B-015

マルチ GPU 上での CUDA 実装による 深層畳み込み敵対的生成ネットワークの並列処理

Parallelization of DCGAN by CUDA implementation on mult-GPU

根本　祐輔 †　　吉田 明正 †
Yusuke Nemoto　　Akimasa Yoshida

1　はじめに

深層学習を応用した敵対的生成ネットワーク（GAN）では，データから特徴を学習することで実在しない画像を生成することができる．この GAN を発展させた DCGAN は，生成器と識別器に畳み込みを使用することで GAN よりも精度の高い画像を生成することができるが，実行時間が長くなる．機械学習の実装には Python が用いられることが多いが，本稿では複数 GPU を利用したモデル並列処理を実現するために，CUDA と OpenMP による実装を行う．本手法では，DCGAN の生成器と識別器を，それぞれ各 GPU に割り付け，2 台の GPU 上で DCGAN のモデル並列処理を実現する．性能評価では，NVIDIAQuadro RTX6000 を 2 台搭載した Xeon サーバを使用してカラー画像生成を行っており，提案手法の有効性が確認された．

2　敵対的生成ネットワーク

本章では，敵対的生成ネットワーク (GAN) 並びに深層畳み込み敵対的生成ネットワーク (DCGAN) について述べる．

2.1　GAN

GAN(Generative adversarial networks) とは，生成器と識別器の 2 つのネットワークが競い合うように学習する生成モデルの 1 つである．生成器は，ランダムなノイズ (標準正規分布) を入力として偽物のデータを生成する．識別器は，生成器で生成した偽物のデータと本物のデータを入力として本物か偽物かを数値で出力する．それぞれ識別結果から逆伝播によりパラメータを更新する．パラメータを更新していき，より誤差のないデータを生成していくモデルである．

2.2　DCGAN

DCGAN(Deep Convolutional GAN) とは，生成器と識別器の 2 つのネットワークに畳み込みを組み込んだモデルである．全結合型の GAN では，各ニューロンが独立しており，ピクセル間の関係性を埋め込むことができない．そこで，畳み込みニューラルネットワークを使用することで学習の精度を向上させる．しかし，層の内部構造が複雑になり多層化したことから従来の GAN よりも学習時間が長くなるという問題がある．本稿では，マルチ GPU を用いて 2 つのネットワークを並列に処理させることで高速化を実現する．先行研究として，NN モデルの高速化 [1] や DCGAN のモデル軽量化などが提案されている [2][3]．

†明治大学大学院先端数理科学研究科ネットワークデザイン専攻
Major of Network Design, Graduate School of Advanced Mathematical Sciences, Meiji University

```
#pragma omp parallel
{
     #pragma omp private(p)
     for (int ep =0; ep < 300; ep++){
          for (int id =0; id < 46; id++){
               p=omp_get_thread_num();
               if(p==0){
               //生成器実行
               }else if (p==1){
               //識別器実行
               }
               #prgma omp barrier
          }
     }
}
```

図 1　DCGAN の OpenMP 実装の部分コード.

3　マルチ GPU を用いた DCGAN のモデル並列

本章では，DCGAN の生成器と識別器のモデルをそれぞれ GPU に割り当て，モデル並列で実行する手法について述べる．

3.1　DCGAN のモデル並列

まず，GPU 上でプログラムを実行するために CUDA を実装を行う．CUDA コードでは，カーネル関数ごとに最適なブロック数とスレッド数を決めることで高速化を図る．畳み込み部分は計算時間の大半を占めるため，処理を複数に分けて実行し速度向上を実現する．また，CUDA カーネルで書かれたコードは，OpenMP を使って 2 スレッド生成され，2GPU で実行される．DCGAN の生成器の処理を GPU 番号 0，識別器を GPU 番号 1 として実行する．

3.2　マルチ GPU のための CUDA コード

DCGAN の畳み込み層などの層に含まれる計算部分を CUDA で記述する．バッチノーマライゼーション関数と ReLU 関数のようにひとまとまりにすることが可能であれば，二つの関数を一つのカーネル関数で記述することで効率化する．そして，カーネル関数ごとに最適なブロック数とスレッド数を定義することで高速化を図る．

3.3　マルチ GPU のための OpenMP コード

マルチ GPU での DCGAN の部分コードを図 1 に示す．図 1 で変数 p はスレッド番号を表しており，0 の場合生成器の学習，1 の場合識別器の学習を行う．最後に，2 つのネットワークの学習で時間差が発生するためバリアで同期を取る．

表 1　性能評価に用いるマシン．

CPU	Intel(R) Xeon(R) 2265 3.50Ghz 12Core
GPU	NVDIA Quadro RTX 6000 24GB × 2
メモリ	DDR4-2933 REG ECC 32GB × 4(128GB)
OS	Ubuntu18.04LTS
CUDA	10.2.89

表 2　RTX 6000 上での DCGAN の実行時間．

	CuPy	PyCUDA	CUDA	CUDA
GPU 数 [台]	1	1	1	2
1 エポックの処理時間 [s]	15.69 (1.00 倍)	9.11 (1.72 倍)	9.06 (1.73 倍)	5.77 (2.77 倍)
1 バッチの処理時間 [s]	0.34	0.20	0.20	0.13

4　マルチ GPU 上での DCGAN の性能評価

本稿では，猫のカラー画像 2944 枚をトレーニング画像としてバッチサイズを 64 とし，300 エポックで評価を行う．CuPy，PyCUDA，CUDA 実行 (1GPU と 2GPU) において 1 エポックごとの処理時間，1 バッチの平均処理時間，速度向上比を確認する．オリジナルモデルの学習の精度を見るために元のモデルと比較した損失の推移の評価も行う．

入力は，100 要素の標準正規分布をノイズとし毎回ランダムに生成している．生成器は 5 層から構成されており，転置畳み込み層，バッチ正規化，ReLU の三つを繰り返し，最後に Tanh で出力する．識別器も 5 層から構成されており，畳み込み層，バッチ正規化，Leaky ReLU の三つを繰り返し， 最後にシグモイド関数で出力する．損失関数には，バイナリークロスエントロピーを使用し，最適化手法には Adam を使用している．

なお，本性能評価に用いるマシン環境を表 1 に示す．

4.1　CUDA 実装の DCGAN の性能評価

提案する CUDA 実装の実行時間は，表 2 に示す通りである．ホストとデバイス間でデータの転送を極力減らすために，必要なデータを先にデバイス上に転送する．訓練画像ファイル読み込みと転送時間を合わせると，0.508[秒] 余分にかかるが 300 エポックの実行では影響が少ない．

1 エポックによる実行時間をみると，2GPU の実行では 5.77[秒] であった．GPU 上で実行するためのライブラリである CuPy と比べて 2.77[倍] の速度向上を得ることができた．先行研究である PyCUDA の実行時間が 9.11[秒] であり，それに比べて 1.58[倍] の速度向上を得ることができた．1GPU での実行から 2GPU での実行では，1.57[倍] の速度向上を得ることができた．1 バッチ (64 枚) の平均学習時間は，0.13[秒] で CUDA，PyCUDA に比べて速くなっていることが分かる．

4.2　DCGAN の損失の評価

損失の推移を見ることで生成器と識別器の学習の度合いを確認することができる．PyCUDA，CUDA における生成器と識別器の損失推移を図 2 に示す．

生成器の損失推移は，図 2(a) に示す通りである．生成器の損失は，本物と判定すれば損失は小さくなる．PyCUDA と同じ程度に損失が小さくなっている．識別器の損失推移は，図 2(b) に示す通りである．識別器の損失は，偽物の画像を判定した損失と本物の画像を判定した損失の合算である．損失が小さいほど偽物と本物の判別ができている．PyCUDA と同じ程度に損失が小さくなっている．

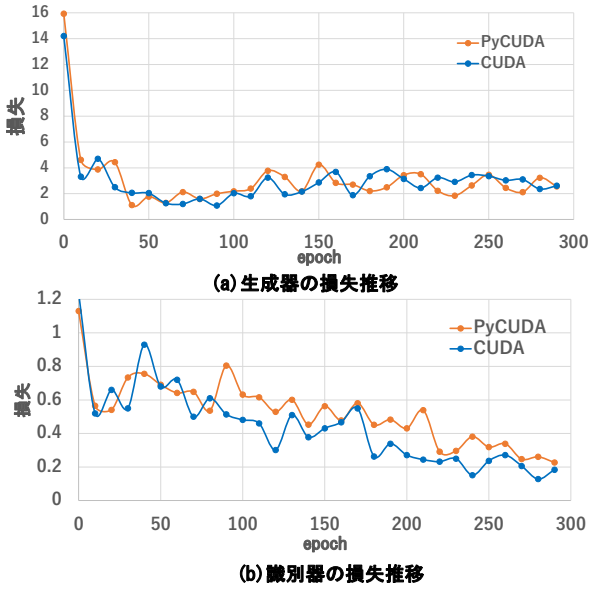

図 2　DCGAN の損失推移．

これらの結果から，PyCUDA 実装と CUDA 実装のモデルに性能の差はなく，提案手法の有効性が確認できる．

5　おわりに

本稿では，マルチ GPU 上での CUDA 実装による深層畳み込み敵対的生成ネットワークの並列処理手法を提案した．DCGAN を OpenMP/CUDA により実装し，RTX6000 搭載 Xeon サーバで性能評価を行ったところ，CuPy 実行と比べて 2.77[倍]，PyCUDA 実行と比べて 1.58[倍] の速度向上を得ることができた．以上の結果から，提案手法の有効性が確認された．

参考文献

[1] Jiawen Liu, Dong Li, Gokcen Kestor, Jeffrey Vetter Runtime Concurrency Control and Operation Scheduling for High Performance Neural Network Training 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019.

[2] Haoyuan Chi, Zebin Zhang, Qiqi Ge, Wenhuan Zhu. Infrared Image Colorization Algorithm Based on DCGAN and Its Edge Device Acceleration, 2022 IEEE International Conference on Civil Aviation Safety and Information Technology (ICCASIT), 2022.

[3] Aswathy Ravikumar, Harini Sriraman Single Node Acceleration of Generative Adversarial Networks using HPC for Image Analytics CIIS '22: Proceedings of the 2022 5th International Conference on Computational Intelligence and Intelligent Systems, Pages 54-59, 2022.

[4] 根本祐輔，吉田明正．DCGAN における PyCUDA 実装による並列処理，情報処理学会第 85 回全国大会，2023.

[5] DCGAN における PyTorch Distributed Data Parallel ライブラリを用いた並列処理，情報処理学会第 84 回全国大会，2022.

[6] pometa0507，DDCGAN-Numpy, https://github.com/pometa0507/DCGAN-Numpy, 2020.