DEIM2021 A14-2

広域分散での深層学習のための通信スケジューリング

小国　英明†　　首藤　一幸†

† 東京工業大学 〒 152–8550 東京都目黒区大岡山 2-12-1

あらまし　深層学習は多層のニューラルネットワークを用いる機械学習手法であり，画像認識や自然言語処理で高い性能を示している．深層学習では，学習モデルの精度を高めるために多くの学習用データが必要であるが，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．このような場合でも，学習モデルを通信し合うことで，データを広域分散させたまま学習を行うことができる．広域分散では，クラスタを用いる分散深層学習と異なり，帯域幅などの環境が非均質である．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，広域分散のような非均質な環境においても適用できる，計算と通信のオーバーラップを行う gossip SGD を提案する．提案手法は適切なタイミングで通信を行うことで，より新しい学習モデルを共有する．複数のデータセットと CNN で実験を行い，この手法の有効性を確かめる．

キーワード　広域分散，gossip，通信効率，深層学習

1 はじめに

深層学習は多層のニューラルネットワークを用いる機械学習手法である．物体検出と画像分類を競う ImageNet Large Scale Visual Recognition Challenge (ILSVRC) で AlexNet [1] が 2012 年に優勝して以来，深層学習は画像認識や自然言語処理など様々な分野で用いられている．

深層学習では，学習用データが多いほど，学習モデルの精度が高くなる．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用データとして使用している [2]．既存の多くの深層学習手法は，学習用データを 1 か所に集めることを必要とする．

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．この場合，学習用データを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，gossip という通信方法を用いて学習モデルを共有する gossip SGD [3–5] が有力な手法である．gossip は非同期的な通信方法であり，スケーラビリティが高い．

広域分散ではない，クラスタを用いた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 [6–10] 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 [4,11–16] が研究されている．しかし，広域分散ではどちらの手法も適していない．広域分散では，クラスタを用いる分散深層学習とは異なり，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，学習用データ，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，クライアントが増加すると，サーバとクライアント間の通信がボトルネックとなる．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，対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．

クラスタを用いる分散深層学習では，通信が課題となることが多い．同期的な通信方法を用いる分散深層学習では，ノードでの計算とモデルの通信をオーバーラップすることでみかけの通信時間を短くする double buffering という手法がある．通信時間が短くなると，学習時間，つまり学習の収束が早くなる．

広域分散においても，通信時間は課題の一つである．本研究では，計算と通信のオーバーラップを行う gossip SGD を提案する．gossip SGD は非同期的な通信を用いるため，通信開始時間を適切に決めて計算と通信のオーバーラップを行うことは難しい．ノード間で通信時間と通信可能ノードを共有することでこれを実現する．情報の共有方法は，通信管理ノードを使う方法と，自律分散で行う方法の 2 通りを述べる．通信管理ノードは学習モデルのような大きなデータの通信は行わないため，パラメータサーバのように通信がボトルネックとなることはない．この手法は広域分散のような非均質な環境でも適用できる．

本稿の構成は以下の通りである．2 章では，本研究に関連する分散深層学習の既存手法を紹介し，3 章で関連研究を述べる．4 章で提案手法を述べ，5 章の実験でその効果を確かめる．最後に，6 章でまとめを述べる．

2 準　　備

本章では，まず非同期的な通信を用いる分散深層学習の既存手法である，gossip SGD を説明する．次に，all-reduce SGD のような同期的な通信を用いる分散深層学習でしばしば使われる，double buffering という手法を述べる．

2.1 確率的勾配降下法と記法

深層学習の多くの手法では，確率的勾配降下法（Stochastic Gradient Descent, SGD）あるいはその応用に基づき，学習モデルを最適化する．SGD は学習用データ x を使用して，学習

Algorithm 1 Gossip SGD (pull; 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
```

モデル w_t を次のように更新するアルゴリズムである．

$$w_{t+1} \leftarrow w_t - \alpha \frac{1}{N} \sum_{n=1}^{N} \nabla f(w_t; x_n)$$

w の添え字はイテレーションの数を表している．すなわち，w_t は t 番目のイテレーションの学習モデルである．α は学習率あるいはステップサイズと呼ばれる定数である．N は一度のモデル更新で用いるデータ数であり，バッチサイズと呼ばれる．α，N はハイパーパラメータであり，学習モデルの収束速度や精度に大きく関わる．また，f は損失関数と呼ばれる関数であり，交差エントロピー誤差をしばしば用いる．

本稿では，K (> 1) ノードが深層学習する状況を考える．t イテレーション目にノード i が持つ学習モデルを $w_{t,i}$ と表記する．また，ノード i の損失関数の勾配平均を $\nabla F_i(w_{t,i})$ と表記し，これを損失関数の勾配とみなすことにする．つまり，

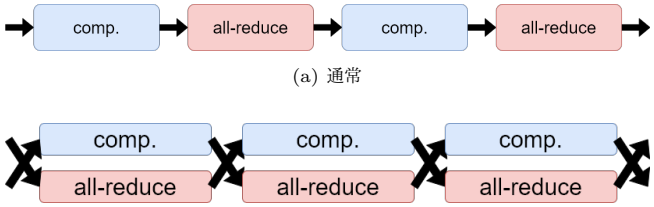
$$\nabla F_i(w_{t,i}) = \frac{1}{N} \sum_{n=1}^{N} \nabla f(w_{t,i}; x_n)$$

である．

2.2 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 は非同期的に自律的な通信を行い，帯域的な情報を緩く共有する方法である．

Algorithm 1 は pull-gossip SGD の擬似コードである．各ノードはそのノードが持つ学習用データ X_i 使用して，SGD に基づき学習モデルを更新する．gossip SGD では，1 回のモデル更新に用いるデータをミニバッチと呼ぶ．また，ノード全体で 1 回のモデル更新に用いるデータをバッチと呼び，バッチサイズはミニバッチサイズのノード数倍である．各ノードでモデル更新を T 回繰り返した後，一様ランダムに選んだノード j と非同期通信を行って学習モデルを受信し，二つの学習モデルの平均をとる．モデルの平均は w_i と w_j の各パラメータの平均である．このような更新を繰り返して，ノード全体のデータを反映した学習モデルを共有する．T は各ノードが他のノードと学習モデルを共有するための通信頻度である．広域分散ではない，クラスタを用いる gossip SGD の既存研究では，Jin らが $T = 1$ を用いている [4]．



(a) 通常

(b) Double buffering

図 1　通常の all-reduce SGD と double buffering の比較

push-gossip SGD は別の gossip SGD であるが，4 章で述べる提案手法に適していない．push-gossip SGD は pull-gossip SGD と同様に，各ノードが持つ学習用データを使用して学習モデルを更新する．そして各ノードは定期的に，一様ランダムに選んだノードと非同期通信を行って学習モデルを送信し，ノードでのモデル更新を行っている間に受信した複数の学習モデルとそのノードが持つ学習モデルの平均をとる．

2.3 Double buffering

double buffering はデータ通信の時間を隠蔽する手法である．分散深層学習に限った手法ではないが，本研究では all-reduce SGD での double buffering を説明する．

all-reduce SGD は各ノードでの損失関数の勾配計算と，all-reduce による全ノードの勾配の平均化を繰り返す手法である．全体としてのモデル更新を数式で表すと

$$w_{t+1} \leftarrow w_t - \alpha \frac{1}{K} \sum_{i=1}^{K} \nabla F_i(w_t)$$

である．all-reduce SGD では全ノードが同じ学習モデルを持つため，w の添え字からノード i を省略している．

double buffering は各ノードで損失関数の勾配計算をするのと並行して，一つ前のイテレーションでの勾配を all-reduce により平均化する．全体としてのモデル更新を数式で表すと

$$w_{t+1} \leftarrow w_t - \alpha \frac{1}{K} \sum_{i=1}^{K} \nabla F_i(w_{t-1})$$

である．∇F_i を計算する際のモデル w の添え字が，t から $t-1$ に変わっていることに注意が必要である．double buffering は通信時間が短くなるが，一つ前のイテレーションの勾配を使ってモデル更新をするため，精度が下がる可能性もある．

図 1 は通常の all-reduce SGD と double buffering を行う all-reduce SGD の学習の様子を図示したものである．“comp.” は各ノードでの損失関数の勾配計算を表し，“all-reduce” は all-reduce による勾配の平均化を表す．通常の all-reduce SGD は勾配計算と all-reduce を交互に行うため，図 1a のように見かけの通信時間と実際の通信時間は等しい．一方，double buffering を用いると図 1b のように，見かけの通信時間は計算時間の分だけ短くなる．

3 関連研究

通信方法に gossip を用いる既存の機械学習手法はいくつか存在する．gossip learning [17] は広域分散環境での機械学習手法である．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．しかし，SVM などのオンライン学習が対象であり，ミニバッチ学習を用いる深層学習には適用できない．gossiping SGD [4] は深層学習が対象であり，通信方法に gossip を用いている．しかし広域分散ではなく，均質なクラスタを用いることを想定しているという点で，本研究と異なる．edge-consensus learning [18] は P2P ネットワーク上での深層学習が対象であり，通信方法に gossip を用いている．edge-consensus learning はデータの分布が独立同分布でない場合でも，学習モデルが収束するように，学習モデルの更新式を工夫している．提案手法と組み合わせることで，学習時間で比較したときの学習の収束が早くなると期待できる．我々の以前の研究 [19] でも，広域分散で有効な gossip SGD を提案している．この手法は，帯域幅が広いノードとの通信を増やすことで通信時間を短くする．一方提案手法は，計算と通信のオーバーラップを行うことで通信時間を短くする．

Federated Learning [20] は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である．Federated Learning はパラメータサーバを用いる手法であり，サーバの性能がクライアントの性能より高いことを想定している．そのため，本研究の対象である非集中な深層学習には適していない．

Hop [21] と Prague [22] は一時的なノードの性能低下を考慮した研究である．Hop は同期的な通信を用いる手法であり，Prague は非同期的な通信を用いる手法である．しかし，つねに通信が非均質であるような環境は考慮しておらず，本研究とは異なる．

all-reduce SGD では，計算と通信のオーバーラップを行う既存手法が存在する [23–25]．1 章で述べたように，同期的な通信を用いる all-reduce SGD は広域分散では適していない．

4 提案手法

本章では，計算と通信のオーバーラップを行う gossip SGD を提案する．

4.1 gossip SGD での計算と通信のオーバーラップ

gossip SGD はノードでのモデル更新を複数回行ってから，学習モデルを受信する．そのため，2 章で述べた double buffering をそのまま適用することはできない．

我々は，学習モデルの受信開始時間を適切に設定することで，計算と通信のオーバーラップをしつつ，可能な限り新しい学習モデルと平均化する手法を提案する．

図 2 は通常の gossip SGD と提案手法の学習の様子を図示したものである．“comp.” は各ノードでのモデル更新を表し，

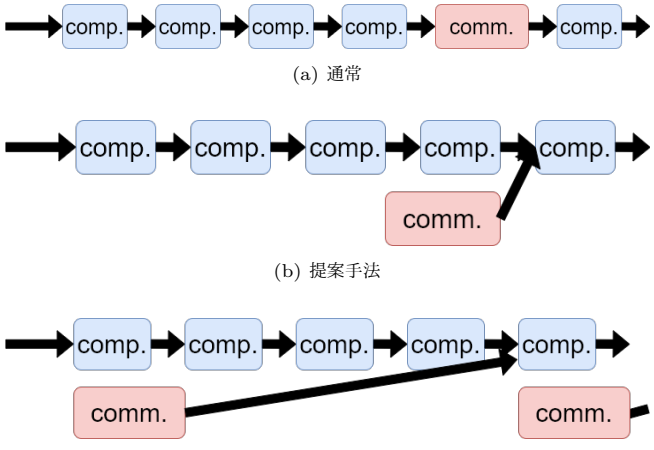


図 2　既存の gossip SGD，提案手法および naive な手法の比較

“comm.” は学習モデルの受信を表す．通常の gossip SGD は図 2a のように，モデル更新の後に学習モデルを受信する．一方，提案手法は図 2b のように，各ノードでのモデル更新と学習モデルの受信を並行して行う．ただし通信開始時間は，モデル更新が終わる時間に間に合う最も遅い時間に設定する．また，図 2c は最初のモデル更新と並行して学習モデルの受信を行う手法を図示したものである．この手法は 5 章で提案手法と比較する．提案手法と比べると，古い学習モデルと平均化することになり，学習の収束が遅くなると考えられる．

広域分散では，各ノードの帯域幅などが非均質である．そのため，提案手法で通信開始時間を適切に決めるためには，ノード間で情報を共有する必要がある．必要な情報は計算時間，通信時間，通信可能なノードの三つである．我々は，通信管理ノードを使う方法と，自律分散で行う方法の 2 通りで，これを実現する．

4.2 通信管理ノード

本節では，通信管理ノードを使う方法を述べる．この通信管理ノードは Prague の Group Generator [22] から着想を得ている．

通信開始時間を決めるために必要な情報のうち，計算時間と通信時間は各ノードで計測できる．しかし，通信時間も通信管理ノードが持つ方が都合が良い．通信管理ノードは通信時間と通信可能なノードを持つことにする．

Algorithm 2 は通信管理ノードの擬似コードである．ノード i は計測した計算時間をもとに，モデル更新が終わる時刻である end_time を予測し，通信管理ノードにその情報を送る．通信管理ノードは GET_START_TIME を実行して，通信可能なノード node_queue から通信先 j を選び，通信を開始すべき時間とともにノード i に返信する．通信を開始すべき時間はノード i, j の推定通信時間 $c_{i,j}$ から求める．ただし，通信可能なノードが存在しない場合は，一つ以上のノードが通信可能になるのを待つ．ノード i は通信が終了したら，通信管理ノードに計測した通信時間を送る．そして，通信管理ノードはその情報をもとに，UPDATE_INFORMATION を実行して，通信可

Algorithm 2 Communication Management Node

Initialize:
  node_queue $\coloneqq \{i\}_{i=1}^{K}$, $c \coloneqq \{\mathrm{INF}\}_{i,j=1}^{K}$
function GET_START_TIME(i, end_time)
  $j \leftarrow$ node_queue.front()
  node_queue $\leftarrow$ node_queue $\setminus \{j\}$
  start_time $\leftarrow$ end_time $- c_{i,j}$
  return j, max(current_time, start_time)
end function
function UPDATE_INFORMATION(i, j, comm_time)
  node_queue $\leftarrow$ node_queue $\cup \{j\}$
  $c_{i,j} \leftarrow$UPDATE_TIME($c_{i,j}$, comm_time)
  $c_{j,i} \leftarrow$UPDATE_TIME($c_{j,i}$, comm_time)
end function
function UPDATE_TIME($c_{i,j}$, comm_time)
  if $(1 - \mathrm{THRESHOLD}) \times c_{i,j} >$ comm_time **then**
    return comm_time
  else if $(1 + \mathrm{THRESHOLD}) \times c_{i,j} >$ comm_time **then**
    return $\dfrac{c_{i,j} + \mathrm{comm\_time}}{2}$
  else
    return comm_time
  end if
end function

能なノード node_queue と推定通信時間 $c_{i,j}$, $c_{j,i}$ を更新する．$c_{i,j}$ と計測した通信時間が大きく異なる場合は，$c_{i,j}$ を計測した通信時間に更新する．それ以外の場合は，$c_{i,j}$ を，$c_{i,j}$ と計測した通信時間の平均値に更新する．$c_{j,i}$ も同様にして更新する．

学習を行うノードと通信管理ノード間では，学習モデルの通信のような大きなデータの通信は行わない．そのためパラメータサーバを用いる手法とは異なり，広域分散であっても通信がボトルネックとなることはない．

4.3 自律分散

本節では，自律分散で行う方法を述べる．前節で述べた方法では，通信管理ノードのみが通信時間と通信可能なノードを知っていればよい．しかし，自律分散で情報を共有するには，全ノードが通信時間と通信可能なノードを知っている必要がある．つまり，各ノードが前節の通信管理ノードと同様に，計算時間と通信可能なノードの管理を行う．通信時間に関しては，各ノードは自分の通信時間のみ必要なため，ノード間で共有する必要がないことに注意が必要である．

自律分散では，複数のノードが同じ通信先 j を選ぶことがある．そのような場合は，ノード j が通信相手を一つ選ぶ．ノード j に選ばれなかったノードは別のノードを通信可能なノードから選ぶことになる．

前節の方法は，通信管理ノードが単一故障点になっているが，自律分散であれば単一故障点は存在しない．自律分散で行う方法の欠点は，通信先を選び直す場合があることと，ノード間の通信が増えることである．そのため，通信管理ノードを使う方法より学習効率が悪くなる可能性がある．

表 1 実験環境

OS	Ubuntu 20.04.1 LTS
Kernel	Linux 4.4.0-79-generic
CPU	Intel Xeon E5-2698 v4
GPU	Tesla P100-PCIE-16GB

5 実　　験

本章では，二つのデータセットを使った実験を行い，4 章で述べた提案手法の有効性を確かめる．実験では，次の三つの手法を比較する．

（1） 計算と通信のオーバーラップを行わない，既存の gossip SGD

（2） 4 章で述べた，学習モデルの受信開始時間を適切に設定し，計算と通信のオーバーラップを行う提案手法

（3） 最初のモデル更新と並行して学習モデルの受信を行うことで，naive に計算と通信のオーバーラップを行う gossip SGD

提案手法は通信管理ノードを使う方法と，自律分散で行う方法の 2 通りの実験を行う．

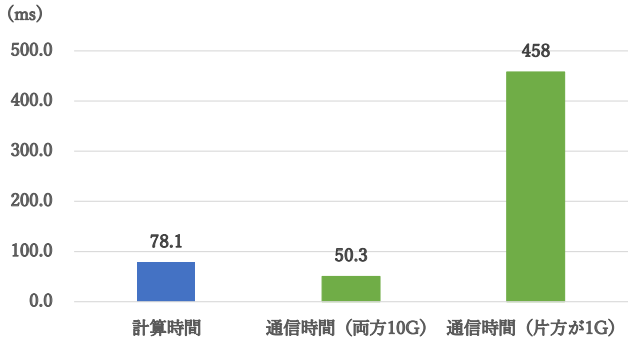
5.1 実験設定

データセットは CIFAR-100 [26] と Fashion-MNIST [27] を用いる．CIFAR-100 は 100 種類のカラー画像からなるデータセットである．1 つのデータは RGB3 チャネルの 32×32 の画像である．学習用データが 50,000 枚，テスト用データが 10,000 枚である．学習用データは各ノードに均等に分配する．Fashion-MNIST は 10 種類の服の白黒画像からなるデータセットである．1 つのデータは 28×28 の画像である．学習用データが 60,000 枚，テスト用データが 10,000 枚である．Fashion-MNIST を用いる実験でも，学習用データは各ノードに均等に分配する．ニューラルネットワークは CIFAR-100 では VGG16 [28]，Fashion-MNIST では Network in Network [29] という畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）を用いる．すべての実験で学習を行うノード数は 8 であり，通信管理ノード使う実験ではこのほかに 1 ノード置いている．精度は各ノードのテスト用データに対する正答率の平均である．また，実験環境は表 1 の通りである．

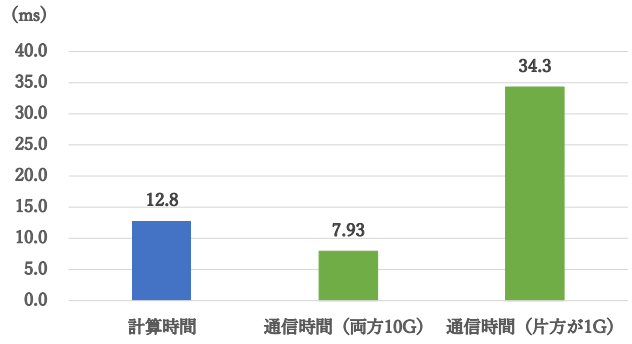
5.2 実験準備

各ノードの帯域幅を変えるために，シミュレータで実験を行う．このシミュレータは学習は実際に GPU 上で行い，通信時間をシミュレーションするものである．損失関数の勾配計算には深層学習フレームワーク Chainer [30] を使用する．勾配計算にかかる時間はばらつきがあったため，10 回実行して計測した時間の平均とする．

通信時間を決めるために，VGG16 と Network in Network を `npz` 形式で毎エポック学習モデルを圧縮保存して容量を調べた．`npz` 形式は numpy 配列を保存する際にしばしば用いられる．10 エポックまで計測したところ，VGG16 はいずれも



(a) VGG16



(b) Network in Network

図 3　計算時間と通信時間の比較

表 2　CIFAR-100 を用いた実験での提案手法と既存 gossip の精度比較

Wide	Accuracy		
	Proposed (manage)	Proposed (no manage)	No overlap
0 node	58.2%	58.3%	57.4%
2 nodes	57.8%	57.8%	57.4%
6 nodes	57.5%	57.7%	57.3%

54 MiB，Network in Network はいずれも 3.5 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 に設定する．ネットワークの遅延はすべてのノードで 5.0 ms とする．

5.3　CIFAR-100

CIFAR-100 では学習率を 0.1 に固定して実験した．まず，ミニバッチサイズと通信頻度 T の調節を行い，適切なミニバッチサイズは 256，通信頻度は $T = 16$ であった．このとき，1 イテレーションの計算時間と通信時間の関係は図 3a のようになっている．帯域幅が広いノード同士では，通信時間は計算時間より短い．しかし，それ以外の場合では，通信時間は計算時間と比べて非常に長くなっている．今回は適切な通信頻度が $T = 16$ であるため，通信は計算で完全にオーバーラップできる．

次に，提案手法，naive に計算と通信のオーバーラップをする手法，そして既存の gossip SGD を比較した．10,000 秒までの最高精度は表 2 のようになった．“Wide” は帯域幅が広いノード数，“No overlap” は既存手法を表す．“manage” は通信管理ノードを使う提案手法，“no manage” は自律分散で行う提案手法である．帯域幅が広いノード数がいくつのときでも，提案手法は既存手法と比べて精度は下がっていない．さらに，提案手法はどちらの方法でも精度はほぼ同じであることも分かる．また，学習曲線は図 4-6 のようになった．横軸は学習時間，縦軸は精度である．“Naive overlap” は naive に計算と通信のオーバーラップをする手法である．比較をしている三つの手法のうち，全ノードの帯域幅が広いときの学習曲線に最も近いのは提案手法である．この実験では，適切な通信頻度が $T = 16$ であるため，通信先を決めるのにある程度余裕がある．そのため，自律分散で行う方法もよい性能を示している．また，naive な手法は逆に最も悪い結果になっており，提案手法のように新しい学習モデルと平均化することが重要であることが分かる．さらに，今回は通信は計算で完全にオーバーラップできるため，naive な手法は帯域幅の広いノードが増えても収束は早くならない．一方，提案手法は帯域幅の広いノードが増えるほど，通信開始時間が遅くできる．つまり，新しい学習モデルと平均化する機会が増えるため，収束も早くなる．

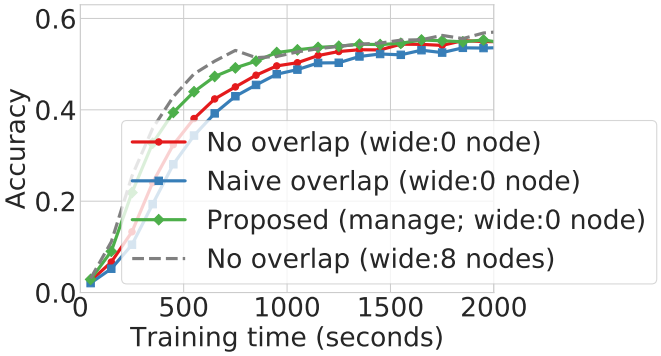
表 3　Fashion-MNIST を用いた実験での提案手法と既存 gossip の精度比較

Wide	Accuracy		
	Proposed (manage)	Proposed (no manage)	No overlap
0 node	86.5%	86.4%	86.0%
2 nodes	86.6%	86.7%	86.1%
6 nodes	86.6%	86.7%	86.2%

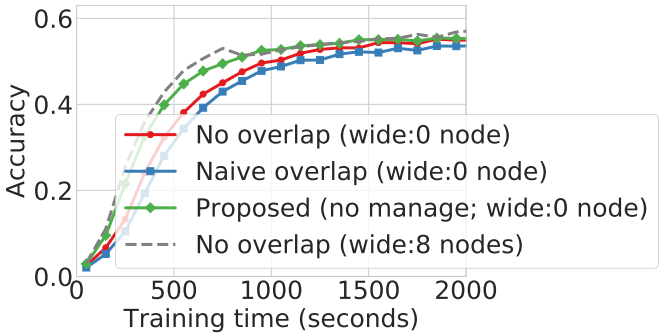
5.4　Fashion-MNIST

Fashion-MNIST では学習率を 0.01 に固定して実験した．最初にバッチサイズと通信頻度の調整を行い，適切なミニバッチサイズは 16，通信頻度は $T = 16$ であった．このとき，1 イテレーションの計算時間と通信時間の関係は図 3b のようになっている．VGG16 の場合と比べると，通信時間に対する計算時間が長い．そのため，計算と通信のオーバーラップの効果も小さくなることが予想できる．計算時間が長くなっているのは，Network in Network のパラメータ数の少なさとミニバッチサイズの小ささが関係していると考えられる．実際，ミニバッチサイズが 32 のときも計算時間はほぼ同じであった．5.2 節で述べたように，Network in Network の容量は VGG16 より小さく，パラメータ数も少ない．パラメータ数が少ない場合，学習率とミニバッチサイズを小さくすると安定して学習が収束する傾向にある．

次に，提案手法，naive に計算と通信のオーバーラップをする手法，そして既存の gossip SGD を比較した．1,000 秒までの最高精度は表 3 のようになった．CIFAR-100 のときと同様，提案手法は既存手法と比べて精度は下がっていない．こちらの実験でも，どちらの提案手法の精度もほぼ同じである．また，学習曲線は図 7-9 のようになった．既存手法が上回っている箇所もあるものの，提案手法の方が収束が早く，全ノードの帯域幅が広いときの学習曲線に近いことが多い．この実験でも，通

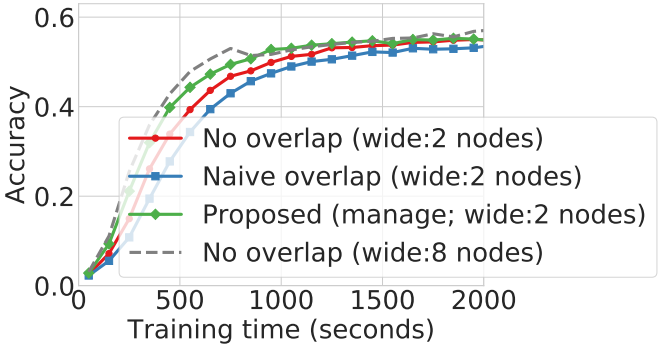


(a) 通信管理ノードを使う方法

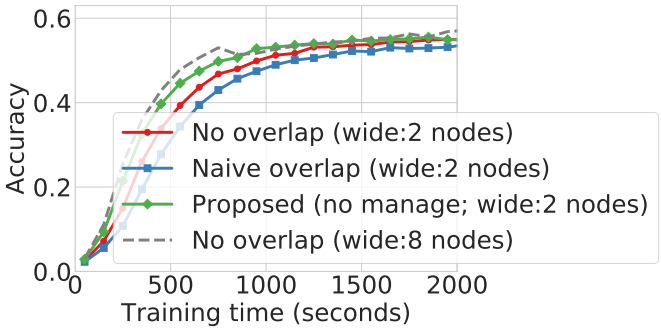


(b) 自律分散で行う方法

図 4　CIFAR-100 を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:0 node)

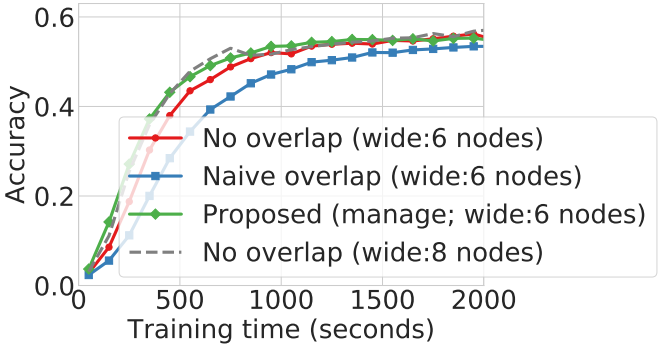


(a) 通信管理ノードを使う方法

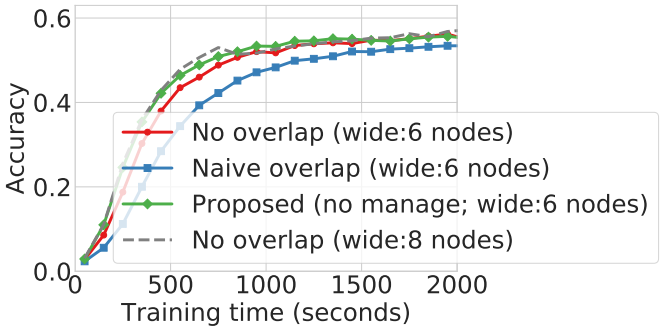


(b) 自律分散で行う方法

図 5　CIFAR-100 を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:2 nodes)



(a) 通信管理ノードを使う方法



(b) 自律分散で行う方法

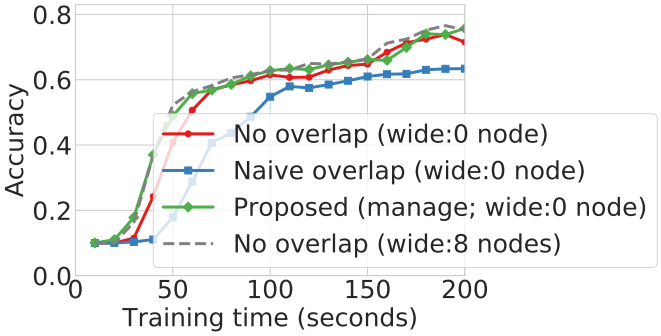
図 6　CIFAR-100 を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:6 nodes)

信先を決めるのにある程度余裕があるため，自律分散で行う方法もよい性能を示している．naive な手法が最も悪い結果になっているのは CIFAR-100 と同様である．

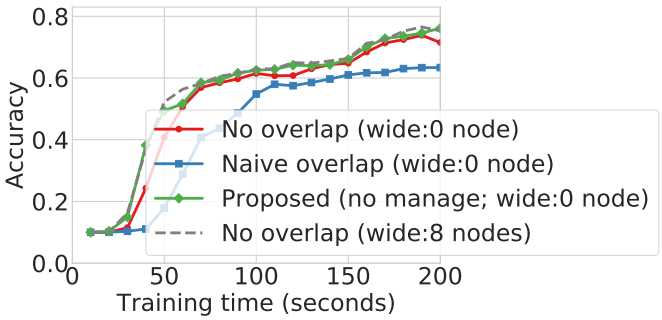
6　ま　と　め

本研究では，広域分散のような非均質な環境でも適用できる，計算と通信のオーバーラップを行う gossip SGD を提案した．gossip SGD は非同期通信を用いるため，通信開始時間を適切に定めることが難しい．提案手法は，ノード間で通信時間と通信可能なノードを共有することでこれを実現する．情報の共有方法は，通信管理ノードを使う方法と，自律分散で行う方法の 2 通りである．前者の方法では，通信管理ノードが通信可能なノードを持っており，通信先と通信を開始すべき時間の指示を行う．後者の方法では，各ノードが通信可能なノードを持っており，通信先の選択は各ノードが行う．自律分散で行う方法は単一故障点が存在しないことが利点である．欠点は，通信先を選び直す場合があることと，ノード間の通信が増えることである．

実験では，二つのデータセットを使って，精度と学習効率を既存手法と比較した．計算と通信のオーバーラップを行うと精度が下がる可能性があるが，提案手法の精度は既存手法とほぼ同じであった．さらに，提案手法は既存手法より収束が早く，良い学習効率を示した．自律分散で行う提案手法は，通信管理

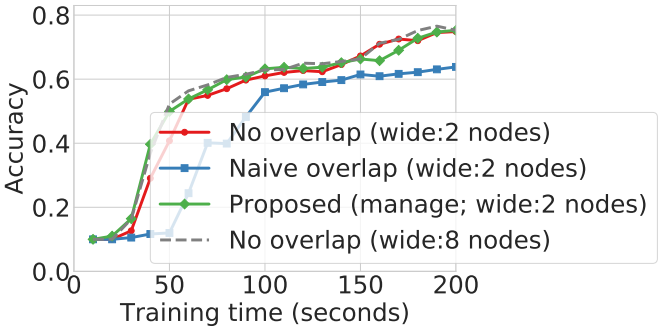


(a) 通信管理ノードを使う方法

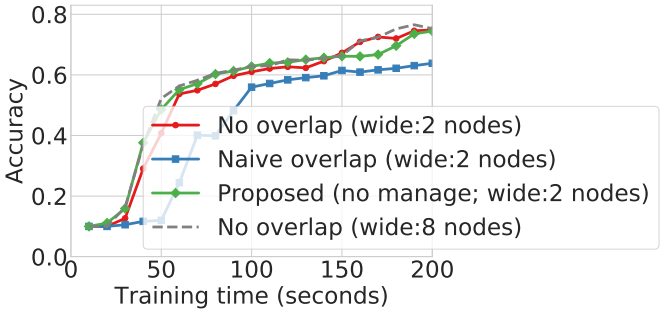


(b) 自律分散で行う方法

図 7　Fashion-MNIST を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:0 node)

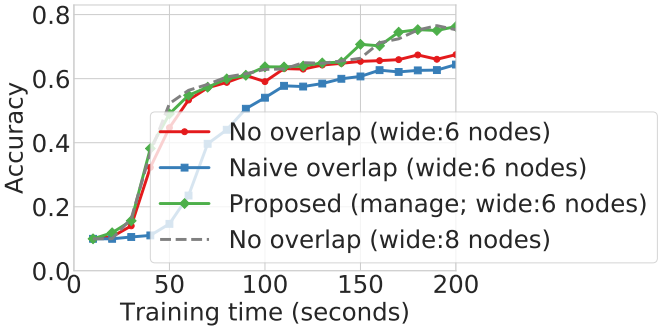


(a) 通信管理ノードを使う方法

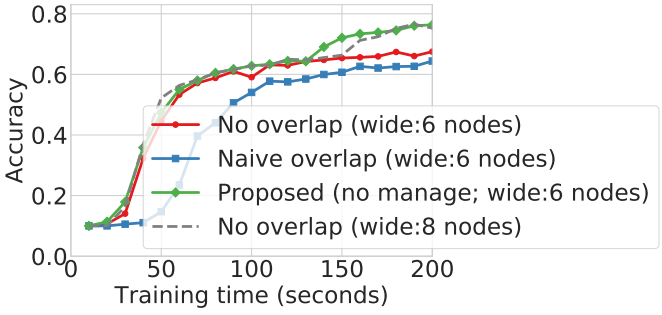


(b) 自律分散で行う方法

図 8　Fashion-MNIST を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:2 nodes)



(a) 通信管理ノードを使う方法



(b) 自律分散で行う方法

図 9　Fashion-MNIST を用いた実験での提案手法と既存 gossip の学習曲線比較 (wide:6 nodes)

ノードを使う提案手法より悪い性能を示す可能性もある．しかし，どちらのデータセットを使った実験でも通信管理ノードを使う提案手法と同等の性能を示した．また，naive に計算と通信のオーバーラップを行う手法は既存手法に劣ることを示し，通信開始時間を適切に決めることが重要であると分かった．

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] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, Vol. 60, No. 6, pp. 84–90, 2017.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Kenta Niwa, Noboru Harada, Guoqiang Zhang, and W Bastiaan Kleijn. Edge-consensus learning: Deep learning on P2P networks with nonhomogeneous data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 668–678, 2020.

[19] Hideaki Oguni and Kazuyuki Shudo. Addressing the heterogeneity of a wide area network for DNNs. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021.

[20] 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.

[21] Qinyi Luo, Jinkun Lin, Youwei Zhuo, and Xuehai Qian. Hop: Heterogeneity-aware decentralized training. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 893–907, 2019.

[22] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 401–416, 2020.

[23] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[24] Dipankar Das, Sasikanth Avancha, Dheevatsa Mudigere, Karthikeyan Vaidynathan, Srinivas Sridharan, Dhiraj Kalamkar, Bharat Kaul, and Pradeep Dubey. Distributed deep learning using synchronous stochastic gradient descent. *arXiv preprint arXiv:1602.06709*, 2016.

[25] Jianyu Wang, Hao Liang, and Gauri Joshi. Overlap local-SGD: An algorithmic approach to hide communication delays in distributed SGD. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8871–8875. IEEE, 2020.

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

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

[29] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[30] 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.