広域分散の非均質性を考慮した深層学習手法

小国　英明†　　首藤　一幸†　　高橋 良希*†

† 東京工業大学 〒 152–8550 東京都目黒区大岡山 2-12-1

あらまし　深層学習では，一般的に学習用データが多いほど精度が高くなるが，プライバシなどの理由からデータを手元に集められない場合がある．このような場合にデータを広域分散させたまま学習を行う手法がある．広域分散では，クラスタを用いる分散深層学習と異なり，帯域幅やデータ，マシンの性能が非均質である．そのため，非同期的な通信方法を用いる gossip SGD が有力な手法である．本研究では，帯域幅が非均質な場合に，既存の gossip SGD を用いると輻輳が発生し，帯域幅が均質な場合と学習効率があまり変わらないことを明らかにする．この課題が通信頻度の変更により解決できることを示し，さらに学習効率を良くする warm-up 手法を提案する．

キーワード　広域分散，gossip，深層学習

1 はじめに

深層学習は機械学習の手法の一つであり，画像認識，音声処理，言語処理など多くの領域で効果を発揮している．深層学習は精度を高めるために，学習に用いるデータを多く用意する必要がある．例えば網膜画像から年齢，性別，喫煙状況，血圧，主要心血管イベントなどを推定する研究では，約 28 万人分の患者のデータセットを学習データとして使用している [1]．

しかし病院が持つ個人の医療データや遺伝子データ，スマートフォンに保存されている画像などは，プライバシなどの理由からデータを手元に集められない場合がある．データを集められない場合，広域分散させたまま学習を行うことになる．

広域分散では，非同期的な通信方法の gossip を用いて学習モデルを共有する，自律分散な gossip SGD [2,3] が有力な手法である．クラスタを用いた，広域ではない通常の分散深層学習では，パラメータサーバを用いて学習モデルを共有する手法 [4–8] や，同期的な通信方法の all-reduce を用いて学習モデルを共有する all-reduce SGD [2,9–15] が主に研究されている．しかし，広域分散ではどちらの手法も適さない．広域分散では，クラスタを用いる分散深層学習と異なり，ネットワークの遅延が大きく，帯域幅が狭い．さらに，帯域幅やデータ，マシンの性能が非均質である．そのためパラメータサーバを用いる手法は，サーバとクライアントの通信がボトルネックとなる．また all-reduce SGD は同期的な通信方法を用いるため，非均質性を吸収することが困難であり，耐故障性の観点からも非同期的な通信方法を用いる gossip SGD に劣る．

我々は，帯域幅が非均質な場合に既存の gossip SGD を用いると，帯域幅が狭いノードの通信に輻輳が発生することを発見した．輻輳が発生する場合，帯域幅が広いノードを増やしても学習効率があまり良くならないことを実験結果を用いて明らかにする．我々は，この課題が学習時間に占める通信時間の割合によるものであり，通信頻度を変えて学習時間に占める通信時間の割合を下げることでこの課題が解決できることを示す．さらにモデルの精度を保ちながら学習効率を良くするために，通信時間に応じて通信先選択確率を変える手法を提案する．

本稿の構成は以下の通りである．2 章では，関連研究を紹介する．3 章では既存の gossip SGD と広域分散での課題を述べ，4 章でその原因を議論し，解決方法及び学習効率を良くする手法を提案する．5 章では実験を行い，4 章で述べた解決方法と提案手法の有効性を確かめる．最後に 6 章でまとめと今後の課題を述べる．

2 関連研究

広域分散環境での機械学習手法に gossip learning [16] 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，通信方法に gossip を用いている．さらに，これを改良した手法も提案されている [17]．しかしどちらもオンライン学習を用いる機械学習を対象としており、ミニバッチ学習が主流の深層学習は扱っていないため，本研究とは異なる．オンライン学習は一つの学習データでモデルを更新することを繰り返す手法であり，ミニバッチ学習は複数の学習データでモデルを更新することを繰り返す手法である．

広域分散ではない，クラスタを用いる分散深層学習手法は様々な提案がされている．Goyal らは，バッチサイズを大きくすることで学習効率を良くする手法を提案している [9]．バッチサイズを大きくする手法は他にも提案されている [10,11]．Lin らは，精度を下げずに通信する勾配を圧縮する手法を提案しており，最大 600 倍の圧縮比を達成している [13]．Shi らは，通信と計算をオーバーラップして学習時間を短くする手法である WFBP [12] を改良した，MG-WFBP を提案している [14]．Yu らは，信頼できないネットワークでも耐性のある手法を提案している [8]．他にも多くの手法が提案されている [2–7,15]．

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 [18] がある．これは集中な環境，つまりサーバの性能がクライアントより高いことを想定しているため，完全分散な

This is an unrefereed paper.

* 現所属：ヤフー株式会社

Algorithm 1 (pull) gossip SGD (run on client i)

```
function CLIENT_LEARN
    Initialize:
        w_{0,i} := w_0, t := 0
    loop
        shuffle(X_i)
        for minibatch x ∈ X_i do
            w_{t+1,i} ← w_{t,i} − α∇F_i(w_{t,i}; x)
            t ← t + 1
            if t ≡ 0 mod T then
                choose j at random
                receive(w_j)
                w_{t,i} ←average(w_{t,i}, w_j)
            end if
        end for
    end loop
end function
```

環境を想定する本研究とは異なる．また，広域分散での gossip SGD と all-reduce SGD を比較した研究もあるが [19]，帯域幅がすべてのノードで均質な場合のみを扱っているため，本研究とは異なる．

3 背　　景

本章では，まず既存の分散深層学習手法の一つである，gossip SGD を説明する．その後我々が発見した，広域分散で gossip SGD を用いる場合に発生する課題を述べる．

3.1 Gossip SGD

gossip SGD は，非同期的な通信方法の gossip を用いて学習モデルを共有する．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．

pull-gossip SGD の擬似コードを Algorithm 1 に示す．各ノードはそのノードが持っている学習用データ X_i を使用して，確率的勾配降下法（SGD）に基づき学習モデルを更新する．1 回の学習に使用するデータ minibatch のデータ数を，これ以降バッチサイズを呼ぶことにする．そして定期的に一様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．既存研究では $T = 1$ が用いられている [2]．

gossip SGD は他に push-gossip SGD という手法もあるが，4.2 節で述べる提案手法に適していない．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．そして一様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．

3.2 広域分散における gossip SGD の課題

gossip SGD は広域分散における有力な手法である．広域分散では，帯域幅やデータ，そしてマシンの性能が非均質である．このような非均質性と非同期的な通信方法を用いる gossip SGD は相性が良いと考えられる．

しかし我々は，帯域幅が非均質な広域分散で用いると，帯域幅が狭いノードの通信に輻輳が発生することを発見した．図（ここに輻輳の図）は 8 ノードで学習した際の各ノードの通信状況を表している．詳しい実験設定は 5 章で述べる．横軸は学習時間，縦軸はノード ID になっている．ノード ID が 6 より小さいノードは帯域幅が 10 Gbps と広く，それ以外のノードは帯域幅が 1.0 Gbps と狭い．この実験で用いた学習モデルの容量は 54 MiB であるため，1 回の通信時間は長い場合でも，

$$\frac{54 \times 2^{20} \times 8}{1.0 \times 10^9} \simeq 0.5\,[\mathrm{s}]$$

である．しかし，図 1a では明らかにこれ以上の通信時間があり，輻輳が発生していることがわかる．

また，このときの学習効率は図 1b のようになっている．凡例の Wide は帯域幅が広いノード数を表している．帯域幅が広いノードを増やしても，すべてのノードの帯域幅が狭いときの学習効率とあまり変わらないことが読み取れる．

このことから輻輳が発生する場合，帯域幅が広いノードを増やしても学習効率はあまり良くならないことがわかる．これが我々が発見した，広域分散における gossip SGD の課題である．また帯域幅が狭いノードとの通信で同期しているかのようになってしまっており，非同期的な通信方法を用いる gossip SGD の良さが活かされていないという点でも，解決すべき重要な課題であると考えられる．

4 輻輳の解消方法と warm-up

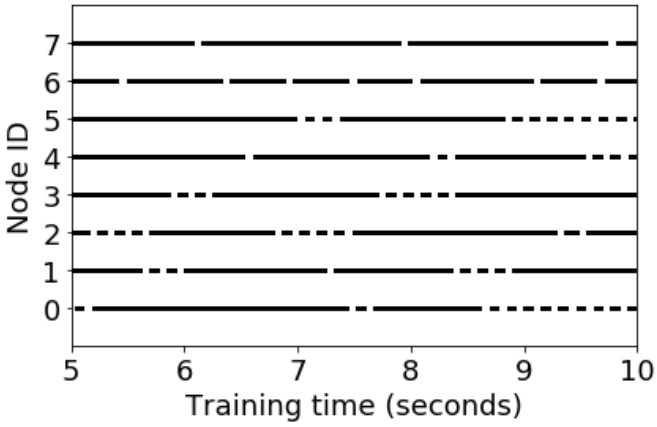
本章では，前章で述べた広域分散における gossip SGD の課題の原因とその解決策を示す．さらに精度を保ちながら学習効率を良くする手法を提案する．

4.1 通信頻度の必要条件

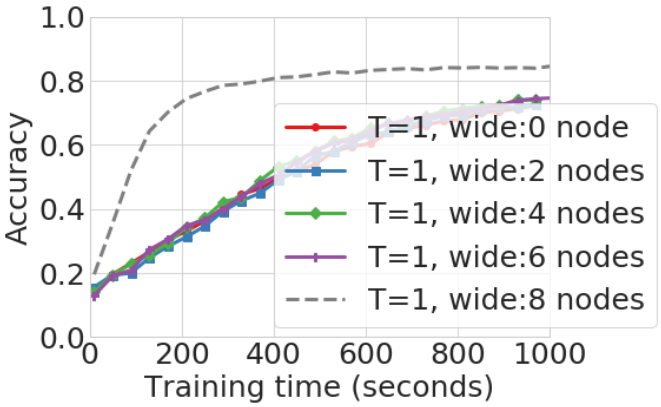
帯域幅が非均質な広域分散での gossip SGD では，帯域幅が広いノード同士の通信時間は短いが，帯域幅が広いノードと狭いノードとの通信時間は長い．よって帯域幅が狭いノードにおいて，帯域幅が広いノードとの通信が頻繁に行われ，通信時間が平均通信間隔より大きくなることがある．このとき，そのノードはいずれかのノードとの通信で待ち時間，すなわち輻輳が発生する．つまり，以下の条件を満たすノード i が存在するとき，輻輳が発生する．

$$\frac{1}{K-1}\sum_{j \neq i}\left(T \times \mathbb{E}\left[G_j\right] + \mathbb{E}\left[C_j\right]\right) < \mathbb{E}\left[C_i\right] \qquad (1)$$

ただし，$K, T, G, C, \mathbb{E}$ はそれぞれ，ノード数，通信頻度，損失関数の勾配計算にかかる時間，通信時間，期待値である．ここでは広域分散の場合を考えているので $K > 1$ である．(1) 式を見ると，輻輳を解消するには $T \times \mathbb{E}\left[G_j\right]$ を大きく，すなわち T



(a) wide:6 nodes の通信状況



(b) 学習曲線

図 1　既存の gossip

を大きくすればよいことがわかる．

したがって広域分散での gossip SGD で輻輳が発生しないためには，任意のノード i に対して以下の条件を満たすように通信頻度 T を大きくする必要がある．

$$T \geqq \frac{(K-1)\,\mathbb{E}\left[C_i\right] - \sum_{j \neq i} \mathbb{E}\left[C_j\right]}{\sum_{j \neq i} \mathbb{E}\left[G_j\right]} \tag{2}$$

4.2　Warm-up

我々は学習の収束を速めるために，学習の初期で通信時間に応じて通信先ノードの選択確率を変える warm-up 手法を提案する．深層学習では，精度を高めるためには学習用データを多くすることが必要である．そのため，ノードの選択確率が一様でないと，モデルの精度が低くなる可能性がある．

しかし学習の初期ではデータが少なくてもモデルの精度を上げることが可能である．つまり帯域幅が広いノードとの通信を頻繁に行い，通信時間を短くする warm-up 手法が有効だと考えられる．

push-gossip SGD の場合，帯域幅が狭いノードはモデルを受信する頻度が低くなることでモデルの更新も少なくなるため，この手法には適していない．一方 pull-gossip SGD の場合，帯域幅が狭いノードであってもモデルを受信する頻度は選択確率が一様ランダムの場合と変わらない．

我々は，各ノードが計測した通信時間情報を送信し合うことで，この情報を共有することにした．具体的にはモデルを通信する際に，互いが持つ通信時間情報を送信することで共有する．これは自ノードで計測した通信時間だけでは，自ノードより帯域幅が広いノードの帯域幅を認識できないためである．この手法では計測した通信時間を使用しているため，最も広い帯域幅は認識できないことに注意する必要がある．つまり帯域幅が広いノードが 1 ノードのみの場合は，すべてのノードの帯域幅が均質だと認識してしまう．

本研究ではネットワークは LAN (Local Area Network) から MAN (Metropolitan Area Network) の規模であり，通信のボトルネックが自ノードの通信性能にあることを想定している．

表 1　実 験 環 境

OS	Ubuntu 16.04.2 LTS
CPU	Intel Xeon E5-2698 v4
GPU	Tesla P100-PCIE-16GB

MAN より規模が大きく通信路の側にボトルネックがある場合，つまり遅延が大きいときの通信時間推定はこの推定手法では十分ではなく，今後の課題である．

提案手法の擬似コードを Algorithm 2 に示す．学習用データ X_i を使用して学習を進めていく部分は Algorithm 1 と同じである．通信先は一様ランダムではなく，選択確率 p に基づいて選択する．p_j はノード j との通信時間 $c_{i,j}$ の逆数である．ただし，自ノードと通信することはないため，$p_i \leftarrow 0$ としている．ノード i の通信時間情報 c_i は計測時間 c 及び，受信したノード j の通信時間情報 c_j を使って更新する．この更新はノード数が K のとき $\Theta(K)$ であり，十分高速である．また，ノード j にモデルのリクエストを送る際に c_i を送り，ノード j は c_j を更新する．

5　実　　験

本章ではまず，通信頻度 T を大きくしたときに帯域幅が狭いノードの通信で輻輳発生するかを調べる．次に 4.2 節で提案した warm-up 手法の効果を確かめる．

5.1　実 験 条 件

データセットは CIFAR-10，CIFAR-100 [20] を用いた．CIFAR-10 は 10 種類，CIFAR-100 は 100 種類のカラー画像からなるデータセットで，どちらのデータセットも学習用データが 50000 枚，テスト用データが 10000 枚である．ニューラルネットワークは VGG16 [21] を用いた．この後のすべての実験でノード数は 8 である．精度はすべてのノードのテスト用データに対する正答率の平均としている．また実験環境は表 1 の通りである．

Algorithm 2 (pull) gossip SGD+warm-up (run on client i)

function CLIENT_LEARN
  Initialize:
    $w_{0,i}$:=w_0, t:=0, c_i:={INF, INF, ..., INF}
  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**
        p:=$\left\{ \frac{1}{c_{i,k}} \right\}_{k=1}^{K}$
        $p_i \leftarrow 0$
        choose j on p
        send(c_i)
        receive(c_j, w_j)
        c:=measured_time()
        $c_{i,j} \leftarrow$UPDATE_TIME($c_{i,j}, c$)
        $c_i \leftarrow$UPDATE_TIMES(c_i, c_j)
        $w_{t,i} \leftarrow$average($w_{t,i}, w_j$)
      end if
    end for
  end loop
end function
function UPDATE_TIME(c_i, c_j)
  if $(1 - \text{threshold}) \times \text{c}_\text{i} > \text{c}_\text{j}$ **then**
    $c_i \leftarrow c_j$
  else if $(1 + \text{threshold}) \times \text{c}_\text{i} < \text{c}_\text{j}$ **then**
    $c_i \leftarrow \frac{c_i + c_j}{2}$
  end if
  return c_i
end function
function UPDATE_TIMES(c_i, c_j)
  for $k \in K$ **do**
    $c_{i,k} \leftarrow$UPDATE_TIME($c_{i,k}, c_{j,k}$)
  end for
  return c_i
end function

5.2 準　　備

実機で多数のノードを用意することや帯域幅を変えることが困難なため，シミュレータを作り実験を行った．このシミュレータは学習は GPU 上で行い，通信時間はシミュレーションを行った．損失関数の勾配計算には深層学習フレームワークである Chainer [22] を使用した．損失関数の勾配計算にかかる時間はばらつきが確認されたため，10 回実行して計測した時間の平均とした．

通信時間を決めるために，VGG16 の分散でない学習を行い，1 エポックごとに npz ファイルとしてモデルを圧縮保存して容量を調べた．npz ファイルは numpy 配列を保存する際にしばしば用いられる．10 エポックまで保存したが，いずれのモデルの容量も 54 MiB であった．帯域幅は狭いノードを 1.0 Gbps，広いノードを 10 Gbps に設定した．また，遅延はすべてのノードで 5.0 ms に設定した．

表 2　バッチサイズを変化させたときの精度

Batch size	Accuracy
64	87.1%
128	87.5%
256	87.4%
512	87.7%

(a) Wide: 0 node

Batch size	Accuracy
64	87.0%
128	87.1%
256	87.5%
512	87.5%

(b) Wide: 8 nodes

表 3　通信頻度を変化させたときの精度

通信頻度 T	Accuracy
1	87.4%
4	87.5%
8	87.4%
16	86.9%
32	86.2%

表 4　warm-up と既存 gossip の精度比較

Wide	Accuracy	
	Warm-up	Normal
2	87.0%	87.3%
4	87.3%	87.0%
6	86.9%	86.7%

(a) CIFAR-10

Wide	Accuracy	
	Warm-up	Normal
2	60.1%	60.3%
4	60.3%	60.2%
6	59.7%	59.6%

(b) CIFAR-100

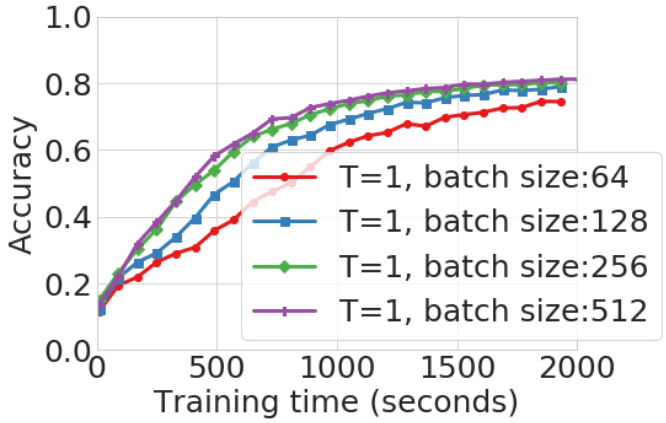
5.3 CIFAR-10

CIFAR-10 の実験では学習率を 0.3 に固定して行った．まず適切なバッチサイズを調べた．精度は表 2，学習曲線は図 2 のようになった．wide は帯域幅が広いノード数を表している．図 2a ではバッチサイズ 256 と 512 の学習効率が同じくらいだが，図 2b ではバッチサイズ 256 の学習効率が最も良い．精度はどちらも変わらないため，以降の実験ではバッチサイズを 256 に固定して行った．

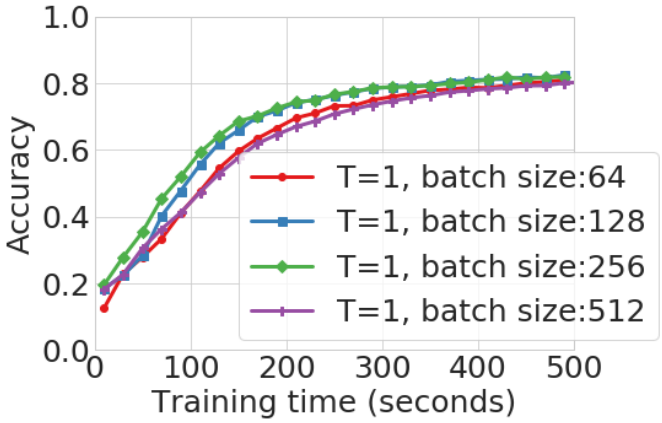
次に通信頻度 T を変えて輻輳が発生するか調べた．$T = 4$ のときの通信状況は図 3a のようになった．図 1a と比べると，全体的に通信時間が短くなっており輻輳が解消されていることがわかる．また学習曲線は図 3b のようになった．図 1b とは異なり，帯域幅が広いノードが多いほど学習効率が良いことがわかる．

そして精度が下がらない範囲で学習効率が最も良い通信頻度を調べた．各実験は一回のみの実行のため，精度が 1%下がった場合に精度が下がったとみなすことにした．すべてのノードの帯域幅が狭いとき，精度は表 3，学習曲線は図 4 のようになった．$T = 32$ のとき，精度が下がっている．また，学習効率は T が大きいほうが良い．よって $T = 16$ が適切な通信頻度である．

帯域幅が広いノードを増やして同様に実験を行った後，warm-up 手法の効果を確かめた．精度は表 4a，学習曲線は図 5a-7a のようになった．warm-up は最初の 1500 イテレーションで行っている．warm-up を行っても精度は低くなっていない．また，帯域幅の広いノードが増えるほど，warm-up の効果が大きくなり学習効率も良くなっていることがわかる．今回の実験では通

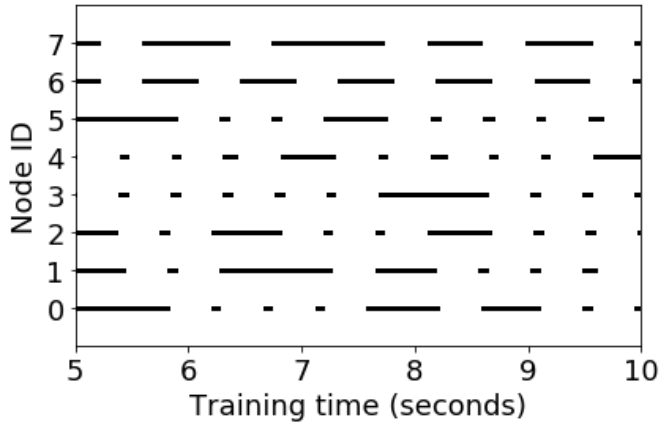


(a) Wide:0 node

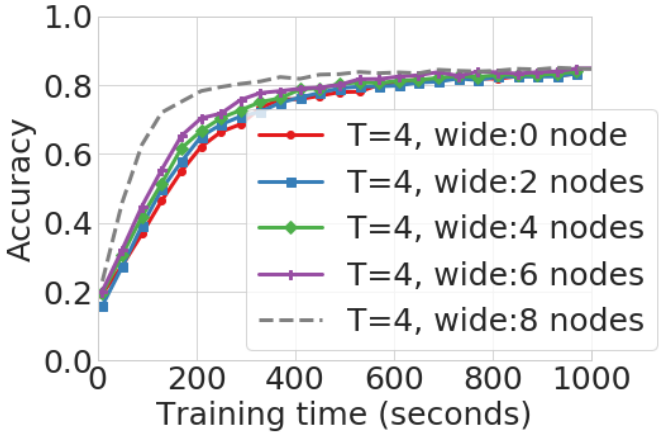


(b) Wide:8 nodes

図 2　バッチサイズの調整



(a) Wide:6 nodes の通信状況



(b) 学習曲線

図 3　$T = 4$ のときの gossip SGD

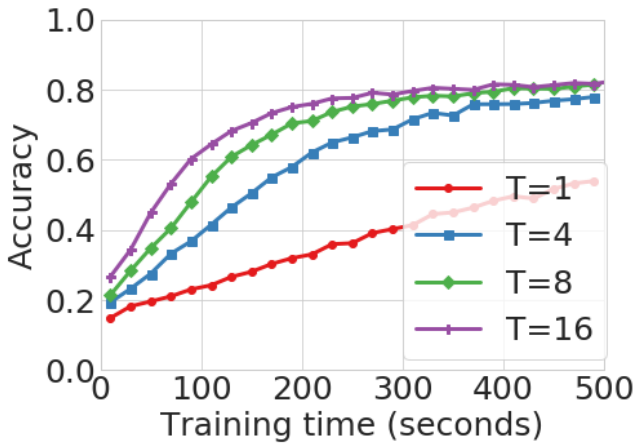


図 4　通信頻度を変化させたときの学習曲線

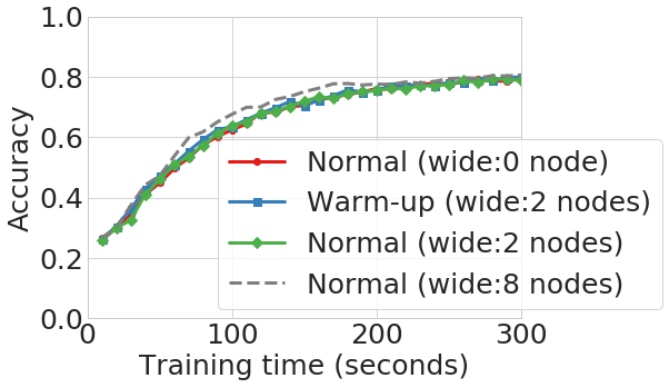
信頻度を調整してから warm-up 手法の調整をしているが，これらは同時に行うこともできる．

5.4　CIFAR-100

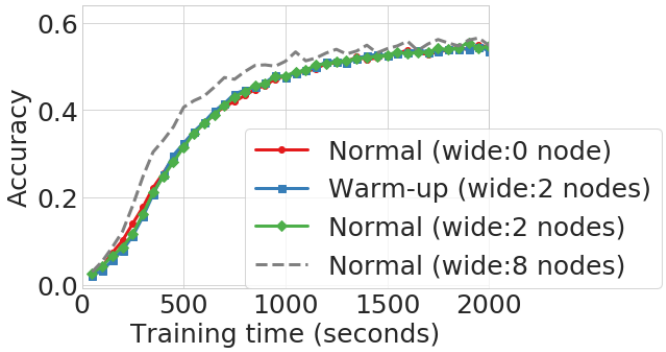
CIFAR-100 の実験では学習率を 0.1 に固定して行った．CIFAR-10 のときと同様にして，適切なバッチサイズと通信頻度の調整を行った．そして warm-up 手法の効果を確かめた．精度は表 4b，学習曲線は図 5b-7b のようになった．warm-up は最初の 5000 イテレーションで行っている．CIFAR-100 のときと同様，warm-up を行っても精度は低くなっておらず，学習曲線を見ると帯域幅が広いノードが増えるほど warm-up の効果が大きくなっていることがわかる．

6　まとめと今後の課題

本研究では，広域分散で帯域幅が非均質な場合に既存の gossip SGD を用いると，帯域幅が狭いノードの通信に輻輳が発生し，帯域幅が広いノードを増やしても学習効率があまり良くならないことを明らかにした．この課題が通信頻度を変えることで解決できることを示し，さらに通信先選択確率を変えることで収束を速くする warm-up 手法を提案した．実験により，通信頻度を低くして輻輳が解消された場合には，帯域幅の広いノードが多いほど学習効率が良いことを示した．帯域幅が広いノードと多く通信をする warm-up 手法により，精度を低くすることなく学習効率を良くできていることも確かめた．

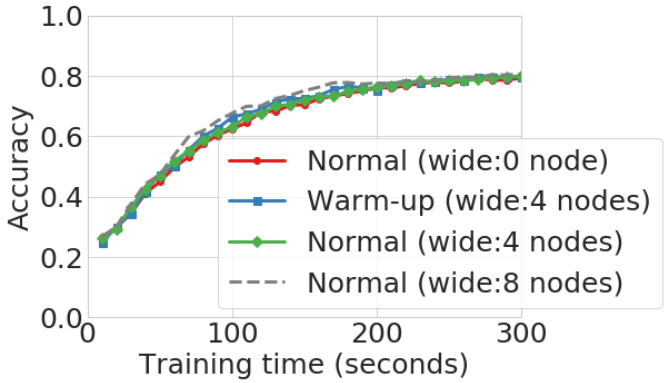


(a) CIFAR-10

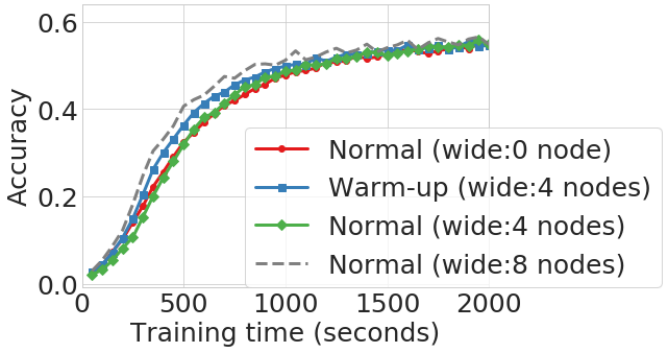


(b) CIFAR-100

図 5　warm-up と既存 gossip の学習曲線比較 (wide:2 nodes)

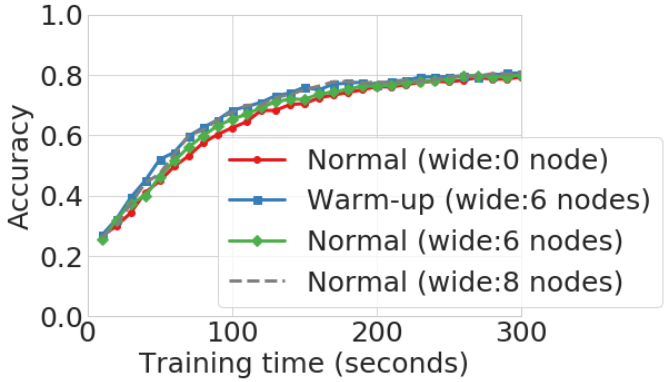


(a) CIFAR-10

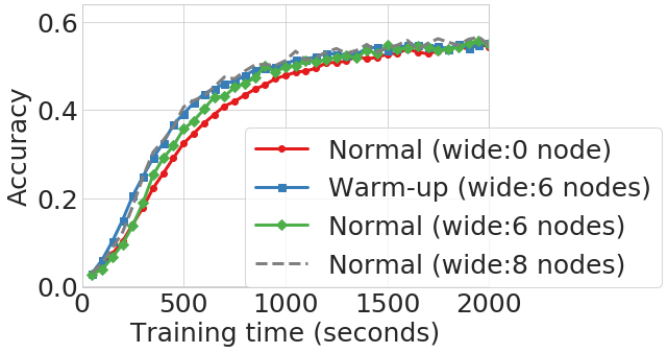


(b) CIFAR-100

図 6　warm-up と既存 gossip の学習曲線比較 (wide:4 nodes)



(a) CIFAR-10



(b) CIFAR-100

図 7　warm-up と既存 gossip の学習曲線比較 (wide:6 nodes)

今後の課題の一つは，データの分布やマシンの計算性能が非均質な場合の warm-up 手法である．このような場合，通信時間以外に計算時間も共有して warm-up を行うことが一つの方法として考えられる．

もう一つの課題は，悪意を持ったノードへの耐性である．本研究ではこのようなノードは想定していないため，学習効率が非常に悪くなることもあり得る．このようなノードが存在しても学習効率があまり悪くならないようなモデルの共有方法及び更新方法を考える必要がある．

謝　　辞

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] Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent. *arXiv preprint arXiv:1803.05880*, 2018.

[4] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, pp. 583–598, 2014.

[5] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: Near-linear acceleration of deep neural network training on compute clusters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] Henggang Cui, Hao Zhang, Gregory R. Ganger, Phillip B. Gibbons, and Eric P. Xing. Geeps: scalable deep learning on distributed gpus with a gpu-specialized parameter server. In *Proceedings of the Eleventh European Conference on Computer Systems, EuroSys 2016, London, United Kingdom, April 18-21, 2016*, pp. 4:1–4:16, 2016.

[7] Hanjoo Kim, Jaehong Park, Jaehee Jang, and Sungroh Yoon. Deepspark: Spark-based deep learning supporting asynchronous updates and caffe compatibility. *arXiv preprint arXiv:1602.08191*, Vol. 3, , 2016.

[8] Chen Yu, Hanlin Tang, Cédric Renggli, Simon Kassing, Ankit Singla, Dan Alistarh, Ce Zhang, and Ji Liu. Distributed learning over unreliable networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 7202–7212, 2019.

[9] 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.

[10] 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.

[11] 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.

[12] Ammar Ahmad Awan, Khaled Hamidouche, Jahanzeb Maqbool Hashmi, and Dhabaleswar K Panda. S-caffe: Co-designing mpi runtimes and caffe for scalable deep learning on modern gpu clusters. In *Acm Sigplan Notices*, Vol. 52, pp. 193–205. ACM, 2017.

[13] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

[14] Shaohuai Shi, Xiaowen Chu, and Bo Li. MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 172–180. IEEE, 2019.

[15] Masafumi Yamazaki, Akihiko Kasagi, Akihiro Tabuchi, Takumi Honda, Masahiro Miwa, Naoto Fukumoto, Tsuguchika Tabaru, Atsushi Ike, and Kohta Nakashima. Yet another accelerated SGD: ResNet-50 training on ImageNet in 74.7 seconds. *arXiv preprint arXiv:1903.12650*, 2019.

[16] 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.

[17] 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.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pp. 1273–1282, 2017.

[19] 小国英明, 高橋良希, 首藤一幸. 広域分散を想定した深層学習手法の比較. データ工学と情報マネジメントに関するフォーラム (DEIM), 2019.

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

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

[22] Seiya Tokui, Ryosuke Okuta, Takuya Akiba, Yusuke Niitani, Toru Ogawa, Shunta Saito, Shuji Suzuki, Kota Uenishi, Brian Vogel, and Hiroyuki Yamazaki Vincent. Chainer: A deep learning framework for accelerating the research cycle. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2002–2011. ACM, 2019.