Vol.2018-HPC-165 No.1
2018/7/30

深層学習におけるBatchNormalization使用時の計算時間と精度の関係性

八島 慶汰[1,a)]　大山 洋介[1]　松岡 聡[2,1]

概要：近年，機械学習，深層学習と呼ばれる手法が盛んに研究されており，画像認識・音声認識・自然言語処理等の広くの分野で他の手法に比べ非常に優れた結果や精度を達成している．特に，ニューラルネットの並列学習の手法であるデータ並列の学習下では，各 GPU が分割されたデータセットを入力としてそれぞれ独立に計算を行い，パラメーターサーバーや All Reduce 等の計算を使用しモデルの更新を行っている．また，ニューラルネットの学習では勾配消失問題や出力の分布を一定に操作するために多くの正則化手法が提案されている．特にその効果が大きいと期待されている Batch Normalization (Batch Norm) が多く用いられており, 既存の Batch Norm では GPU 毎に独立して正則化が行われている．本論文では GPU デバイスで独立に正規化が行われた場合と GPU デバイス間においてまたがるように正規化が行われた場合で精度や計算時間等にどのような影響を与えるのかどうかを実験した．

1. はじめに

近年，画像認識や自然言語処理を始めとする数多くの分野で深層学習を用いた手法が広く用いられ優れた結果を達成している．深層学習とはニューラルネットと呼ばれるネットワーク構造を成している計算モデルについて，エッジの重みを学習データによって最適化する手法である．ニューラルネットの構造はより深く複雑な構造となりその結果ネットワークの表現力が増したため他の手法に比べ高い精度を達成する事が可能となった．また 2 つのニューラルネットを競合し学習させる GAN や強化学習を用いてニューラルネット自身の構造を学習する研究も行われている．一方で深層学習では深く複雑化したニューラルネットにより誤差関数の勾配が前層の方まで伝わらず消失してしまう勾配消失の問題が生じ，モデルが適切に学習を行えない場合や過学習を起こすなどの問題が生じてしまう．
この問題に対してニューラルネットワーク中では数多くの計算手法や正則化手法が存在している．本論文では Batch Norm の手法について，定義通りに GPU 間に通信と同期を伴い正規化を行う場合と GPU ごとに独立して正規化を行い通信を省略する場合とで精度や計算時間にどのような変化が現れるのかなどを調査した．

1　東京工業大学　情報理工学院　数理・計算科学系
2　理化学研究所　計算科学研究センター
a)　yashima.k.ac@m.titech.ac.jp

2. 関連研究

2.1 Deep Neural Network, AlexNet

AlexNet とは画像，動画認識をタスクとした大会である ILSVRC において, それまでに比べて高い認識率で優勝を収めた Alex Krizhevsky らの考案した Deep Neural Network である [4]．このニューラルネットは畳み込み 5 層と全結合層 3 層から構成されるネットワーク構造を成している．今まで人力で行っていた画像からの特徴量の抽出の手法の設計などをニューラルネット自らが学習を行い高い認識性能を可能にした．またこのネットワークでは ReLU 関数や Dropout などの計算技術や GPU を使用した並列計算などの現在深層学習で多く用いられている構成要素がすでに導入されている．この論文中では GPU のメモリ容量の影響でモデルを複数の GPU に分割して計算を行っている．

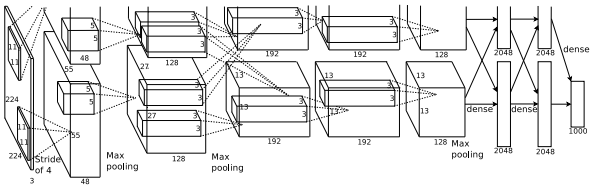

図 1　AlexNet の構造 [4]

2.2 ニューラルネット中の正則化手法

誤差逆伝播法で深く複雑なニューラルネットの学習を進める際，活性化関数の微分係数が複数回乗算される影響により活性化関数の出力の分布が偏っていると勾配消失問題が起こりやすくなり，その結果として適切に学習が進まなくなる恐れがある．その解決策としてニューラルネット中の活性化関数に ReLU 関数を使用したり，重みや出力を正則化し出力の分布を操作する手法が多く用いられている．ReLU 関数を使用することで誤差逆伝播の際に微分が 0 になり勾配が消失してしまうことを防ぐ事ができる．各レイヤーの出力ごとに正則化を行う Layer Normalization [1] や，出力のチャンネルごとに正則化を行う Group Normalization [8]，ニューラルネットの重み自体を正則化する Weight Normalization [5] などの多くの手法が研究されている．
以下では入力を x_i，正規化された出力を y_i，また 0 除算を防ぐための小さな数を ϵ とする．

Layer Normalization

H はレイヤー内ユニットの総数を表している．

$$\mu = \frac{1}{H}\sum_{i=1}^{H} x_i, \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i-\mu)^2}$$
$$y_i = \frac{g_i}{\sigma}(x_i - \mu)$$

Group Normalization

特定レイヤーの出力について，チャンネルをグループ分けしそのグループのチャンネル内ごとに正規化を行う．チャンネルのグループ分けする数を $\#(G)$ で表すと以下のようになる．

$$S_i = \left\{ k \middle| k_N = n \in N, \left\lfloor \frac{k_C}{C/\#(G)} \right\rfloor = \left\lfloor \frac{i_C}{C/\#(G)} \right\rfloor \right\}$$
$$\mu = \frac{1}{m}\sum_{k\in S_i} x_k, \sigma = \sqrt{\frac{1}{m}\sum_{k\in S_i}(x_k-\mu)^2 + \epsilon}$$
$$y_i = \frac{x_i - \mu}{\sigma}$$

Weight Normalization

ニューラルネット中の重みベクトル W についてベクトル v とスカラー g を用いて分解する．これらは誤差関数の勾配から計算される．

$$W = \frac{g}{||v||}v$$

このように重みを分解することで，重みのパラメータ数が少ない場合に Batch Norm などでは計算時間が多く生じてしまう問題に対し，少ない時間で同等の正規化が行える．

2.3 Batch Normalization

Batch Normalization とは Google の Sergey loffe によって提案された手法である [3]．ある特定の層の出力を平均が 0，分散が 1 になるように以下のように正規化を行う．Batch Norm 直前の入力を $X = \{x_1, x_2, x_3, \cdots x_n\}$，正規化した出力を $Y = \{y_1, y_2, y_3, \cdots y_n\}$ とすると (ただし n はバッチサイズ) 以下のように正規化した出力 y_i に線形変換を行い次層の入力として渡している．

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2$$
$$y_i = \frac{x_i - \mu}{\sqrt{\sigma^2+\epsilon}}$$

正規化を行うことで出力の分布が一定となり安定した学習が可能になる．その結果学習率に高い値を設定することでより早い学習の収束や，学習精度 (汎化性能) の向上が見込める．

3. 実験

3.1 実験環境

本調査の実験では東京工業大学学術国際情報センターの TSUBAME-KFC を使用した．TSUBAME-KFC では 1 ノードに CPU2 基，GPU4 基が搭載されている．以下表 1 に TSUBAME-KFC 1 ノードの実行環境を記す．

表 1 TSUBAME-KFC の実行環境
1 ノードあたりの構成を示している

CPU	Intel(R) Xeon(R) CPU E5-2620 v2 ×2
周波数	2.1 GHz
物理コア数	6
論理コア数	12
L3 キャッシュ	15 MB
メモリ	64 GB
GPU	NVIDIA Tesla K80 ×4
単精度 FLOPS	5.6 TFLOPS
メモリ	計 24 GB, GDDR5
メモリバンド幅	計 480 GB/s
インターフェース	PCI Express Gen3 × 16, 16 GB/s
ECC	無効
Auto Boost	有効
インターコネクト	4X FDR InfiniBand
バンド幅 (実行)	6.8 GB/s
NIC インターフェース	PCI Express Gen3 × 8, 8 GB/s
トポロジー	two star-coupled network
スイッチ	Mellanox SX6036
OS	CentOS 7.3.1611

3.2 実験設定

学習のモデルは ILSVRC の 2014 年において提案された畳み込み層 13 層と全結合層 3 層からなる VGG16 と呼ばれるニューラルネットを使用した [6]．モデルの構成は図 2 のようになっている．すべての畳み込み層の直後に Batch Norm を使用し正規化されたものを線形変換しその出力を次層の ReLU 関数の入力としている．
学習, テスト用のデータセットとして Cifar-10 を使用した．Cifar-10 は 32× 32 のサイズ，RGB カラーからなり学習データ 5 万枚テストデータ 1 万枚から構成されている．モデルの学習には 5 万枚の学習データ，精度の計算には学習データとテストデータの両方を使用した．

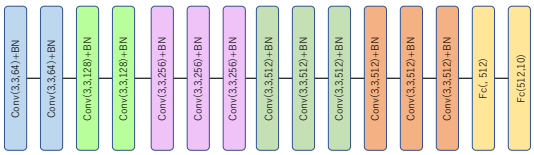

図 2　VGG-16 の構成図

GPU 間で独立に Batch Norm を行う場合，2.3 で説明したような計算を各ワーカー GPU で処理している．
GPU 間にまたがるように Batch Norm を行う場合、各 GPU で平均と二乗平均と平均を計算しそこから正規化を行う．
例として 2GPU で行う場合, 計算式で表すと以下のようになる．Batch Norm 直前の各 GPU の入力と正規化された出力を以下のように表す．

$$
\begin{aligned}
X_{GPU1} &= \{x_{1\_gpu1}, x_{2\_gpu1}, x_{3\_gpu1}, \cdots x_{n\_gpu1}\} \\
X_{GPU2} &= \{x_{1\_gpu2}, x_{2\_gpu2}, x_{3\_gpu2}, \cdots x_{n\_gpu2}\} \\
Y_{GPU1} &= \{y_{1\_gpu1}, y_{2\_gpu1}, y_{3\_gpu1}, \cdots y_{n\_gpu1}\} \\
Y_{GPU2} &= \{y_{1\_gpu2}, y_{2\_gpu2}, y_{3\_gpu2}, \cdots y_{n\_gpu2}\}
\end{aligned}
$$

これを GPU 間にまたがって正規化を行う場合次のような計算を行う．

$$
\begin{aligned}
\mu_{gpu1} &= \frac{1}{n}\sum_i x_{i\_gpu1} \\
\mu_{gpu2} &= \frac{1}{n}\sum_i x_{i\_gpu2} \\
\overline{\mu} &= \frac{1}{\#(GPU)}\sum_k \mu_{gpuk} \\
\nu_{gpu1} &= \frac{1}{n}\sum_i x_{i\_gpu1}^2 \\
\nu_{gpu2} &= \frac{1}{n}\sum_i x_{i\_gpu2}^2 \\
\overline{\nu} &= \frac{1}{\#(GPU)}\sum_k \nu_{gpuk}
\end{aligned}
$$

平均と二乗平均を各 GPU で計算し All Reduce 処理を行い GPU 間に渡る平均と二乗平均を計算する．All Reduce で計算された平均と二乗平均の 2 つからデータの正規化を行う．

$$
\begin{aligned}
\overline{\sigma^2} &= \frac{1}{\#(GPU)}\sum \sigma_{gpu}^2 \\
&= \frac{1}{\#(GPU)}\sum\left(\frac{1}{N}\sum x_i^2 - 2\overline{\mu}\frac{1}{N}\sum x_i + \overline{\mu^2}\right) \\
&= \frac{1}{\#(GPU)}\sum\left(\frac{1}{N}\sum x_i^2\right) - \overline{\mu^2} \\
&= \overline{\nu} - \overline{\mu^2} \\
y_i &= \frac{x_i - \overline{\mu}}{\sqrt{\overline{\sigma^2} + \epsilon}}
\end{aligned}
$$

以上のようにしてデータの正規化を行う．GPU 間で独立に正規化を行う場合と同様にして，正規化された出力データ Y を線形変換し次の層の ReLU 関数の入力として利用する．

3.3 実験結果

今回は実験モデルのニューラルネットに対して GPU 間で独立に正規化を行う場合と GPU 間にまたがって行う場合とを比較する．すべての層で Batch Norm を行った場合，各 GPU でのバッチサイズを変化させた場合で実験を行った．

Batch Norm を各 GPU 間でまたがるように行う場合

バッチサイズを 64 と固定にし，畳み込み層の直後に使用している Batch Norm を GPU 間で独立に行った場合 (normalBN)，GPU 間にまたがるように行う場合 (multigpuBN)，Batch Norm を行わない場合 (noBN) の 3 種で比較した．

図 3 を参照すると，Batch Norm を行った場合の 2 つと行わなかった場合とで学習の loss の下がり具合や accuracy の上昇率に差は現れているのでニューラルネット中に Batch

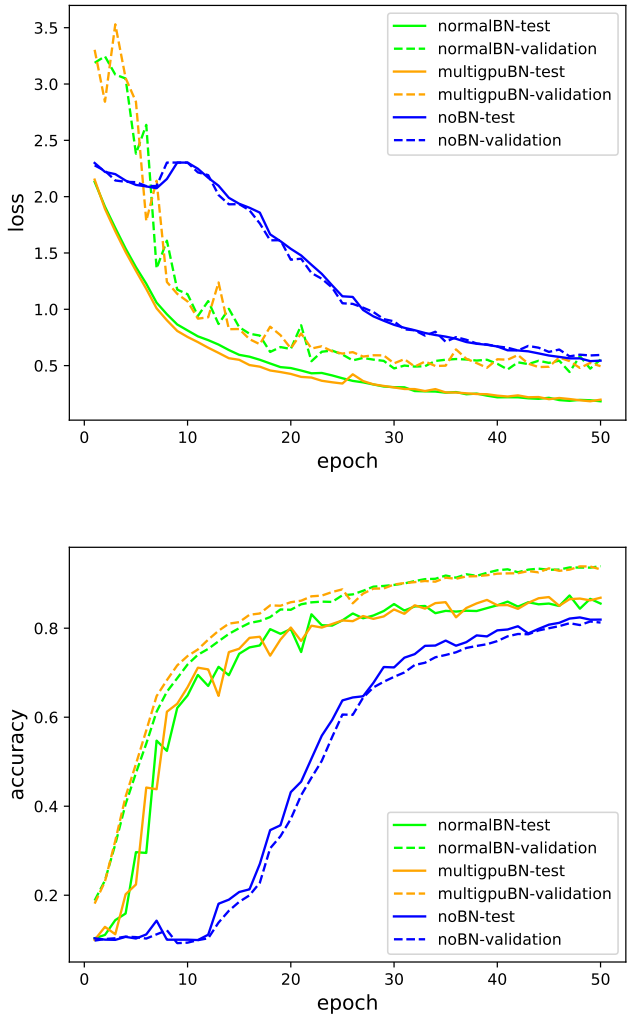

図 3　BatchNorm の使用法による Loss,Accuracy の変化

Norm を使うことは有効であることが確認できる．しかし GPU 間でどのように使用しているのかの 2 種を比較した際,GPU 間にまたがって Batch Norm を使用しても学習精度に差異が無いことがグラフから読み取れる．また計算時間の面で比較すると，GPU 間にまたがるように行った場合のほうが約 1.5 倍ほど増加した (表 2 参照)．

表 2　バッチサイズ=64 の場合の総実行時間

Batch Norm の使用法	実行時間 (sec)
normalBN	804
multigpuBN	1211
noBN	748

バッチサイズを変化させて行う場合

各 GPU でのバッチサイズを 2 から 64 まで変化させ学習にどのような変化が生じるのかの実験を行う．バッチサイズが 2 枚の小さなサイズで学習を行った場合，各 GPU で独立に正規化行うと学習がうまく進まなかったのに対して GPU 間にまたがるように行うと学習が適切に進んだ．バッチサイズを 64 で各 GPU で独立に正規化を行った場合とバッチサイズ 2 で GPU 間にまたがるように正規化を行った場合の学習結果を図 4 に記す．
学習データに対する正解率に差は生じているもののテストデータに対する正解率はほぼ同等の数値のために両者のモデルは同じくらい汎化していると言える．この結果から GPU で独立に正規化を行う際に，バッチサイズがとても小さい場合は学習が進まないのに対して GPU 間にまたがり正規化を行うことで学習がうまく進むことが確認できる．

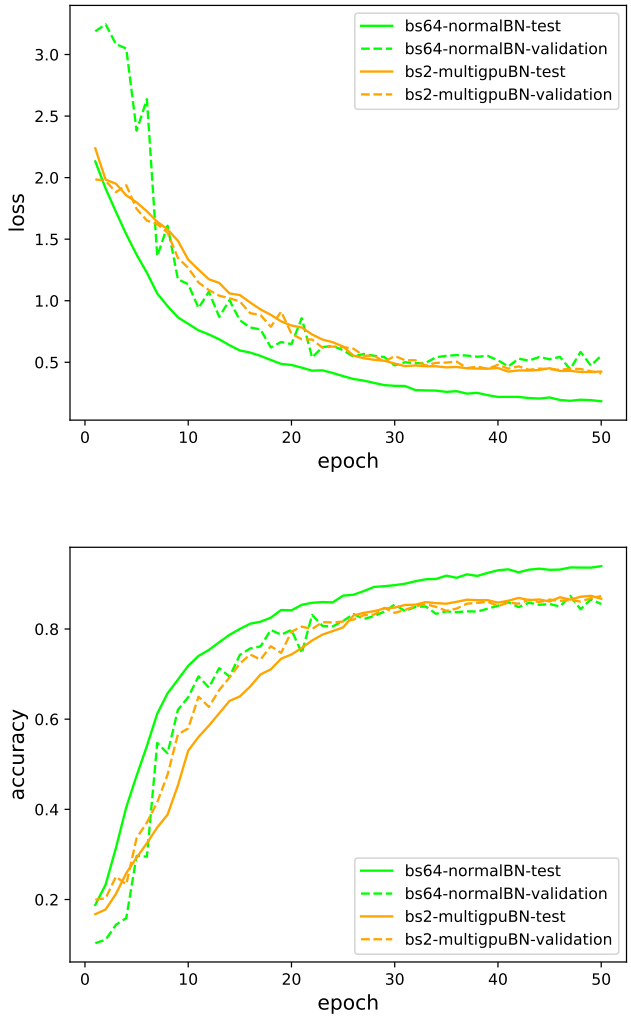

図 4　バッチサイズと BatchNorm の使用法による Loss,Accuracy の変化

バッチサイズが各 GPU で 4 枚の場合についての結果を図 5 に記す．
バッチサイズが 2 の場合と異なり，各 GPU で学習を行っても学習は進むがバッチサイズがある程度の大きさを持っている場合と比較して進行状況や精度が悪いことが確認できる．バッチサイズが 4 枚で Batch Norm の使用法が違う

2 種を比較した際，GPU 間にまたがるように正規化したほうが適切に学習が進んでいる．

3.4 考察

Batch Normalization ではバッチサイズが小さい場合についてデータの正規化が適切に行われないため学習に影響を及ぼしていることが確認できる．またそのような問題に対して GPU 間にまたがるような正規化を行うことでデータの正規化が正しく進み学習が安定して進むことも確認できた．

Cifar10 や ImageNet のような画像では小さいバッチサイズが必要となることは少ないが，動画解析やより大きなサイズの画像を入力としてニューラルネットで学習させる際に GPU のメモリなどの影響でバッチサイズが小さくなってしまう場合，GPU 間にまたがるように正規化を行うことで学習がより安定して進められることなどが考えられる．計算時間についても GPU 間にまたがり正規化を行う際に同期の必要性があるために実行時間が長時間かかってしまうことが考えられるが，実際の 1 イテレーションの間に何が原因でどれほどのボトルネックとなっているのかなどをより精密に検証する必要がある．

4. おわりに

本実験ではデータセットとして Cifar-10 の画像データを使用し Batch Norm に関する実験を行ったが，有効性を確かめるためにはより大きなデータセットでより深いニューラルネットを使用し検証を行う必要がある．

より深いニューラルネットでの実験

今回使用した VGG-16 のモデルでは 16 層からなるニューラルネットだが，GoogLeNet [7] や ResNet [2] のような 100 層以上の非常に深いニューラルネットワークでの実験を行い，実際に Batch Norm の使用がどのような影響を及ぼすのかを検証する必要がある．またニューラルネット中でも前方の層や後方の層のみに焦点を当てて Batch Norm の有効な使用法についても同様に検証する必要がある．

より大きなデータセットでの実験

最適な使用法の研究について，本実験ではデータセットとして Cifar-10 を使用しているが ILSVRC で実際に使用されている ImageNet などの 256×256 ピクセルのような大きなサイズの画像で 1000 クラス分類を行いデータセットの大きさによりどのように変化するのかを実験する必要がある．

謝辞

本研究は，産総研・東工大実社会ビッグデータ活用オープンイノベーションラボラトリ (RWBC-OIL) の活動として実施し，JST CREST(JPMJCR1303, JPMJCR1687) の支援を受けたものである．

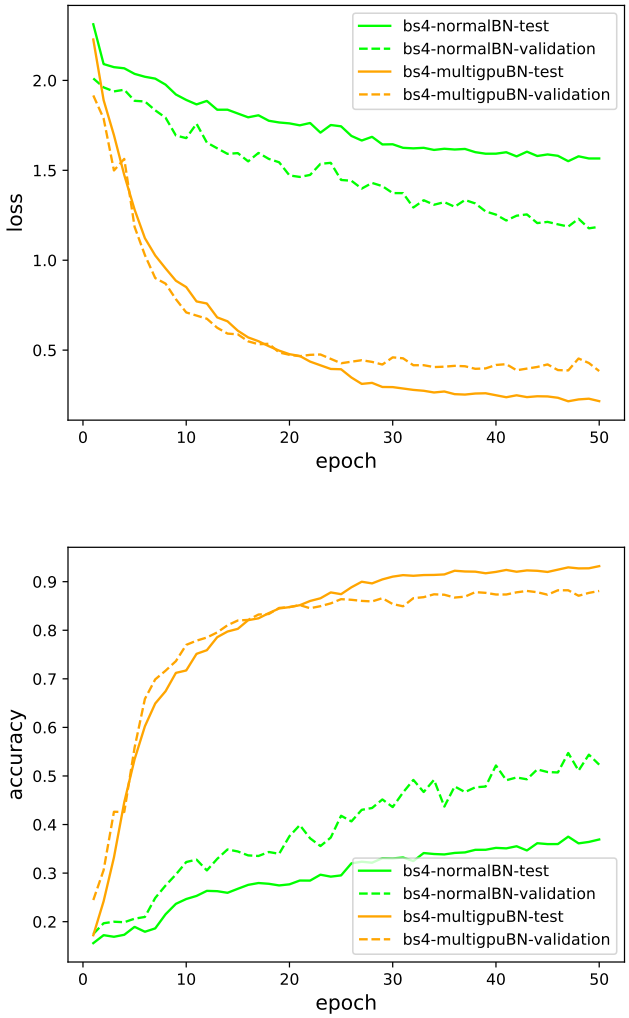

図 5 バッチサイズと BatchNorm の使用法による Loss,Accuracy の変化

参考文献

[1] Ba, J. L., Kiros, J. R. and Hinton, G. E.: Layer normalization, *arXiv preprint arXiv:1607.06450* (2016).
[2] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016).
[3] Ioffe, S. and Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv:1502.03167* (2015).
[4] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, pp. 1097–1105 (2012).
[5] Salimans, T. and Kingma, D. P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks, *Advances in Neural Information Processing Systems*, pp. 901–909 (2016).
[6] Simonyan, K. and Zisserman, A.: Very deep convolutional

networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556* (2014).
[7] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. et al.: Going deeper with convolutions, Conference on Computer Vision and Pattern Recognition (2015).
[8] Wu, Y. and He, K.: Group normalization, *arXiv preprint arXiv:1803.08494* (2018).