データ偏り，gossip と all-reduce の学習曲線 (MNIST)



(a) Gossip, $T = 1$　(b) All-reduce, $T = 1$

図 6　16 ノードでバッチサイズを変化させたときの学習曲線 (CIFAR-10)

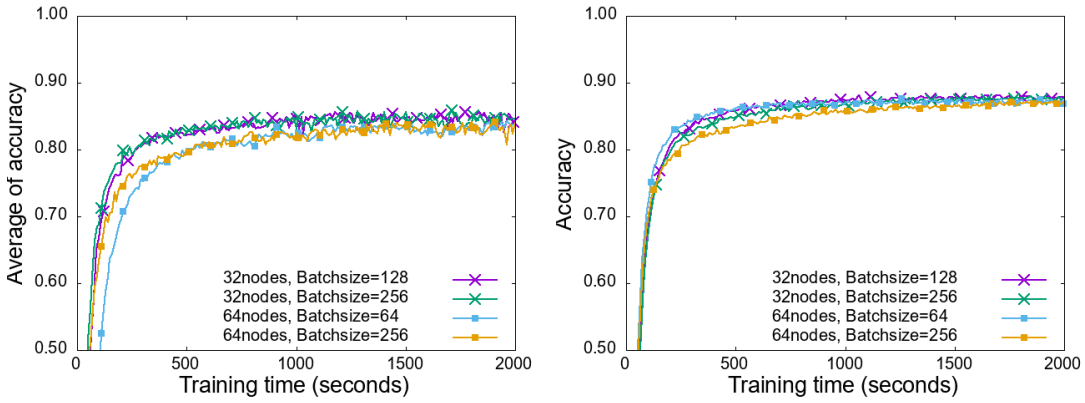


(a) Gossip,（バッチサイズ）= 256　(b) All-reduce,（バッチサイズ）= 256

図 7　16 ノードで通信頻度を変化させたときの学習曲線 (CIFAR-10)

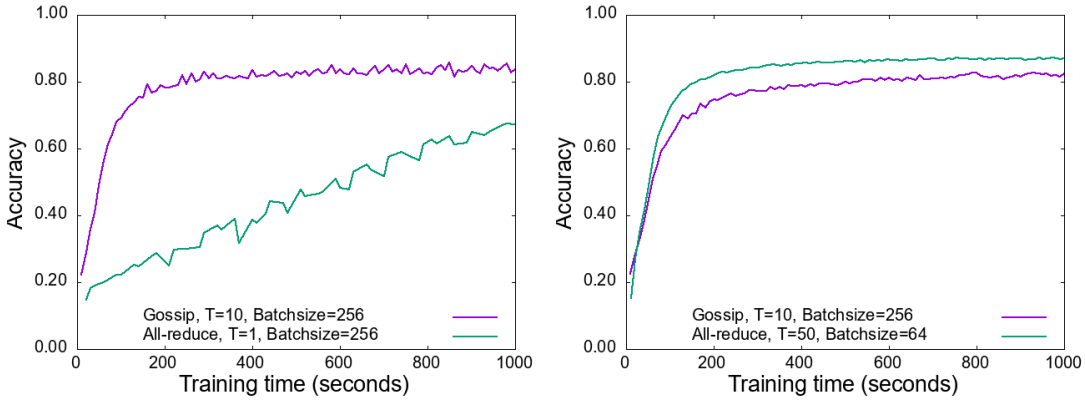
が向上している．この要因としては，MNIST で用いたモデルに比べて，CIFAR-10 で用いたモデルの容量が大きいことが考えられる．モデルの容量が大きいと，学習時間に対して占める通信時間の割合も大きくなる．するとバッチサイズを大きくしたことによる，学習するデータ数に対する通信頻度の低下が学習効率の向上につながると考えられる．gossip SGD, all-reduce SGD ともに 256 が適切なバッチサイズであることが分かる．

これに対し，バッチサイズを固定し，通信頻度を変化させた実験が図 7 である．こちらは MNIST の実験と同様，通信頻度によって学習効率が変化している．gossip SGD では，$T = 10$ までは学習効率が向上しているが，$T = 50$ では少し学習効率が低下している．一方，all-reduce SGD では，$T = 50$ までは学習効率が向上している．通信頻度を変化させるのは all-reduce SGD の方が効果が高いことが分かる．この要因は，すべての



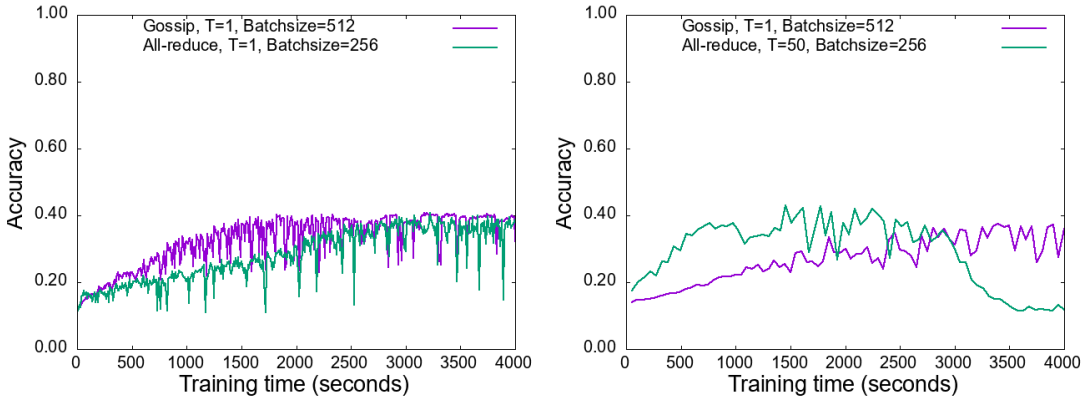
(a) Gossip, $T = 10$　　(b) All-reduce, $T = 50$

図 8　32, 64 ノードでバッチサイズを変化させたときの学習曲線 (CIFAR-10)



(a) 16 ノードの gossip と all-reduce ($T = 1$) の学習曲線　　(b) 64 ノードの gossip と all-reduce の学習曲線

図 9　gossip と all-reduce の学習曲線 (CIFAR-10)



(a) 16 ノードの gossip と all-reduce ($T = 1$) の学習曲線　　(b) 64 ノードの gossip と all-reduce の学習曲線

図 10　データ偏り，gossip と all-reduce の学習曲線 (CIFAR-10)

ノードのモデルを平均化する all-reduce の効果が非常に高いためだと考えられる．

32, 64 ノードでは以上の結果をもとに，通信頻度を gossip SGD では $T = 10$, all-reduce SGD では $T = 50$ に固定した．バッチサイズは,（バッチサイズ）×（ノード数）が一定のものと，バッチサイズが一定のものについて実験を行った．

図 8 がその実験結果である．gossip SGD, all-reduce SGD ともに 32 ノードではバッチサイズを変化させても学習効率はほとんど変化していない．一方，all-reduce SGD の 64 ノードの実験ではバッチサイズが 64 の方が学習効率がわずかに良い．ノード数が変化したとき，all-reduce SGD は一般的に（バッチサイズ）×（ノード数）を一定にすることが多く，この実験でもそちらの方が学習効率が良いことが分かる．また，gossip SGD, all-reduce SGD ともにノード数の変化による学習効率の

変化がほとんどないことも分かる．

図 9(a) は 16 ノードでの，gossip SGD と all-reduce SGD の $T = 1$ の学習効率を比較したグラフである．gossip SGD の方が all-reduce SGD の $T = 1$ よりも学習効率が良いことが分かる．一方，図 9(b) は 64 ノードでの，gossip SGD と all-reduce SGD の最も学習効率が良いものを比較したグラフである．わずかに all-reduce SGD の方が gossip SGD よりも学習効率が良いことが分かる．

次に，データの分布に偏りがある場合の実験を行った．最初に 16 ノードの実験で適切なバッチサイズと通信頻度を調べ，その結果に基づき 32, 64 ノードの実験を行った．

図 10(a) は 16 ノードでの，gossip SGD と all-reduce SGD の $T = 1$ の学習効率を比較したグラフである．どちらの実験でも精度が 40%ほどしか出ておらず，CIFAR-10 では MNIST と異なり各ノードが持つラベルの種類数が少ないと十分な精度が出ないことが分かる．

5 関 連 研 究

本章では，分散機械学習の関連研究を述べる．

広域分散環境での機械学習手法に gossip learning [13] がある．これはプライバシを考慮し，データをノードから移動させずに学習を行う手法である．さらに，これを改良した手法も提案されている [14]．しかしどちらも深層学習は扱っていないため，本研究とは異なる．

プライバシを考慮したものではないが，データがノードから移動しない分散深層学習の研究は多く存在する [2], [3], [4], [5], [15]．しかし，これらはクラスタを用いる分散深層学習の研究であり，広域な環境を想定したものではないため，本研究とは異なる．

プライバシを考慮し，データをノードから移動させない広域分散深層学習の手法に，パラメータサーバを使用する Federated Learning [6] がある．これは集中な環境を想定するため，完全分散な環境を想定する本研究とは異なる．

6 まとめと今後の課題

本稿では，広域分散深層学習の学習効率を，gossip と all-reduce の 2 つの通信パターンについて比較した．帯域幅が限られ遅延が大きい広域な環境では，データの分布に偏りがないとき，クラスタを使用する分散深層学習で多く用いられている all-reduce SGD の $T = 1$ の場合よりも，非同期的な gossip SGD の方が学習効率が良いことが分かった．しかし，データの分布に偏りがある場合は，gossip SGD の方が学習効率が悪くなることも分かった．さらに all-reduce SGD についても通信頻度を変化させると学習効率が向上し，帯域幅やマシンの性能が均質な環境では，gossip SGD を上回ることが実験により示された．

今後の課題としては，ノードごとに帯域幅や遅延，性能が異なる，非均質な環境を想定した実験が考えられる．本稿で想定した均質な環境では all-reduce SGD が良い結果を示すことが多かったが，非均質な環境では非同期的な gossip SGD の方が良い結果を示すことも予想される．

謝　　辞

本研究は JSPS 科研費 16K12406 の助成を受けたものです．本研究の一部は、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）の委託業務として行われました．

文　　献

[1] Ryan Poplin, Avinash V Varadarajan, Katy Blumer, Yun Liu, Michael V McConnell, Greg S Corrado, Lily Peng, and Dale R Webster. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, Vol. 2, No. 3, p. 158, 2018.

[2] Peter H Jin, Qiaochu Yuan, Forrest Iandola, and Kurt Keutzer. How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581*, 2016.

[3] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes. *arXiv preprint arXiv:1711.04325*, 2017.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[5] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. ImageNet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 2018.

[6] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[7] Rolf Rabenseifner. Optimization of collective reduction operations. In *International Conference on Computational Science*, pp. 1–9. Springer, 2004.

[8] Pitch Patarasuk and Xin Yuan. Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, Vol. 69, No. 2, pp. 117–124, 2009.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Takuya Akiba, Keisuke Fukuda, and Shuji Suzuki. ChainerMN: scalable distributed deep learning framework. *arXiv preprint arXiv:1710.11351*, 2017.

[13] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, Vol. 25, No. 4, pp. 556–571, 2013.

[14] 高橋良希, 首藤一幸. P2P ネットワーク上のデータに対する偏りのない機械学習手法. データ工学と情報マネジメントに関するフォーラム (DEIM), 2017.

[15] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.