

DeepL Proに登録して、プレゼン資料を編集しましょう  
詳しくは[www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)をご覧ください。

PowerSGD：実用的なローランク

分散型最適化のための勾配圧縮

Thijs VogelsSai Praneeth KarimireddyMartin Jaggi

EPFLEPFLEPFL

ローザンヌ（スイス） ローザンヌ（スイス） ローザンヌ（スイス） thijs.vogels@epfl.ch sai.karimrieddy@epfl.ch martin.jaggi@epfl.ch

# アブストラクト

本研究では、データ並列分散最適化における通信のボトルネックを軽減するための非可逆勾配圧縮法を研究しています。現在の圧縮方式は、注目されているにもかかわらず、スケーリングがうまくいかなかったり、目標とするテスト精度を達成できなかったりする。本研究では，パワーイテレーションに基づく新しい低ランク勾配圧縮器を提案する．この圧縮器は，i) 勾配を高速に圧縮し，ii) 圧縮された勾配をall-reduceを用いて効率的に集約し，iii) SGDと同等のテスト性能を達成することができる．提案されたアルゴリズムは、高度に最適化された市販の分散通信ツールを用いて通常のSGDとベンチマークした際に、壁時計のように一貫した速度を達成した唯一の手法です。また、一般的なデータセットにおいて、畳み込みネットワークやLSTMの学習時間が短縮されたことを実証しました。このコードは <https://github.com/epfml/powersgd>で公開されています[。](https://github.com/epfml/powersgd)

# はじめに

同期データ並列SGDは、深層学習モデルの学習を高速化するための最も一般的な手法です（Dean et al. , 2012; Iandola et al. , 2015; Goyal et al. , 2017）。このようなモデルの勾配ベクトルは大きくなる可能性があるため、ワーカー間でそれらの勾配を共有するために必要な時間は、深層学習のトレーニングのスケーラビリティを制限します（Seide et al. , 2014; Iandola et al. , 2015; Lin et al. , 2018）。

これまでの研究では、この問題に対する解決策として、非可逆的な勾配圧縮が提案されています。顕著な例としては、グラデーションの座標をその符号のみで置き換えること（Seide et al. , 2014; Carlson et al. , 2015; Bernstein et al. , 2018, 2019; Karimireddy et al. , 2019）、個々の座標を量子化すること

(Alistarh et al. , 2017; Wen et al. , 2017)、勾配の低ランク近似(Wang et al. , 2018)などがあります。これらの作品は、いくつかの設定では全精度SGDよりも高速化を示していますが、コモディティハードウェアであっても、高速なネットワークと高度に最適化された通信バックエンドがあれば、その高速化は消えてしまうことがわかりました。また、先行研究の中には、SGDに比べてテストの精度が低下するものもあります。これらの問題を解決するために、次の3つの見解を組み合わせました。 i) 線形コンプレッサー演算子は、all-reduceを用いた集約を可能にすることでスケーラビリティを達成する。 ii) エラーフィードバックは、一般的な偏ったコンプレッサーで収束を保証する。

まず、SGDのための様々な勾配圧縮スキームの特性を調べ、どのようなものが高いスケーラビリティに不可欠であるかを明らかにする。特に、現在提案されているグラディエント・コンプレッサーは線形ではないことに注意したい。その圧縮されたメッセージは、生のグラデーションとは異なり、階層的に加算することができない。このため、現在の圧縮SGDアルゴリズムでは、効率的な*reduce操作*を用いてグラデーションを集約することができず、代わりに*gather*操作が必要となる。現在の深層学習フレームワークは、単独または主にall-reduceに依存しており、これは、通常のSGDが高速通信ハードウェアでうまくスケールする理由の鍵となっている（cf. Awan et al. , 2018; Panda et al. , 2019）。

次に、エラーフィードバック（計算された勾配と圧縮された勾配の差を保存し、次の反復でそれを再挿入する）を使用すると、収束と圧縮の両方が改善されることが最近示されました。

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

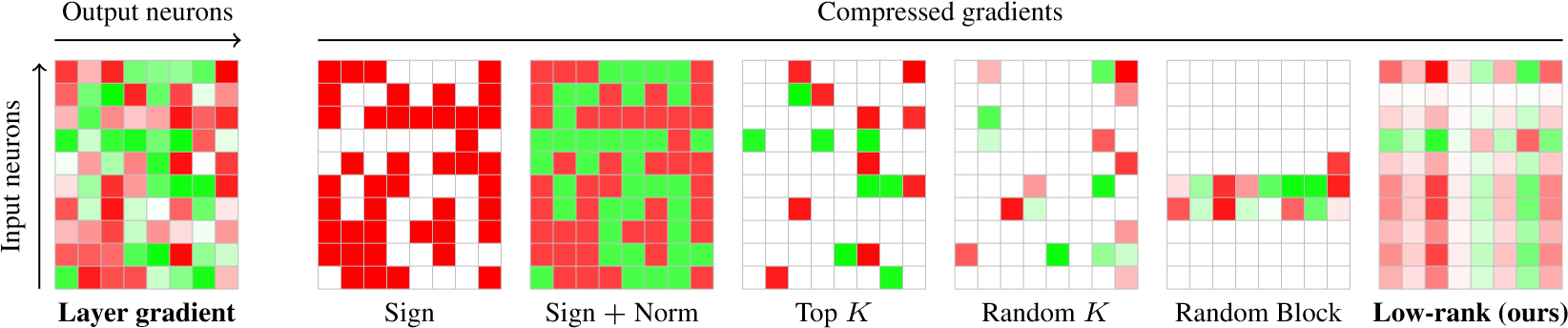


図1：本稿で比較した圧縮方式。左：レイヤーのグラデーションを行列として解釈したもの。右図様々な圧縮方式の出力結果。実装の詳細は付録Gにあります。

と圧縮スキームの一般化を図ることができる(Karimireddy et al. , 2019)。これにより、一般的な偏りのある勾配圧縮スキームが目標とするテスト精度に達することができる。

第三に、現代のオーバーパラメータ化された深層学習モデルの汎化能力は、低ランク化に関連しているという証拠が増えています（Arora et al. , 2018; Martin & Mahoney, 2018; Collins et al. , 2018）。私たちのように）低ランク更新を使用することは、暗黙的にスペクトル正則化を実行していると見なすことができ（Gunasekar et al. , 2018）、したがって、良好な一般化特性を持つことが期待できます（Yoshida & Miyato, 2017）。さらに、Wangら（2018）は、深層学習モデルの確率的勾配の固有スペクトルが減衰することを示し、ランクベースのスキームが収束性を犠牲にすることなく積極的な圧縮から逃れることができることを示唆しています。

本研究では，上記の見解を念頭に置いてPOWERSGDを設計します。POWERSGDは、一般化された*べき乗*反復（サブスペース反復として知られている(Stewart & Miller, 1975)）を用いて、勾配の低ランク近似を計算します。この近似式は計算量が少なく、高価な特異値分解を必要としません。効率的な近似の質を向上させるために，前回の最適化ステップで得られた近似を再利用して，パワー反復を*ウォームスタートさせます．*勾配を120倍以上に圧縮することで，CIFAR10のRESNET18では通信時間（符号化と復号化を含む）を54%削減し，WIKITEXT-2のLSTMでは90%削減することができました．また、フルテスト品質までのエンドツーエンドのウォールクロックトレーニング時間は、RESNET18では24％、LSTMでは55％短縮されました。

# 関連作品

グラデーション圧縮 様々な圧縮方式（図1）が提案されている。Alistarhら(2017)とWenら(2017)は各勾配座標を量子化し、Seideら(2014)とCarlsonら(2015)、Bernsteinら(2018, 2019)とKarimireddyら(2019)は勾配の各座標をその符号に置き換え、Linら(2018)とStichら(2018)とWangniら(2018)は最大の少数の座標を使用し、Konecnˇyら(`) (2016)とWangら(2018)は低ランク近似を使用しています。

Wangら（2018）によるSpectral Atomoは、おそらく我々の仕事に最も近い。これは、勾配の特異値ベクトルの重要度サンプリングを行い、不偏的な圧縮スキームです。しかし、それは反復ごとに完全な特異値分解を必要とし、それゆえに計算上非現実的です。

可換性のある圧縮と加算 Yu et al.(2018)は、勾配加算を伴う圧縮の可換性が、*リングオールリデュース*による効率的な集約を可能にすると強調しています。しかし、ほとんどの圧縮はこの特性を欠いています。Yuらは、グラジエント座標間の時間的に一貫した相関を利用して、線形に圧縮しています。POWERSGDは、我々が「線形性」と呼ぶ同様の特性を持っています。

誤差フィードバック 最初に(Seide et al. , 2014)で紹介され、(Stich et al. , 2018)で凸ケースについて分析されたエラーフィードバックは、ワーカーのグラデーションと圧縮されたグラデーションの差(すなわち*エラー)を*計算し、それを次のグラデーションに戻す(*フィードバック*)ことを含みます。Karimireddyら(2019)とStich & Karimireddy(2019)は、エラーフィードバックのフレームワークをさらに発展させ、レートを改善して一般化した。非凸の設定では、Karimireddyら（2019）は、偏った圧縮方式（例えばsignまたはtop*-K*）を使用する場合、収束と一般化の両方でエラーフィードバックが重要であることを示す。一般的に、エラーフィードバックを備えたバイアス圧縮方式は、バイアスなしの対応策よりも性能が高い傾向にある。Linら（2018）による実用的なアルゴリズムも、エラーフィードバックを備えた近似的なtop*-K*圧縮器としてのものです。

低ランク手法 最近の作品では、現代のオーバーパラメータ化されたディープネットワークでは、最終的に学習されたモデルが「低安定ランク」を持つことが論じられている（Martin & Mahoney, 2018; Li et al. , 2018）。これは、実質的にオーバーパラメータ化されているにもかかわらず、彼らの印象的な汎化特性を部分的に説明することができる（Arora et al. , 2018）。明示的なスペクトル正則化を加えることで、このようなモデルの性能がさらに向上することが示されている（Mazumder et al. , 2010; Yoshida & Miyato, 2017）。我々のように）低ランク更新を使用することは、同様の正則化を暗に実行しているとみなすことができます（Gunasekar et al,

2018).対象行列が（単なる低安定位ではなく）正確に低位であることがわかっている場合、Yurtseverら（2017）は、エラーフィードバックを必要とせずに、勾配の低位近似を用いてオプティマに収束させることが可能な場合があることを示している。

# 方法

機械学習モデルのデータ並列最適化では，多数の*W人の*作業者が同じモデルパラメータ**x**∈*Rd*を共有している．それぞれのワーカーは，独立した確率的勾配を計算し，その勾配を平均化して集約[[1]](#footnote-1)その集約結果に基づいてモデルパラメータを更新することで，**x**を反復的に更新する．

アルゴリズム1

ランク-。

*r*

P

OWER

SGD圧縮

1: 更新ベクトル∆*wは，*個々のモデルパラメータに対応するテンソルのリストとして扱われる。ベクトル状のパラメータ（バイアス）は圧縮されずに集約される。その他のパラメータは行列に整形されます。以下の関数は、これらの行列を個別に処理します。各行列*M*∈*Rn×mに対して、*対応する*Q*∈*Rm×r*がi.i.d.標準正規分布から初期化されます。

2：関数COMPRESS+AGGREGATE(更新行列*M*∈*Rn×m*, 前の*Q*∈*Rm×r*)

3: *P*←*MQ*

4: *p* ← すべて reduce mean(*p*) *.*ここで、*P* = *W*1 (*M*1 + ... + *MW*)*Q*

5: *p*ˆ←オルソゴナライズ(*p*) *.*直交する列

6: *Q*←*M*＞Pˆ

7: *q* ← all reduce mean(*q*) *.*これで 

8: 圧縮表現（*P,Q*ˆ ）を返す。

9: 関数の終了

10：関数DECOMPRESS(*P*ˆ∈*Rn×r*, *Q*∈*Rm×r*)

11: リターン*PQ*ˆ >

12: 関数の終了

POWERSGD 圧縮 モデルの各層を独立して近似します。完全連結層のパラメータ（密な行列乗算）とその勾配は、固有の行列構造を持っています。畳み込み層のパラメータは、完全連結層を2Dグリッドの入力に繰り返し適用したものと自然に解釈できます。実際には、これは4次元の勾配テンソルにおいて、入力とカーネルの次元を平坦にすることに相当します。ニューラルネットワークにはバイアスベクトルも含まれていますが、これらは通常、パラメータ空間のごく一部を構成しており、圧縮せずに集約することができます。

各パラメータの勾配*M*∈*Rn×m*に対して、*PQ*>がMをよく近似するような行列*P*∈*Rn×r*、*Q*∈*Rm×r*を見つけることがランク*R*行列近似の目的です。POWERSGDは、このような近似式を計算するために、*r > 1*に一般化された部分空間反復*-power* iterationを1ステップ使用します。これは、1回の右乗算、1回の左乗算、および直交化を行います。直交化にはグラムシュミット法を用いました。これは、行列の列数が非常に少ない（1～4列）ためであり、圧縮手順の中で最もコストのかかる部分です。さらに、前のステップで計算された近似値を再利用することで、部分空間の反復を「ウォームスタート」させます。ウォームスタートを導入することで、*1回の部分*空間反復処理で、高価な特異値分解による最良のランクR近似と同じ性能の因子分解*M* ∼ *PQ*>を得ることができます。

ワーカー間の効率的な集約 データ並列最適化では，ワーカーの勾配の*平均値*を近似したい．POWERSGDが、*W人の*ワーカーからの対応するグラデーション[*M*1 *...MW*]のリストを操作するとします。アルゴリズムの中で*Mが出現するのは、*いずれも（線形）行列の乗算の後に、ワーカーに対する（線形）平均値の減少を行うものです。これにより、実用的な不変性が導入されます。つまり、バッチサイズ*B×W*の1つのワーカーでの実行は、バッチサイズ*B*ずつの*Wの*ワーカーでの実行と同等です。この性質を「線形性」と呼びます。詳細は、付録A.3を参照してください。

POWERSGDが線形であることの重要な利点は，gather演算を必要とするのではなく，all-reduceプロトコルを用いて実装できることです．この違いを説明するために，*W個の*行列の和を計算したいとします．

(

a) 集合

(

b) 縮小

all-reduce法では、加算の連想性を利用して、計算を(*M*1+*M*2)+(M3+*M*4)と書き換えることができます。

これにより、分割統治のアプローチが可能となり、右図のように、合計タスクを複数のワーカーに分割することができます。*W個の*ワーカーを使った場合、計算時間と通信時間は、all-reduceではO(*logW*)となり、all-gatherではO(*W*)となります。

スケーリングの改善に加えて、全還元通信は*二重圧縮を避けることが*できるため、パラメータサーバーの設定よりも好ましい。パラメータサーバーでは、「クライアント→サーバー」と「サーバー→クライアント」の両方の通信を圧縮する必要があります（Caldas et al. , 2018; Bernstein et al. , 2019; Seide et al. , 2014）。私たちは、圧縮と集約を1つのステップに統合することでこれを回避します。

誤差フィードバックSGD POWERSGDスキームは偏っている（つまり、ランダムな勾配を圧縮・解凍しても期待通りのオリジナルが得られない）ため、エラーフィードバックを使用します（Seide et al. , 2014; Karimireddy et al. , 2019）。私たちのエラーフィードバックのバージョン（アルゴリズム2）は、圧縮後の*モメンタム*を導入することでオリジナルを拡張しています。この単純な拡張により、モーメンタム付きのSGD用に調整されたものと同じ学習率とハイパーパラメータを再利用することができます。

Algorithm 2 Distributed Error-feedback SGD with Momentum

1：ハイパーパラメータ：学習率*γ*、モメンタムパラメータ*λ*

2：モデルのパラメータ**x**∈*Rd*、モーメンタム**m**←**0**∈*Rd*を初期化、作業者間で複製

3: 各ワーカー*w* = 1*,...,W*において、次のことを行います。

4: メモリの初期化 *ew* ←**0**∈*Rd*

5: for each iterate *t* = 0*,...* do 6: 確率的勾配*gw*∈*Rd*を計算する。

7: ∆*w* ←*gw* + *ew .*誤差のフィードバックを更新に組み込む

8: C(⊿*w*)←COMPRESS(⊿*w*)

9: *ew* ← ∆*w* - DECOMPRESS(C(∆*w*)) *.*ローカルエラーを記憶する

10: c(⊿)←集約(c(⊿1*),...,*c(⊿*w*)) *.*グラデーションの交換

11: ∆0 ← decompress(c(⊿)) *.*更新プログラム∈*Rd)を*再構築

12: **m** ← *λ***m** + Δ0

13: **x** ←**x**-*γ*（△0＋**m**

14: エンドフォー

15: End at

# POWERSGDの分析

このセクションでは，POWERSGDのさまざまな側面を個別に検討し，i) エラーフィードバックを使用する効果，ii) 「ウォームスタート」の効果，iii) 近似順位を変化させた場合のテスト精度と圧縮率のトレードオフ，を経験的に理解したいと考えています。

4.1 エラーフィードバックの効果

POWERSGDの基本アルゴリズムとしてエラーフィードバックSGDを使用することには、2つの利点があります。第一に、それは我々が偏った圧縮機を使用することを可能にする。第2に、EF-SGDは収束性を改善し、より良いテスト精度を得ることができます（Karimireddy et al.、2019）。

テスト精度の向上を説明するために，誤差をフィードバックする偏った圧縮器であるPOWERSGDと，不偏の低ランク近似とを比較します。行列*M*∈*Rn×mを*近似するために、不偏のランク*R近似*器は、E[*UU*>]=Imとなるようなランダムな行列*U*∈*Rm×r*をサンプリングし*、*低ランク近似値として(MU,*U)*を出力します。この方式は、以下のように不偏的である。

e[(*mu*)*u*> ] = m e*[* uu> ] = mi = m *.*

POWERSGDは，この偏りのないスキームの自然な偏りのある対応策です．表1は，誤差フィードバックを備えた我々の偏りのある近似器が，画像分類において偏りのない演算子よりも優れていることを示している．

|  |  |
| --- | --- |
| 表1:エラーフィードバックのある場合とない場合のランクベース圧縮。バイアスのかかったPOWERSGDは，テストの精度において，バイアスのかかっていない線形ランク*R*圧縮器を上回る． | 表2：ベストランク2近似とPOWERSGDの比較。ウォームスタートにより、テストの精度が向上し、POWERSGDと比較しても遜色ありません。 |

は，最良のランク 2 近似の性能を有しています．

アルゴリズム テスト精度 データ/エポック

SGD 94*.*3% 1023 MB

アルゴリズム テストの精度

94

*.*

4

%

94

*.*

0

%

順位-1 POWERSGD 93*.*6% 4 MB

順位-2 POWERSGD 94*.*4% 8 MBBest approximation 94*.*4%

偏りのないランク1 71*.*2% 3 MB ウォームスタート（デフォルト

アンバイアス・ランク2 75*.*9% 4 MB ウォームスタートなし

画像分類 - RESNET18 on CIFAR10

表 3：POWERSGD とアルゴリズムの比較 テストの精度 エポック毎のデータ送信 バッチ毎の時間 ランクを変化させた場合スフの場合

SGD 94*.*3% 1023 MB (1×) 312 ms +0% 適正なランク、POWERSGD ×) 229 ms -26% ランク 1 93*.*6% 4 MB (243

ランク 2 94*.*4% 8 MB (136×) 239 ms -23% RESNET18 および LSTM ランク 4 94*.*5% 14 MB (72×) 260 ms -16% の学習を、通信量の削減により高速化します。

同等のテスト品質を実現 言語モデリング - LSTM on WIKITEXT-2 with regular SGD in Algorithm Test perplexity Data sent per epoch Time per batch same number oferations.

バッチごとの時間には SGD 917730 MB (1×) 300ミリ秒 0% フォワード/バックワードパス ランク 110225 MB (310×) 131 ms -56% 順位 29338 MB (203×) 141 ミリ秒 -53

(定数）を設定します。)については、セクション5を参照してください。

順位 49164 MB (120×) 134 ミリ秒 -55

実験のセットアップを行います。

4.2 ウォームスタートの効果

POWERSGDは勾配行列の最適なrank*-r*近似を計算するのではなく、power iterationに基づいた安価で忠実度の低い近似を使用します。表6でPOWERSGDとSpectral Atomoのバッチごとの時間を比較すると、特異値分解を避けることの重要性がわかります。POWERSGDのような形状のグラデーションでは、確率的グラデーションのSVDを計算するのに673msかかり、これはミニバッチグラデーションを6回計算するのに相当します。これに対し，ランク2のPOWERSGDでは，16人のワーカー間の通信を含めても，1回のフルステップで105msしかかかりません．

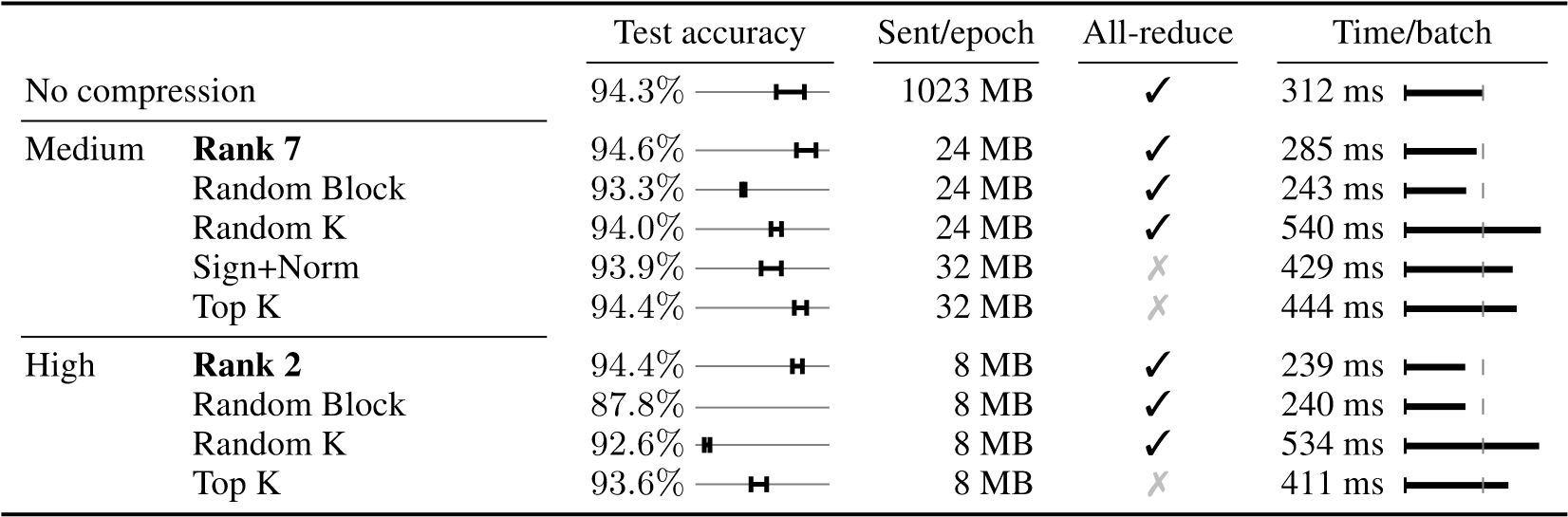
表 2 の「ウォームスタートなし」と「最良の近似値」のテスト精度を比較してみてください。POWERSGD の主な特徴は，累乗反復アルゴリズムを初期化するために以前に計算された行列近似を再利用する*ウォームスタート*戦略です．累乗反復を行う行列が一定であれば、最適なrank*-r*近似値を復元します（付録の定理Iを参照）。我々は、基礎となる行列が変化する場合でも、この戦略が意味を持つことがあると主張する。

タイムステップ*t*で、前の因数分解を活用して、*Pt*-1と*Qt*-1を出発点として再利用することで利益を得ることができるとします。これが真実である可能性は低いですが、*Mt*と*Mt*-1が完全な勾配の確率的な近似である場合、関数が滑らかであり、小さな更新ステップを取るだけなので、E[*Mt*]≒E[*Mt*-1]であることが期待できます。この結果は、Oja's algorithm for *stochastic power iteration* (Oja, 1982)に似ており、したがって、近似品質が向上する可能性があります。表2で経験的に示すように、この「ウォームスタート」戦略は、POWERSGDと、よりコストのかかるベストランク*R*近似とのテスト精度の差を縮めるのに十分です。

4.3 ランクを変えた場合の効果

POWERSGDでは、ユーザーがその勾配近似のランクを選択することができます。近似品質と圧縮・解凍・転送コストとのトレードオフを表3に示します。画像分類と言語モデリングの両方のタスクにおいて、POWERSGDが達成するテスト品質は、ランクの増加とともに向上しています。どちらの場合も、通常のSGDと同程度、あるいはわずかに上回る品質に達しています。

表 4: 統合環境における Error-feedback SGD のさまざまな圧縮演算子の比較。CIFAR10 において、16 個の GPU を使用して Momentum (Algorithm 2) を用いた Error-feedback SGD の 300 エポックを実行し、学習率を全精度 SGD 用に調整した。なお、ランク2と7のPOWERSGDのバリエーションは、達成されたテストの精度とバッチあたりの時間（前進、後退、圧縮、解凍、およびグラディエントアグリゲーションの合計時間）の間で最適なバランスをとっています。



# 結果

実験のデフォルト設定

|  |  |  |
| --- | --- | --- |
|  | データセット | CIFAR10 |
|  | 建築 | RESNET18 |
|  | 労働者数 | 16 |
|  | バックエンド | NCCL (PYTORCHで最速) |
|  | バッチサイズ | 128×労働者数 |
|  | モメンタム | 0.9 |
|  | 学習率 | 16人のために調整された-0*.*1 x  SGDでは16。ワーカーの数でリニアにスケールアップ |
|  | LRディケイ | エポック150および250での*/*10 |
|  | LRウォームアップ | シングルワーカーのLRからスタートして、5エポック内で線形的に |
|  | # エポック | 300 |
|  | 重量減衰 | 10−4,  BatchNormパラメータは0 |
|  | 繰り返し | 3、種を変えて |
|  | エラーバー | 最小-最大 |

このセクションでは、ディープニューラルネットワークの分散最適化におけるPOWERSGDの実用性を示します。POWERSGDの圧縮スキームは、i)高速で、SGDのテスト性能に匹敵すること、ii)最適でない通信バックエンドを使用しても、ワーカーの増加に伴ってうまくスケールすること、iii)大規模なモデルの学習時間を大幅に短縮することを示します。

ほとんどの解析はCIFAR10上で，右表の設定で行われています．WIKITEXT-2の言語モデリングのためのLSTMの追加評価により，POWERSGDの汎用性を検証しました．8台のマシンに16個のGPUを搭載し，高速ネットワーク（10Gbit/s）で接続しています．意味のあるタイミングを得るために，比較したすべてのオプティマイザーを同程度に最適化することを目指しました。パフォーマンスの最適化については，付録Hにリストを掲載しています．また、POWERSGDや、モーメンタムによるエラーフィードバックを使用する他の圧縮アルゴリズムにも*同じ*パラメータを使用しています。比較対象であるSpectral Atomo (Wang et al. , 2018) およびSignum (Bernstein et al. , 2019) の学習率は、別個にチューニングしました cf. Appendix I.

5.1 他のコンプレッサーとの比較

圧縮最適化におけるエラーフィードバックは，偏りのあるものを含む多数の圧縮スキームの使用を可能にする．図1に示した潜在的な圧縮演算子を表4で比較しています。達成されたテストの精度と，1つのミニバッチを処理するのにかかった時間の合計に基づいて，圧縮機を評価します．前者は、圧縮演算子の精度を総合的に評価したものであり、後者は、フォワードパス、バックワードパス、グラジエント圧縮・解凍、グラジエント通信に要する正味の時間である。本研究では、中程度と高程度の2つの圧縮レジームについて検討した。

符号ベースの手法で実現される32倍程度の圧縮では、Random Blockを除くすべての圧縮方式が、フル精度SGDに近いテスト精度を達成しています。これは、この領域のすべての圧縮方式（Random Block を除く）が十分な圧縮品質を得ていることを意味します。高圧縮（128×）では、POWERSGD が目標とするテスト精度を達成した唯一の方式として特に際立っています。

圧縮率が中程度の場合も高程度の場合も、高精度 SGD よりも高速なのは POWERSGD と Random Block だけです。どちらも単純な線形スキームであるため、all-reduce をサポートしていることに注意してください。Random *K* も all-reduce をサポートしていますが、圧縮と解凍の両方の段階でランダムメモリアクセスのオーバーヘッドがかなり大きいため、全体的に遅くなっています。 表 5: RESNET18 トレーニングの 1 回の反復に費やした時間の内訳 (秒単位)。POWERSGD（ランク2）はall-reduceを採用しているため、グラデーションのエンコード/デコードにかかる時間は一定です。

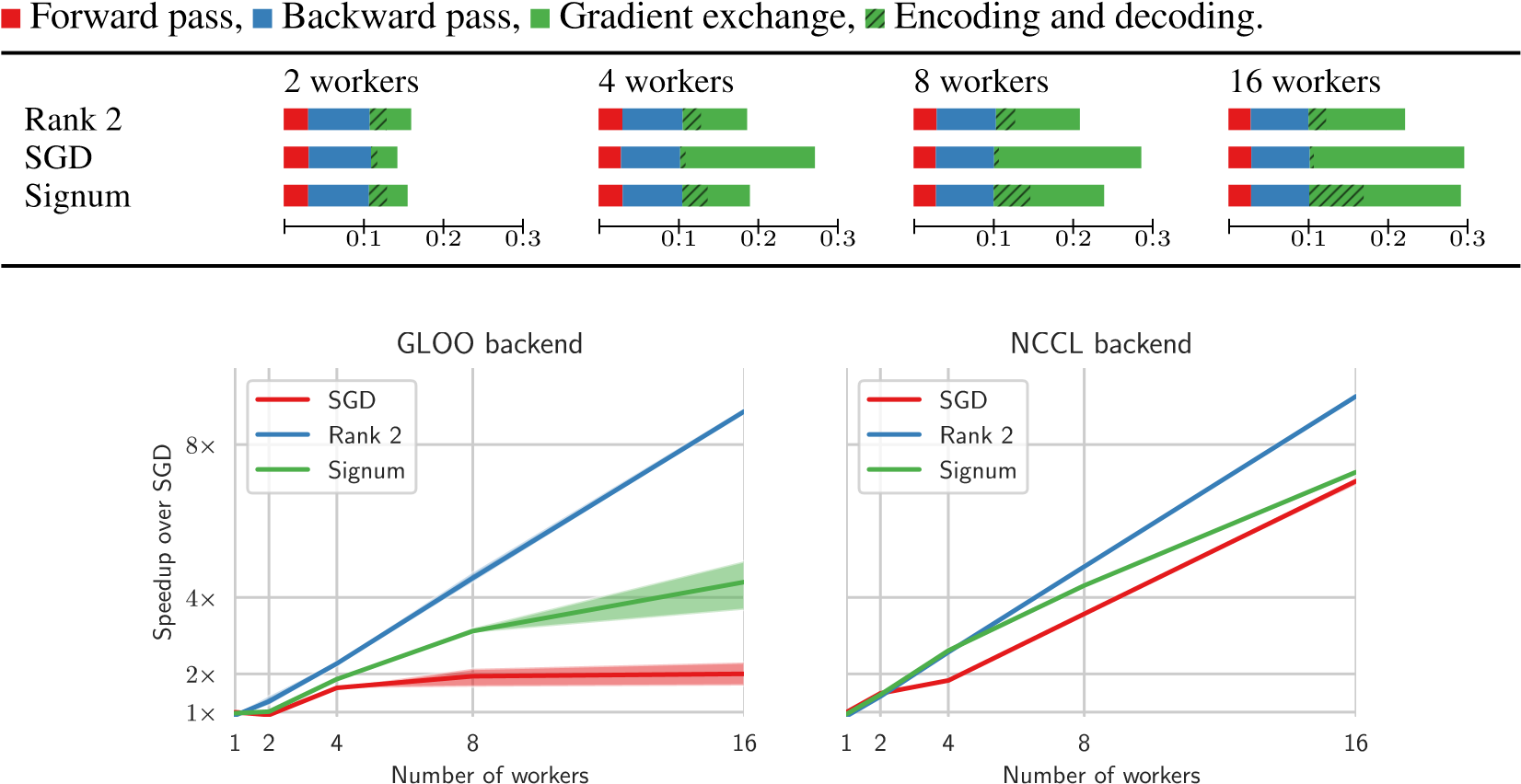


図3：CIFAR10におけるPOWERSGDのスケーリングを、2つの通信バックエンド上の全精度SGDおよびSignum（Bernstein et al. , 2019）と比較したもの。バッチサイズはワーカー数に応じて線形に増加します。1エポックの学習時間を1ワーカーのSGDと比較しています。なお、全体を通して使用されている高速なNCCLバックエンドは、我々の手法よりもベースラインに恩恵を与えている。

はSGDよりも優れています。このように、最新のGPU対応インフラでは、行列の乗算に依存するPOWERSGDが、他の圧縮方式よりも高速かつはるかに正確に動作します。

5.2POWERSGD のスケーラビリティ

ここでは、POWERSGDがワーカー数の増加に伴ってどのようにスケールするかを調査し、ワーカー数が大幅に増加した場合に何が期待できるかを明らかにします。さらに、これらの結果が通信バックエンドの選択にどのように依存するかを調べます。POWERSGDをBernsteinら（2019）のSGDとSignum（signSGD with majority vote）に対してベンチマークを行いましたが、これは分散アルゴリズムの現在の最先端であると考えています。

表5は、各ミニバッチ（つまり1ステップ）にかかった時間を、フォワードパス、バックワードパス、グラデーション交換（通信）、圧縮/解凍に分けて詳細に示したものです。フォワードパスとバックワードパスに費やされる時間は、すべてのアルゴリズムとワーカーの数で一定です。SGDとPOWERSGDはいずれもall-reduceを使用しているため、グラデーションの通信時間（表5の緑の実線）は、ワーカー数の増加に応じて緩やかに変化します。オールリデュースの代わりにオールギャザーを使用するSignumでは、急激な増加が見られます。4人の場合はPOWERSGDと同程度の時間ですが、16人になるとコストが高くなります。

all-reduceとall-gatherを比較した場合、デコード時間にはもうひとつ微妙な影響があります。all-reduceでは、*集約の*ステップと*通信の*ステップが同時に行われます。各ワーカーはあらかじめ集約されたグラデーションを受け取るため、解凍のコストはワーカーの数に依存しません。一方、all-gatherでは、ワーカーは*W個の*圧縮されたグラデーションを受け取り、個別に解凍して集約する必要があります（多数決または平均値を使用）。そのため、all-gatherでの伸長にかかる時間は、ワーカーの数に比例します。このことは、表5のハッシュ化された領域を比較するとわかります。この観察結果は、スケーラビリティのためのreduce操作の重要性を物語っています。

次に，最適化されたNCCLと低速のGLOOという2種類のバックエンドを調べました．最適化されたNCCLバックエンドでは，3つの手法とも適度にスケーリングされますが，Signumは対数プロットで傾きが1未満となり，サブリニアなスケーリングを示しています．遅いGLOOバックエンドでは，POWERSGDがその高い圧縮率のために優れたスケーリングを維持している唯一の手法であることが特徴的です．

表6：CIFAR10 での結果。

コントラスト トランク-2 スペクトル アルゴリズム テスト精度 データ/エポック1 バッチあたりの時間

アトモ（Wang et al , 2018）と

Signum（Bernstein et al. , 2019）です。 SGD 94*.*3% 1023 MB 312 ms +0%

アトモ 92*.*6% 113 MB 948 ms +204

POWERSGDは同じことを実現しています。

シグナム 93*.*6% 32 MB 301 ms -3%

テストの精度をフルに発揮 ランク2 94*.*4% 8 MB 239 ms -23%

デフォルトのエポックバジェット内のSGD。

表7：言語モデリングにおいて アルゴリズム テスト perplexityData/ epochTime per batch

ランク 4 POWERSGD は、SGD 91 7730 MB 300 ms +0% の目標テスト精度を達成し、Signum 142 242 MB 424 ms +41% と、SGD よりも大幅に高速化しました。ランク 4 91 64 MB 134 ms -55% 達成

5.3 その他のタスクと方法

表6では、POWERSGDを最先端の圧縮最適化アルゴリズムと比較しています。

SignumとSpectral Atomoです。各ステップで完全なSVDを実行するためのコストは、Spectral

Atomoは高性能な環境では実用的ではなく、特に他の手法のテスト精度に及ばないことを考慮すると、そのような結果となりました。Signumの性能ははるかに高く、SGDに比べてわずかにスピードアップしています。POWERSGD は、比較した手法の中で最も高速かつ高精度です。

POWERSGDの優位性は、非常に大きなモデルを使用する場合、つまり、通信が実際にボトルネックになる場合に顕著に現れます。これを検証するために、Signum、全精度SGD、POWERSGDを実行して、RESNET18よりもモデルサイズが大幅に大きい言語モデリングタスクでLSTMを学習させました（付録Fを参照）。全精度 SGD のテストスコアと一致させるには、ランク 4 近似を使用する必要がありました（セクション 4.3 参照）。POWERSGDは通信量を90%削減し、全体の実行時間を55%削減しましたが、Signumは全精度SGDよりも遅くなり、テストスコアも悪化しました。

表3、6、7に対応するテスト精度の収束曲線を付録Cに示します。これらの図では、どのような目標精度に対しても、時間対精度の改善を読み取ることができます。また、新しいタスク（WIKITEXT-2のトランスフォーマーを用いた言語モデリング）とパブリッククラウド上のより多くのワーカー（32人）にPowerSGDを使用したケーススタディを付録Dに掲載しています。

# 結論

勾配圧縮は、同期分散最適化における通信のボトルネックに対処するための有望なアプローチである。しかし、既存の圧縮手法は、最適化された全還元勾配集約を行うSGDよりも実行速度が遅いか、より重要なことに同じテスト性能に達しないため、これまで広く採用されていませんでした。私たちは、POWERSGDを初の実用的な勾配圧縮法と考えており、実際に適用する準備が整っていると考えています。

POWERSGDの実用性の鍵となるのは、安価に計算できる線形圧縮スキームであり、全還元の勾配集約を可能にすると同時に、全精度SGDのテスト性能に匹敵する性能を実現しています。このSGDと比較したスピードアップは、NLPでよく見られるような大規模なモデルでは実際に*増加します。*さらに、エラーフィードバックアルゴリズムを変更した結果、POWERSGDはSGDをプラグインで置き換えることができ、追加のハイパーパラメータチューニングの必要性がなくなりました。POWERSGDのこれらの特性により、フル精度のSGDで可能なものよりもさらに大きなモデルを、さらに多くのワーカーを使ってトレーニングすることができると期待しています。

POWERSGDでは、バッチサイズを大きくすることで高速な学習が可能になりますが、バッチサイズを大きくすると、最終的に「汎化ギャップ」に悩まされることが知られています（Shallue et al.、2018）。これは直交する問題であり、我々は大規模な学習を解決するための次のステップとして捉えています。私たちの実験では、POWERSGDがSGDよりも高いテスト精度を達成できることを確認しています。低ランク化と汎化の間の興味深い関連性と合わせて、このことは、POWERSGDが大規模なバッチトレーニングにおける汎化のギャップを埋めるのにも役立つ可能性があることを示しています。

謝辞

また，貴重なご意見をいただいたAlp Yurtsever氏，Tao Lin氏，およびご意見をいただいた査読者の方々に感謝いたします。本プロジェクトは、SNSF grant 200021\_175796、およびGoogle Focused Research Awardの支援を受けています。

# リファレンス

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. QSGD: Communication-efficient sgd via gradient quantization and encoding.In *Advances in Neural Information Processing Systems (NIPS*), 2017.

Arbenz, P. 大規模な固有値問題を解くためのレクチャーノート。*D-MATH, ETH Zürich*, 2, 2016.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach.In *International Conference on Machine Learning (ICML*), 2018.

Awan, A. A., Chu, C.-H., Subramoni, H., and Panda, D. K.Optimized broadcast for deep learning workloads on dense-GPU infiniband clusters:MPIかNCCLか？In *European MPI Users' Group Meeting (EuroMPI*), 2018.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling.In *International Conference on Learning Representations (ICLR*), 2019.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signSGD: compressed optimisation for non-convex problems.In *International Conference on Machine Learning (ICML*), 2018.

Bernstein, J., Zhao, J., Azizzadenesheli, K., and Anandkumar, A. signSGD with majority vote is communication efficient and fault tolerant.In *International Conference on Learning Representations (ICLR*), 2019.

Caldas, S., Konecný, J., McMahan, H. B., and Talwalkar, A. Expanding reach of federated learning by reducing client resource requirements. *arXiv*, abs/1812.07210, 2018.

Carlson, D., Cevher, V., and Carin, L. Stochastic Spectral Descent for Restricted Boltzmann Machines.In *International Conference on Artificial Intelligence and Statistics (AISTATS*), 2015.

Collins, E., Bigdeli, S. A., and Süsstrunk, S. Detecting memorization in ReLU networks. *arXiv*, abs/1810.03472, 2018.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. 大規模分散ディープネットワーク.In *Advances in Neural Information Processing Systems (NIPS*), 2012.

Ghadimi, S. and Lan, G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming.*Mathematical Programming*, 156(1-2):59-99, 2016.

Goyal, P., Dollar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: training imagenet in 1 hour. *arXiv*, abs/1706.02677, 2017.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry.In *International Conference on Machine Learning (ICML*), 2018.

Iandola, F. N., Ashraf, K., Moskewicz, M. W., and Keutzer, K. FireCaffe: near-linear acceleration of deep neural network training on compute clusters. corr abs/1511.00175 (2015), 2015.

Karimireddy, S. P., Rebjock, Q., Stich, S. U., and Jaggi, M. Error feedback fixes SignSGD and other gradient compression schemes.In *International Conference on Machine Learning (ICML*), 2019.

Konecnˇy, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated` Learning:Strategies for improvement communication efficiency. *arXiv*, abs/1610.05492, 2016.

Li, Y., Ma, T., and Zhang, H. Over-parameterized matrix sensing and neural networks with quadratic activationsにおけるAlgorithmic regularization（アルゴリズムによる正則化）。*学習理論に関する会議（COLT）*、2018年。

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression:分散学習のための通信帯域の削減。In *International Conference on Learning Representations (ICLR*), 2018.

Martin, C. H. and Mahoney, M. W. Implicit selfregularization in deep neural networks:Evidence from random matrix theory and implications for learning. *arXiv*, abs/1810.01075, 2018.

Mazumder, R., Hastie, T., Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices.*機械学習研究会誌*, 11(Aug):2287-2322, 2010.

Oja, E. 主成分分析器としての簡易ニューロンモデル。*数理生物学雑誌*, 15(3):267-273, 1982.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling.In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Panda, D. K. D., Subramoni, H., and Awan, A. A. High performance distributed deep learning:初心者向けのガイドです。In *Symposium on Principles and Practice of Parallel Programming (PPoPP*), 2019.

Robbins, H. and Monro, S. A Stochastic Approximation Method.*The Annals of Mathematical Statistics*, 22(3):400-407, September 1951.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns.In *Annual Conference of the International Speech Communication Association (INTERSPEECH*), 2014.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training. *arXiv*, abs/1811.03600, 2018.

Stewart, G. Simultaneous Iteration for computing invariant subspaces of non-Hermitian matrices.*Numerische Mathematik*, 25(2):123-136, 1976.

スチュワート、G.とミラー、J.は、行列の固有ベクトルを計算するための同時反復の方法。*数値解析のトピックスII*，169-185頁，1975年。

Stich, S. U. and Karimireddy, S. P. The error-feedback framework:Better rates for sgd with delayed gradients and compressed communication. *arXiv*, abs/1909.05350, 2019.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified SGD with memory.In *Advances in Neural Information Processing Systems (NeurIPS*), 2018.

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. ATOMO:

原子的スパース化による通信効率の良い学習。In *Advances in Neural Information Processing Systems (NeurIPS*), 2018.

Wangni, J., Wang, J., Liu, J., and Zhang, T. Gradient sparsification for Communication-efficient distributed optimization.In *Advances in Neural Information Processing Systems (NeurIPS*), 2018.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad:Ternary gradients to reduce communication in distributed deep learning.In *Advances in Neural Information Processing Systems (NIPS*), pp.1509-1519, 2017.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving generalizability of deep learning. *arXiv*, abs/1705.10941, 2017.

Yu, M., Lin, Z., Narra, K., Li, S., Li, Y., Kim, N. S., Schwing, A. G., Annavaram, M., and Avestimehr, S. Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training.In *Advances in Neural Information Processing Systems (NeurIPS*), 2018.

Yurtsever, A., Udell, M., Tropp, J. A., and Cevher, V. Sketchy decisions:最適なストレージを用いた凸型低ランク行列最適化。In *International Conference on Artificial Intelligence and Statistics (AISTATS*), 2017.

Zhao, J. signSGD with majority vote. github.com/PermiJW/signSGD-with-Majority-Vote, 2019.[Online; accessed 12-May-2019].

1. Bernsteinら（2019）は、平均化ではなく多数決で1ビットのグラデーションを集約するSignumを提案している。 [↑](#footnote-ref-1)