大規模並列環境における少精度型を用いた
ディープラーニングの学習精度の検証

大山洋介
東京工業大学 情報理工学院

1. はじめに

深層学習(ディープラーニング,以下 DL)とは生物の神経回路を模した計算モデルである Deep Neural Network(以下 DNN)を用いた機械学習の一手法である。DL は ILSVRC(画像認識に関する競技会)で CNN(Convolutional Neural Network)が他の古典的な機械学習アルゴリズムと比較して高い推論制度を達成した[1]ことを一つのきっかけとして機械学習分野における一種のブレイクスルーとして近年非常に注目されている。

DNN や CNN の計算は行列同士の積(GEneral Matrix Multiply, GEMM)や畳み込み演算が計算時間において支配的であると言われていることから，学習や推論に単体の GPU や GPU クラスタが用いられることが多い。複数の GPU を用いて学習を行う場合，データ並列学習と呼ばれる一回の学習ステップで使用されるデータサンプルの集合(ミニバッチ)を GPU 数に分割して計算を行う手法が広く用いられている。この際，高いスケーラビリティを維持するためには GPU ごとのサンプル数を一定に保って並列数を増加させる(弱スケールする)必要があるが，サンプル数が過度に増加することによってモデルパラメータの更新量のランダム性が低下し，推論精度が悪化することが指摘されている。

本研究では，DL が自然科学的なデータセットと冗長なモデルを用いることから計算要素の数値誤差に対してロバストであるという仮定を用いた上で，通信データを低い精度で表現することで推論精度を損なわずに並列化を行う手法を提案した。特に，特定の通信精度に対する推論精度の悪化が未知である場合に，精度を適合的に変更することで常に最適な通信手法を選択する手法の予備評価を行った。

本報告では，東京大学情報基盤センター「若手・女性利用者推薦」平成 29 年度前期研究課題「大規模並列環境における少精度型を用いたディープラーニングの学習精度の検証」によって得られた研究成果を報告する。

2. データ並列学習

DL におけるデータ並列学習とは，主に確率的勾配降下法(Stochastic Gradient Descent，以下 SGD)を用いてモデルパラメータの学習を行う際の並列化手法のひとつである。

SGD では時刻tにおけるパラメータ$W^{(t)}$をデータセットからランダムに選択された部分集合D(ミニバッチ)によって次のように反復的に更新する:

$$W^{(t+1)} \leftarrow W^{(t)} - \frac{\eta^{(t)}}{|D|}\sum_{x\in D}\frac{\partial L}{\partial W}(x; W^{(t)})$$

ここで$\partial L/\partial W$はコスト関数のパラメータについての勾配，$\eta^{(t)} > 0$は学習係数と呼ばれる更新度合いを決定するハイパーパラメータである。

データ並列学習では SGD の各データサンプル$x \in D$についての計算に依存性がないことを利用し，複数の GPU で勾配を並列に計算し，その後勾配の総和を GPU 間通信により計算することで並列化を行う。この際，各 GPU が計算するミニバッチDの部分集合D'についての勾配$\sum_{x\in D'}\frac{\partial L}{\partial W}(x; W^{(t)})$は GPU 内で計算できることから，並列数にかかわらず各 GPU が送信・受信するデータ量は一定である(パラメータ数のみに比例する)。この通信には，プロセス間のデータをリダクションし結果を全プロセスが取得する all-reduce が使用される(図 1)。

データ並列学習では 1 回のステップ(図 1 の 1.～3.)で GPU が要する計算時間は$O($ミニバッチサイズ$/$GPU 数$)$であるのに対し，通信時間はデータサイズが一定であるという性質から一般に$O(\log($GPU 数$))$となる。よって，ミニバッチサイズを一定に保ったまま(計算内容を保ったまま)並列数を増加させた場合には通信時間がボトルネックとなる傾向にある[4]。

一方で，並列数の増加にともなってミニバッチサイズを増加させる(弱スケールによる並列化を行う)場合は通信性能による性能低下が起こりづらいが，一般に SGD においてミニバッチサイズを過度に増加させると推論精度が悪化するため，サンプルあたりの計算速度の向上が学習自体の速度向上に必ずしも寄与するわけではない。近年ではモデルパラメータの最適化手法の改良などの理論的な方面からのアプローチにより単一モデルの学習を推論精度の悪化なく 1000 GPU 程度までスケールする成果が報告されているが[2]，これらの手法が多様な DNN のすべてについて適用可能であるかどうかは依然明らかではない。

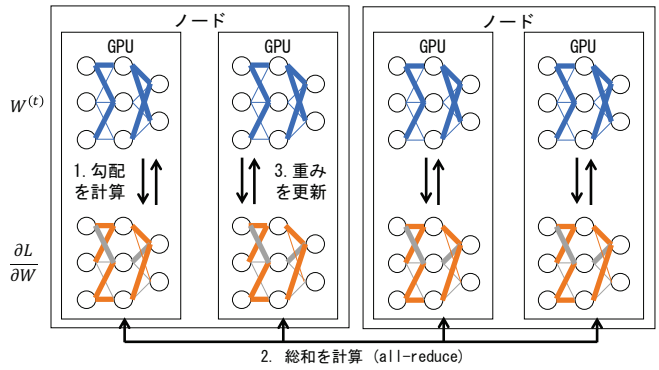

図 1: データ並列学習。

1. コスト関数の勾配計算，2. GPU 間通信による勾配の総和の計算，3. 重み(パラメータ)更新を繰り返すことによって学習が進行する。

3. 低精度型を用いた通信手法に関する先行研究

DL は自然科学的なデータセットを用いて学習を行うことが多く，また学習手法としてもランダム性をともなう SGD が用いられることから，厳密な計算精度を要求するような HPC アプリケーションと比較して個々の計算要素の数値誤差に対して比較的ロバストであると言われている。これを反映した例として，NVIDIA 社の GPU では Pascal 世代より半精度浮動小数点数演算がハードウェアレベルでサポートされ，さらに Volta 世代では Tensor Core と呼ばれる4×4行列(半精度)同士の行列積を計算するユニットが導入されるなど，DL 向けの低精度な演算がサポートされつつある。

このような低精度な演算・データ型を通信部分に導入することにより通信量・通信時間を削減する手法についても研究が行われている。例えば，Seide ら[5]はコスト関数の勾配を符号により 1 bit に符号化する手法(1-bit SGD)を提案しており，この手法は Microsoft が開発している DL フレームワークである CNTK に実装されている。しかし 1-bit SGD では all-reduce 通信の際に 2 回の符号化・復号化を行う必要があり，また通信中に発生する符号化誤差を次回の通信に持ち越す必要があるなど，通信中に(単純な要素ごとの加算と比較して)複雑な処理が必要となる問題がある。

一方で我々は 8 bit の浮動小数点型(以下 fp8)を用いて MPI のユーザ定義型・演算によって低精度な all-reduce 通信を行う手法を提案した[4]。fp8 は半精度浮動小数点型の上位 8 bit を使用した型であることから，半精度型の演算・変換に対応した CPU・GPU で容易に加算や変換を行うことができる(図 2)。fp8 を用いた加算アルゴリズムは TSUBAME-KFC/DL の 16 ノードで 256×10^6要素の MPI_Allreduce を実行した場合について単精度型を用いた場合と比べて 3.2 倍の高速化を達成したほか，CaffeNet と GoogLeNet という二つの CNN の学習においてミニバッチサイズを固定したまま推論精度を損なわずにそれぞれ 2.7 倍,2.2 倍の高速化を達成した。

一方で，fp8 を用いた場合は MPI プロセス数やオーバーフローを防ぐためのスケーリングを行う回数によって学習後の推論精度が大きく異なる場合があることが明らかになった。しかし，一般にデータ型の精度が推論精度にどの程度影響するかどうかは明らかではなく，ミニバッチサイズの問題と同様に計算速度と学習速度のトレードオフを調整する必要性が依然存在する。

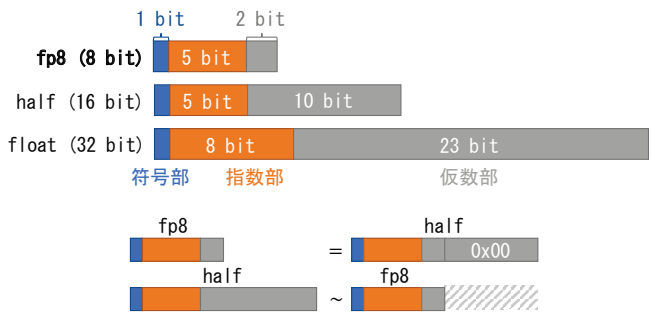

図 2: 浮動小数点数型の構成(上)および fp8 と半精度浮動小数点数型(half)の変換[4]。
fp8 と half は下位 8 bit の追加または無視により容易に変換することができる。

4. 提案手法

本研究では通信に使用する型(以下，通信手法)を適合的に選択することにより，低精度な通信手法による推論精度の悪化を防止しつつ通信量を削減する手法を提案する。提案手法では，以下の手順によって学習時に周期的に最適な通信手法を選択する(図 3)。

1. 学習のある時点でモデルパラメータや momentum 等の学習の状態に関するデータをコピーし，その時点から異なる通信手法を用いて並行に一定ステップの学習を行う。
2. すべての通信手法について，さらに一定ステップの学習を継続した際に得られる推論精度を学習曲線(推論精度やコスト関数をステップ数や時間を軸としてプロットした曲線)の性能モデル[6]により予測する。
3. 単位時間あたりの最も推論精度の向上が大きい通信手法を選択し，1. で学習した地点からさらに一定ステップの学習を行う。
4. ユーザが指定したステップ数に達するまで 1. 以降を繰り返す。

今回は提案手法で得られる推論制度に関する予備評価という目的で，1. の複数の通信手法による学習を逐次的に行い最良の手法を選択する手法を DL フレームワークである Caffe[7]に実装した。学習状態の保存・復元については GPU メモリ上にデータをコピー・復元するよう実装した。

通信手法としては，単精度(float)と fp8 のほかに，より精度が低く通信量を削減できる手法として 1 つの数値を N bit($N = 1,2,4,8$)で符号化する手法を実装した(以下 table(N))。この手法では fp8 でスケーリング係数を決定する手法と同様にテンソルごとに勾配の絶対値をランダムサンプリングし，その内の$99\times\frac{i}{2^{N-1}}$ パーセンタイル($i = 1,2,\cdots,2^{N-1}$)の値を絶対値とした代表値とした符号化テーブルを作成する。通信の際には 1-bit SGD と同様にテーブルを用いて符号化・復号化を行う。

学習曲線のモデルについては Domhan ら[6]により提案されている初等関数 5 種類の線形結合をモデルとし，過去に選択された通信手法による学習曲線の全サンプルを用いて非線形最小二乗法によりフィッティングを行った。

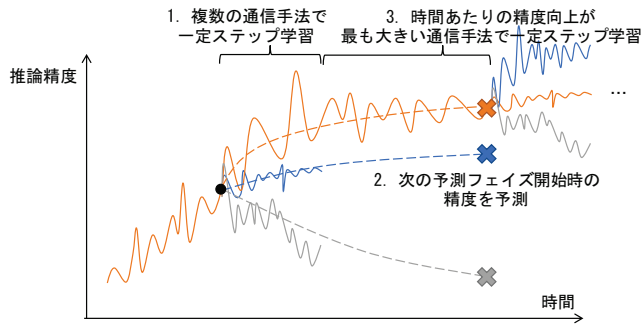

図 3: 適合的な通信手法の選択手法。

1. ～3. を繰り返すことで，その時点で最も時間あたりの推論精度の向上が最も高い通信手法で学習を行う。

5. 評価

本研究では提案手法を DL フレームワークの Caffe[7]に適用し，AlexNet[1]の一部のレイヤーの順番を入れ替えたネットワークである CaffeNet と GoogLeNet[8]の 2 つの CNN の学習について Reedbush-H 上で評価を行った。データセットとしては ILSVC 2012 データセット[3]のうちランダムに選択した 16 クラスのみを使用し，学習係数等のハイパーパラメータについては Caffe に付属しているサンプルファイルの設定を変更せずに用いた。また，ミニバッチサイズはどちらの CNN についても 256 とし，それぞれ 90，120 epoch の学習を行った。

学習に使用する並列数は，ミニバッチサイズ 256 の場合の 1 ステップの実行時間を計測した上で CaffeNet については 2 ノード(P100 GPU×4)，GoogLeNet については 8 ノード(P100 GPU×16)とした。また，ノード内(2 GPU)の勾配の加算については NVIDIA GPU 用の集団通信ライブラリ NCCL を用いて単精度で行い，その後のノード外の加算について前述の単精度・fp8・table(1)による加算を実装した。

提案手法を用いた際の CaffeNet の学習曲線を図 4 に示す。なお，提案手法の 1. と 3. で用いるステップ数としては，今回試行した内で特に提案手法の学習時間が短かったそれぞれ 200, 400 ステップを用いた。結果より，fp8 と table(1)を用いた場合は一定 epoch の学習時間について float と比較してそれぞれ 2.3，2.5 倍の高速化を達成した一方で，それぞれ top-1 accuracy(推論精度)はそれぞれ 1.7%，13.5%程度低下した。一方で提案手法(図 4 の“spec”)は 2.1 倍の高速化を達成し，かつ推論精度については float をわずかに上回る結果となった。提案手法で実際に選択された通信手法としては，初回のステップのみ float が選択され，その後は fp8, table(1), adaptive のいずれかが選択された。これにより，通信手法を適合的に変更することで推論精度を保ったまま学習することができたと考えられる。

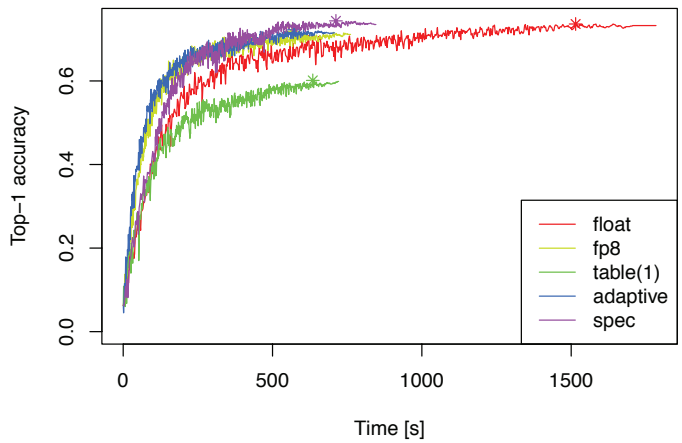

図 4: 提案手法(“spec”)を用いた CaffeNet の学習結果。
提案手法の学習曲線は各ステップで選択された通信手法のみによる経過時間を用いてプロットした。
“adaptive”はパラメータのテンソルごとに最も高速な通信手法を選択した場合の結果を表す。
各項目の星印は最も Top-1 accuracy が高い点を表す。

提案手法を用いた際の GoogLeNet の学習曲線を図 5 に示す。図 5 では 1. と 3. のステップ数としてそれぞれ 100, 200 を用いた。CaffeNet では通信手法により推論精度が悪化するケースがあった一方で，GoogLeNet を用いた評価ではいずれの通信手法についても最良 top-1 accuracy の差が 1%以内に収まった。この理由としては，CaffeNet と GoogLeNet のレイヤー数や畳み込みフィルタ等のネットワーク構造の違いや並列数の違いによって計算誤差に対する影響度合いが異なったためであると考えられる。

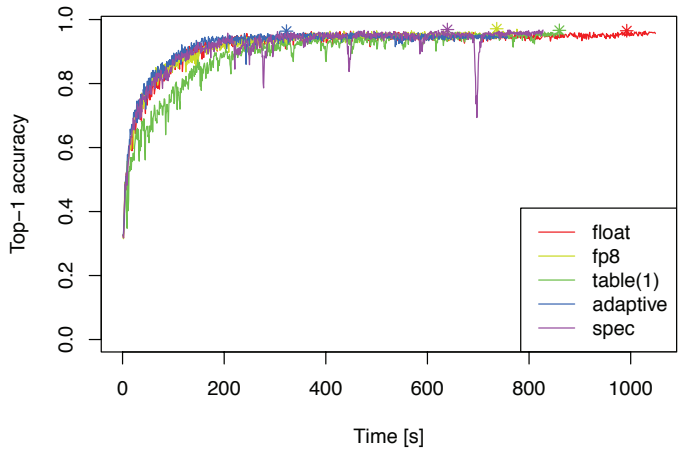

図 5: 提案手法(“spec”)を用いた GoogLeNet の学習結果。

6. まとめと今後の課題

結果より，適合的に通信手法を選択することによって単一の低精度通信手法を用いる場合に生じる悪化を防止できるケースが確認できた。一方で，本報告では提案手法 1. の部分については逐次的に通信手法の評価を行っており，実質的な学習時間の短縮は実現できていない。よって今後の課題としては，この投機的な学習についてマルチジョブを用いて並行に実行する手法を実装・評価することが挙げられる。

参 考 文 献

[1] A. Krizhevsky, I. Sutskever, and H. Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25 (NIPS2012), (11):1-9, 2012.

[2] T. Akiba, S. Suzuki, and K. Fukuda. Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes. arXiv e-prints, 2017.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, (3):211-252, 2015.

[4] 大山洋介, 野村哲弘, 佐藤育郎, 松岡聡, “ディープラーニングのデータ並列学習における少精度浮動小数点数を用いた通信量の削減,” 情報処理学会研究報告, Vol. 2017-HPC-158, No. 30, pp. 1-10, 2017.

[5] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech 2014), pp. 1058-1062, 2014.

[6] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2015), pp. 3460-3468, 2015.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv e-prints, 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2015), 2015.