CC-002

モデル分散深層学習への Approximate Computing 適用時のモデル精度・処理性能の評価
Evaluation of Model Accuracy and Processing Performance when Applying Approximate Computing to Model Distributed Deep Learning

菅　真樹[1)]　　中村　太[1)]
Masaki Kan　Dai Nakamura

1　はじめに

深層学習では、自然言語タスクなど大規模モデルの学習処理が課題になっている。大規模モデルの学習を行うためには、単一の計算機や GPU などのアクセラレータの演算性能やメモリ容量が限られているため、複数の計算機・アクセラレータを利用した分散深層学習が行われている。分散深層学習を実現する方法としては、データ並列やモデル並列処理が挙げられるが、大規模モデルを学習するためにはモデル分散処理が必要とされている。

本稿では大規模モデルの学習や推論の高速・省電力化のため、Approximate コンピューティングを用いたモデル分散処理の提案を行う。そこで，Approximate コンピューティング (以下、AC) の例として、 NW 転送部分に誤差を許容する Approximate ネットワーク (以下、AN)[1] や，転送データを量子化（低ビット化）した場合の学習精度への影響や学習処理時間について評価を行った．評価結果を踏まえ，転送データの特性に応じた量子化方式を提案し評価を行う．更に，モデル分散深層学習処理時のネットワーク帯域の性能へ影響を評価することで，AN の必要性について議論を行う．

2　分散深層学習と Approximate Computing

2.1　分散深層学習

近年深層学習モデルの大規模化が進み，学習処理に大量の計算リソースが必要になるため，様々な方法の並列処理，分散処理が行われてきた．

最も一般的に利用されているのは，データ並列処理であり，同一モデルの重みを複数のワーカーにコピーし，同時に演算を行うものである．これには，勾配の同期が必要となり，All-Reduce などの集合通信の採用や，Parameter Server などのアプローチが取られてきた．モデルが GPU などの演算装置のメモリに搭載可能であればこの方法を適用することが出来る．

しかし，モデルが単一の GPU に乗り切らない場合にはデータ並列を単純に適用することが出来ないため，ニューラルネットワークを分割し，それぞれの演算処理を複数の GPU や計算機に分割して行う，モデル並列方式が行われている．これらを単純に行うだけでは通信の待ち時間が発生するため処理が非効率となってしまうため，通信と計算をオーバラップして処理するため，Gpipe[2], PipeDream[3] などでは，ミニバッチをマイクロバッチに更に分割してパイプライン実行するアプローチが提案されている．また，滝澤ら [4] は，パイプライン並列処理の分割位置やパイプライン数による性能向上率への影響について評価を行っている．

2.2　Approximate コンピューティング

Approximate コンピューティング (AC) とは演算や NW などの精度を低下させる代わりに，性能向上および省電力化などを狙う技術である [5]．具体例としてはループ計算の処理の一ををスキップする Loop perforation, 低ビット演算により算術計算の近似や精度を低下，通信の非同期処理や通信エラーの許容 [1] などがある．また，データストレージ・メモリに対してもエラーを許容することが可能である．

AI モデルの学習ではユーザの要求レベルの推論精度を満たした学習モデルを作成することが目的であるため，演算器や NW などの計算機コンポーネントに AC を適用しても学習モデルの推論精度が維持できるか，あるいは推論精度が下がってもその代わりに省電力効果や処理性能の向上などの効果が得られるのであれば，有用であると考える．

現在の深層学習への AC 適用の例として，モデルの量子化 (Quantization) が上げられる．これらはモデル肥大化によるメモリ利用量の削減や，エッジ計算機での低消費電力演算を実現するために利用されており，基本的には学習処理は高精度の演算で行い，後から量子化することが行われている [6]. また，学習時・推論時の低ビット演算は幾つか提案されており，Tensor コアを活用して FP16 で演算する，混合精度計算 (Automatic Mixed Precision) は現在の Tensorflow や PyTorch で既に実装されている．更なる深層学習への低ビット化の採用については，シミュレーションレベルの検証が取り組まれている [7, 8, 9].

3　提案：モデル分散深層学習への AC 適用

本研究では，モデル分散深層学習への AC 適用方法について議論する．図 1 に本稿の提案概要図を示す．例として，ニューラルネット（例えば ResNet[10]）を分割し，計算機 2 台で分散実行する場合を示す．

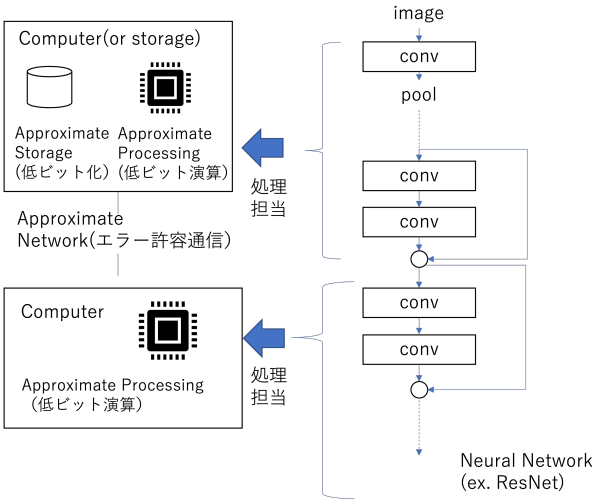

図 1　提案アーキテクチャ概要図

上部の計算機ではイメージデータの読み込み，モデル

1)　合同会社リトルウイング Little Wing, LLC

前段の学習・推論処理を行い，下部の計算機ではモデル後段の学習・推論処理を行う．このような構成において，AC を適用出来る要素としては，それぞれの計算機のストレージ，メモリ，演算器，計算機間を接続するネットワークなどが該当する．

それぞれの構成要素では，低ビット化や，エラーを許容したネットワークやメモリ読み書きなどが適用可能であると考えるが，実際に演算，ストレージ，ネットワークへそれぞれどの程度の Approximate 性を適用可能であるかは明らかではない．

例えば，ストレージのイメージデータにエラーが含まれていても学習への影響が少ないことが予想される．AI 学習では一般に，汎化性能を向上させるため，データにあえてノイズを加えるなどの Data Augmentation 手法が採用されており，AI 学習演算は元々データがビット単位で正当であることが求められている計算ではないためである．

一方，演算器のそれぞれがニューラルネットのモデルの各層に対してどの程度低ビット化してよいか，ネットワークはどの程度エラーを許容してよいかは明らかではない．また，処理性能は通信性能と演算性能のバランスで決まるため，単にネットワークを高速化しても分散処理においてネットワークがボトルネックでなければ AN を適用する意味が少ないため，学習精度と処理性能双方への影響を踏まえて検討する必要がある．

また，モデル前段を担当する計算機はストレージ内のプロセッサを活用することが想定できる（In-Storage Computing）．ストレージ装置内の計算リソースは限られていることが想定されるため，In-Storage Computing での前段モデルでの低ビット演算はより積極的に活用される可能性がある．

本研究では，モデル分散深層学習への Approximate Network 適用方法についてモデル精度，処理性能の両面で評価を行う．

4　評価

4.1　評価システム

本 AN の適用時のモデル分散深層学習への影響を評価するために，PyTorch を用いてモデル分散深層学習システムを実装した．実装システムでは，2 段のモデル分散深層学習処理をパイプライン並列で行う動作が可能である．

動作環境として Google Cloud Platform 上の V100 インスタンスを利用した．評価目的に応じて 1 インスタンスの 2GPU で分散させた場合と，2 インスタンスで 1GPU ずつ利用して分散処理をさせている．インスタンス間の実効帯域は，iperf で計測したところ約 15.6Gbits/sec であるため，インスタンス間の NW は 20GbitEther 相当であると予想される．

4.2　ネットワークにランダムビット反転エラーを適用した場合の学習精度への影響

本節では，ネットワークへランダムにビット反転するエラーを適用した場合の評価を行う．モデルは ResNet56 を利用し，データセットとしては CIFAR10 を利用した．また，本稿ではモデル分散深層学習の Forward 処理のデータ転送に対してのみエラーおよび後述の量子化（低ビット化）転送を行う．これは，本評価のモデル分散深層学習方式では，Backward のデータ転送時間よりも Forward のデータ転送処理時間の方が多く，演算器と NW 性能が変化した場合に先にボトルネックになるのは Forward のデータ転送処理と考えているためである．

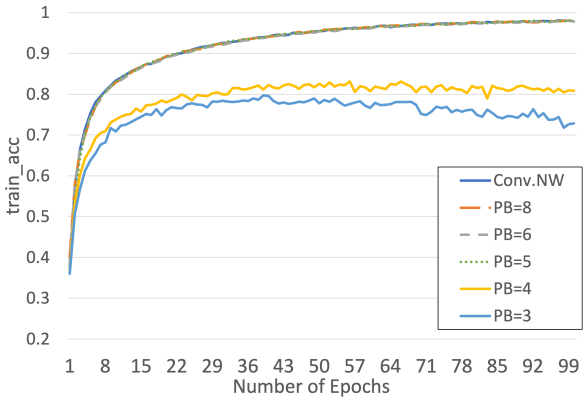

図 2　AN 適用時精度

図 2 に,ResNet56 の最初の Residual Block の後で分割した際のトレーニングデータに対する精度を示す．学習時に通信するデータは 32bit 浮動小数点の集合であるが，それらの上位ビットを保護し (256-QAM)，下位ビットにエラーを許容 (1024-QAM) する形で伝送することをソフトウェアで模擬した (なお，この条件では 8.8 倍の実効帯域向上が期待出来る [1])．図 2 に示すように，上位 5 ビット以上を保護することで精度に影響を与えず学習が可能であることが分かる．すなわち、エラーを許容した NW 伝送をモデル分散学習に適用出来ることが期待出来る．

4.3　送信データを量子化（低ビット化）した場合の精度低下

次に，転送データの 1 部のビットをカットすることで量子化（低ビット化）を行いデータ転送する場合の精度影響の評価を行った．本節の実験では ResNet56 のほぼ中間の Residual Block の後でモデルを分割した．通信データの低ビット化処理は，保護するビットを残し，それ以外のビットを送らず，データ受信後にビットを 0 として扱っている．計算処理は float32 に復元してから行っている．

量子化を行わない場合と，5 種類の量子化の方式，次節で説明する提案方式の量子化方式について評価を行った．5 種類の量子化方式は，符号部を送らずに指数部を全て送る（符号部なし 8+0 ビット），符号部を送らずに指数部を 1bit だけ送る（符号部無し 1+0 ビット），符号部を送り仮数部 7 ビットを送る（符号部あり 1+7 ビット），符号部を送り指数部 1 ビットだけ送る（符号部あり 2+0 ビット），符号部を送り指数部 6 ビットと仮数部 1 ビットを送る（符号部あり 7+1 ビット）の通りである．

評価結果として，図 3,4 にバッチサイズ 256，パイプライン並列数 4 で 50 エポック分の学習を行った場合の学習およびテストデータへの精度を示す．なお，精度差を明示するために学習精度の 0-80%の範囲の表示をしていない．

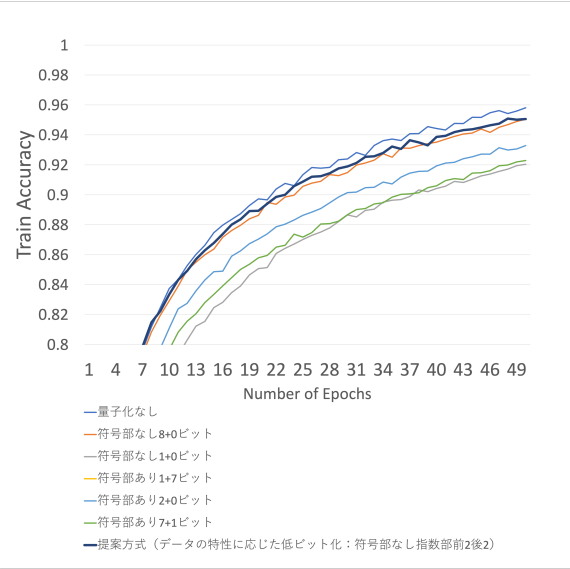

図 3　転送データ低ビット化（Train データ精度）

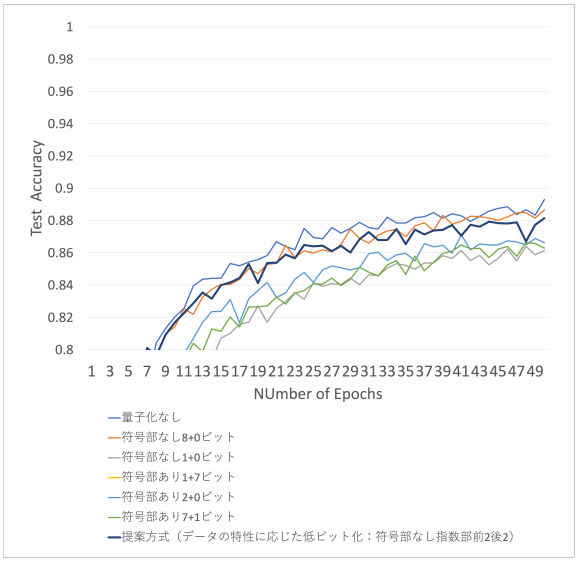

図 4　転送データ低ビット化（Test データ精度）

まず，指数部を送らない方式は（符号部無し 1+7 ビット）学習が全く進まなく精度が約 10 % のままであるため，グラフの表示範囲にない．つまり，指数部を送るかどうかが学習に大きな影響を与える．

一方，符号部の有無は精度への影響は少なく，指数部を全て送った符号部なし 8+0 ビットについては量子化なしと近い精度を示した (Train データへの精度で約 0.8 %, Test データの精度で約 0.6 %の精度低下)．1 ビットのみを送る符号部なし 1+0 ビットでも，約 3.5%程度の精度劣化で済んでいることが示されている．また，学習データの精度とテストデータへの精度は同様の傾向を示し，転送データの量子化によって汎化性能が大きく犠牲になるということは無いことが分かる．

多くのデータを転送しなくても精度への影響が少ない理由は，ResNet の Residual Block 単位で分割しているためと考える．つまり，ReLu 関数を通った後であるため，ビット数が削減されても，下限の値か，x>0 かどうかは情報として後段モデルに伝わるため，学習が進むことが予想される．符号部を送る必要がないのも同様の理由である．一方，本稿で評価を行っていないが，Backward に適用した場合には，勾配を送るため符号部が重要になると考えており，更なる評価が必要と考える．

4.4　データ分布に応じた量子化転送

4.3 節の評価を踏まえると，転送データの特性に応じた量子化が必要であると考える．

図 5 にモデル分散時の転送データのうち，1Epoch 目の処理から 100 万個のデータを抽出し，ヒストグラムを記載したものである．転送データは 32bit の浮動小数点であり，X 軸は 8bit の指数部表現を示し，Y 軸は出現回数である．ReLu 関数の後であるため，約 18%のデータが 0 であることが分かる．また，約 99.99%以上のデータが指数部表現で上位 2 ビット，下位 2 ビットの範囲に含まれていた．極めて少数であったためグラフに表現されていないが，0.006%のデータが指数部表現において 1000010 の範囲に含まれている．

学習が進んだ際にも同様の特性を示すかどうかは定か

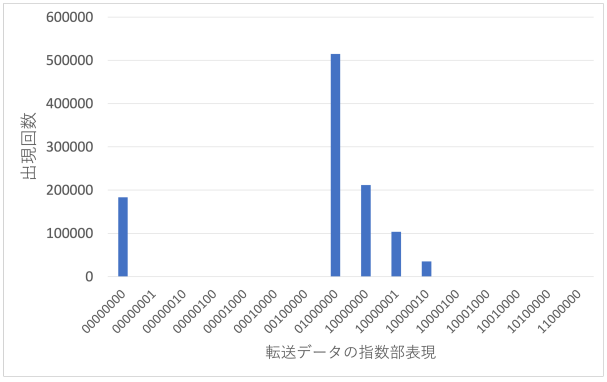

図 5　転送データのヒストグラム

ではないが，この結果を踏まえ，データ転送の低ビット化方式として，指数部上位 2bit, 下位 2bit を送り，その他のビットについては 0 として扱う方式を提案する．なお，0.006% のデータを考慮しなければ下位 1bit にすることも可能である．この方式でデータを転送した結果についても，前述の図 3,4 内に含めている．

提案方式の 50 エポック目の Train データに対する精度と，Test データに対する精度は，それぞれ量子化無し（32 ビット転送）と比較して，約 0.8%, 約 1.0% の精度低下であった．これは，指数部 8bit を全て転送した場合とほぼ同程度の精度であることが分かり，提案方式により約 2 倍のデータ転送効率が得られることが分かる．

4.2 節と合わせて考察すると，指定ビット以外を 0 に置き換えるよりも，ランダム反転エラーを適用した場合の方が精度低下が低い傾向にある．これは，指定ビット以外を一律に 0 にすると全ての情報が失われ，量子化された値のみが後段に利用されるが，ランダム反転の場合は一定確率で一部の情報が残るため，それが好影響を与えると予想される．

4.5　通信性能と処理性能の関係

本節では NW 帯域のモデル分散深層学習への性能影響について評価を行う．ただ，本稿で開発した 2 ノード，V100 1 枚ずつのモデル分散深層学習システムでは，NW 帯域は充分高速であり性能ボトルネックはほぼない

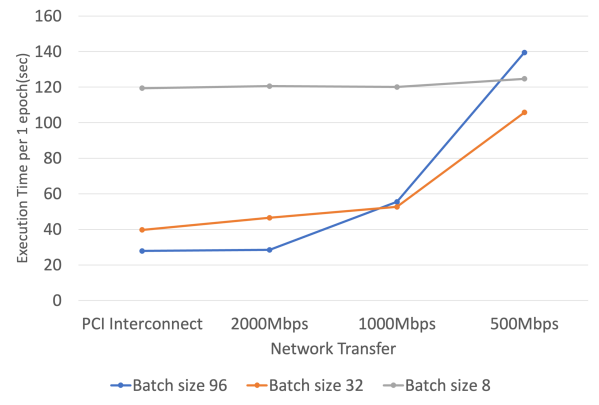

図 6　ImageNet 学習に対する転送帯域変化時の処理性能への影響

ため，ノード間のネットワーク通信に TC を用いて帯域制限を行うことで，低帯域 NW 環境での性能評価を行う．

まず，CIFAR10 および ResNet56 を用い，バッチサイズ 256，パイプラインのための分割数が 4 の場合の処理性能について表に示す．

帯域	1Epoch 分の学習時間 (秒)
20Gbps	33.54
1000Mbps	33.29
500Mbps	33.27
100Mbps	138.58

表 1　CIFAR10, ResNet56 時の帯域毎の処理性能

本実験条件では，100Mbps まで帯域を制限しないかぎり大きな性能悪化は発生しないことが分かる．これは，本条件のモデル分散深層学習処理が，1 マイクロバッチごとの Forward や Backward の演算処理時間とそれに伴う次段の NW 転送の時間を比較した際に，演算処理時間が大きいためである．一方，100Mbps まで帯域を制限することで性能悪化となることから，このような環境では ApproximateNW を導入して 5 倍以上の NW 帯域を確保することで性能が向上できることが分かる．

一方，データサイズやモデルサイズが大きくなると，NW ボトルネックになりやすい．ImageNet を用いた性能評価結果を 図 6 に示す．本実験ではモデル分割位置を 3 つめの Residula Block に固定し，NW 帯域性能を比較した．バッチサイズが大きい場合（処理性能重視），1,000Mbps でも NW 帯域が不足することが分かる．

以上を踏まえると，データサイズおよびモデルサイズ，バッチサイズを大きくすることで，Forward 処理のデータ転送量が増加するため，AN を適用しネットワーク帯域を仮想的に向上させることで処理性能を向上させることが可能であることが分かる．演算部に低ビット化などの AC を導入することで演算処理が向上するとそれに応じて必要なネットワーク帯域が向上するため，それに合わせてネットワーク帯域を向上させることが必要であると考える．

5　おわりに

本研究では、モデル分散深層学習に対して AC を適用した場合の精度や性能への影響を評価した．まず，モデル分散深層学習のネットワーク転送について AN を適用し，上位ビットを保護し (256-QAM)，下位ビットにエラーを許容 (1024-QAM) し，データをランダムにビット反転するエラーを適用する場合には，約 8.8 倍の NW 帯域向上に加え，学習精度を維持が可能である．転送データ量子化 (符号部無し，指数部のみ 8bit 転送 NW 帯域 4 倍：32bit->8bit) では Train データへの精度で約 0.8 %, Test データの精度で約 0.6 %の精度低下，転送データ量子化 (符号部無し，指数部 1bit では，NW 帯域 32 倍：32bit->1bit) では約 3.5 % の精度低下することを示した．さらに，転送データの特性を踏まえ，指数部の上位 2bit，下位 2 ビットのみ送る方式（転送 NW 帯域 8 倍：32bit->4bit) を提案し，8 ビット転送した条件とほぼ同精度の学習が可能であることが分かった．

また，NW 帯域を変更し処理性能を計測すると，ネットワーク帯域が細い環境では，ネットワークボトルネックにより処理性能が悪化することを示し，AN 適用可能性があることを示した．今後の課題としては，Backward 通信への AN 適用，演算処理に対しても AC を適用し，演算とネットワーク双方の Approximate 化の影響について評価を行うことが挙げられる．

謝辞

この成果は，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 (JPNP16007) の結果得られたものです．

参考文献

[1] 藤原, 藤木, 石井, 松谷, 天野, 鯉渕：“Approximate Hybrid HPC ネットワークの研究”, FIT2016: 第 15 回情報科学技術フォーラム (2016).

[2] Y. Huang, et al.: “GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism”, Curran Associates Inc., Red Hook, NY, USA (2019).

[3] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons and M. Zaharia: “Pipedream: Generalized pipeline parallelism for dnn training”, Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP ’19, New York, NY, USA, Association for Computing Machinery, pp. 1–15 (2019).

[4] 滝澤, 矢崎, 石畑：“パイプライン並列分散深層学習の一実装手法の評価”, 情報処理学会論文誌, **63**, 5, pp. 1206–1215 (2022).

[5] A. Agrawal, et al.: “Approximate computing: Challenges and opportunities”, 2016 IEEE International Conference on Rebooting Computing (ICRC), pp. 1–8 (2016).

[6] R. Krishnamoorthi: “Quantizing deep convolutional networks for efficient inference: A whitepaper”, CoRR, **abs/1806.08342**, (2018).

[7] R. Banner, I. Hubara, E. Hoffer and D. Soudry: “Scalable methods for 8-bit training of neural networks”, Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18, Red Hook, NY, USA, Curran Associates Inc., pp. 5151–5159 (2018).

[8] T. Zhang, Z. Lin, G. Yang and C. D. Sa: “Qpytorch: A low-precision arithmetic simulation framework” (2019).

[9] K. Higuchi, C. Matsui, N. Misawa and K. Takeuchi: “Computation-in-memory simulation platform to investigate inference accuracy degradation by device non-ideality interactions in deep neural network applications”, Japanese Journal of Applied Physics, **61**, SC, p. SC1054 (2022).

[10] K. He, X. Zhang, S. Ren and J. Sun: “Deep residual learning for image recognition”, CoRR, **abs/1512.03385**, (2015).