6Q-01

DCGANにおけるPyCUDA実装による並列処理

Parallelization of DCGAN by PyCUDA Implementation

根本　祐輔†　　吉田 明正†
Yusuke Nemoto　　Akimasa Yoshida

1　はじめに

深層学習技術の一つにGANがある．GANはデータの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．DCGANは生成器と識別器に畳み込み層を使用することで通常のGANよりも精度の高い画像を生成することができるが，学習時間が長いことが指摘されている．Pythonプログラムにおいて GPU 計算を行うためには，CuPy や PyCUDA の利用が考えられるが，CuPy は関数ごとに GPU 計算を行っているため性能を最大限に引き出すことが難しい．本手法では PyCUDA を用いて，DCGAN のプログラム全域にわたって CUDA コアおよびデバイスメモリを有効活用することにより，高速化を実現する．NVIDIA Quadro RTX6000 上で 2944 枚のカラー画像を用いて画像生成を行った結果，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，GAN 並びに DCGAN(Deep Convolutional GAN) について述べる [1].

2.1　GAN

敵対的生成ネットワークは，GAN(Generative adversarial networks) と呼ばれる．生成器 (Generator) と識別器 (Discriminator) の二つのネットワークから構成され，このネットワークが互いに学習していく構造になっている．生成器では，与えられたランダムなノイズを基に本物に近い偽物の画像を生成する．識別器では，本物の画像または生成された偽物の画像を入力として，真偽を判定する．識別器で与えられた真偽の誤差を基にして，生成器と識別器のパラメータを更新する．生成器は識別器を本物画像と判定させるために，識別器は本物画像を本物と偽物画像を偽物と判定できるように競い合うように学習する．GAN を高速化させるために，畳み込み計算を近似する方法 [2] や DCGAN のモデル軽量化 [3] が提案されている．

2.2　DCGAN

DCGAN は，二つのネットワークの中間層に畳み込み層を使用している．従来の GAN では，中間層で全結合層を使用してきたが，ノイズから生成される偽物画像の精度が悪いという欠点がある．そこで中間層に畳み込み層や転置畳み込み層を使用することで，学習の精度が向上し欠点を一つ解決できる．しかし，層が多層化になったことなどから処理が重くなり学習時間が長くなるという問題が発生する．本稿では，PyCUDA を利用することで学習の高速化を実現する．

†明治大学総合数理学部ネットワークデザイン学科
Department of Network Design, School of Interdisciplinary Mathematical Sciences, Meiji University

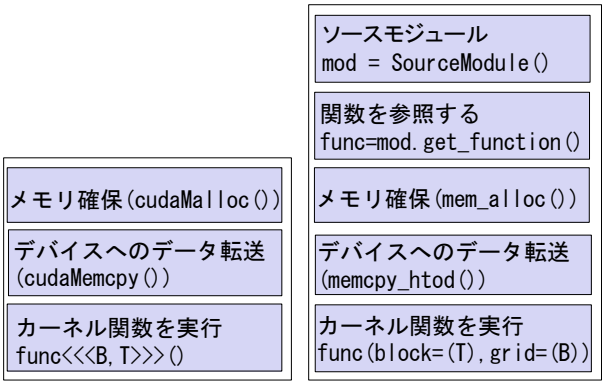


図 1　CUDA と PyCUDA のホストコード.

3　GPU 上での Python プログラムの実行

GPU 向けの汎用並列コンピューティングプラットフォームに CUDA がある．CUDA とは，GPGPU を目的とした C 言語で使われるフレームワークである．C 言語で CUDA を使用する場合のホストコードの手順を図 1(a) で示す．ここで，B はブロック数，T はスレッド数を表す.

3.1　CuPy

CuPy は，NumPy と互換性がある GPU 上で実行できるライブラリである．しかし，CuPy は GPU 上で実行できるにも関わらずブロックとスレッド数を適切に定義することができないため，処理速度が遅くなってしまうことがある．

3.2　PyCUDA

PyCUDA は，CUDA のコードを Python コードから呼び出すことができるが，ホストコードでの関数コールのコードに違いがある．PyCUDA 実行でのホストコードを図 1(b) に示す．ここで，.cu ファイルを Python コードから呼び出すために，ソースモジュールの定義と関数の参照を行う．

4　DCGAN の GPU 上での並列化

本章では，DCGAN における NumPy，CuPy，PyCUDA の 3 つの GPU 実装について述べる．

4.1　NumPy 実装

DCGAN に含まれる転置畳み込み層や畳み込み層などのすべてを NumPy で記述する [4].

4.2　CuPy 実装

DCGAN で使われている NumPy ライブラリを．CuPy に変更して実装を行う．

表 1　性能評価に用いるマシン.

CPU	Intel(R) Xeon(R) 2265 3.50GHz 12Cores
GPU	NVDIA Quadro RTX 6000 24GB × 2
メモリ	DDR4-2933 REG ECC 32GB × 4(128GB)
OS	Ubuntu18.04LTS
Python	3.8.8
CUDA	10.2.89

表 2　RTX 6000 上での DCGAN の実行時間.

	NumPy	CuPy	PyCUDA
1 エポックの処理時間 [s]	101.63 (1.00 倍)	15.69 (6.48 倍)	9.11 (11.15 倍)
1 バッチの処理時間 [s]	2.21	0.34	0.20

4.3　PyCUDA 実装

DCGAN の畳み込み層などの層に含まれる計算部分をカーネル関数で記述する．変数の設定やランダムノイズの生成は Python コードで行う．カーネル間の変数の値は反映されるためカーネル間で同期をしなくても正常に動作する．

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できれば，二つの関数を一つのカーネルで記述する．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．転置畳み込み層と畳み込み層は処理が重いため，複数のカーネル関数を使って，それぞれを効率的に GPU を利用することにより高速化が実現できる．

5　NVDIA RTX6000 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．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，CuPy と PyCUDA の学習の精度を見るために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て毎回ランダムに生成している．生成器は 5 層あり，転置畳み込み層，正規化，ReLU の三つを繰り返し，最後に Tanh で出力する．識別器は 5 層あり，畳み込み層，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．性能評価に用いるマシンは，表 1 に示す通りである．

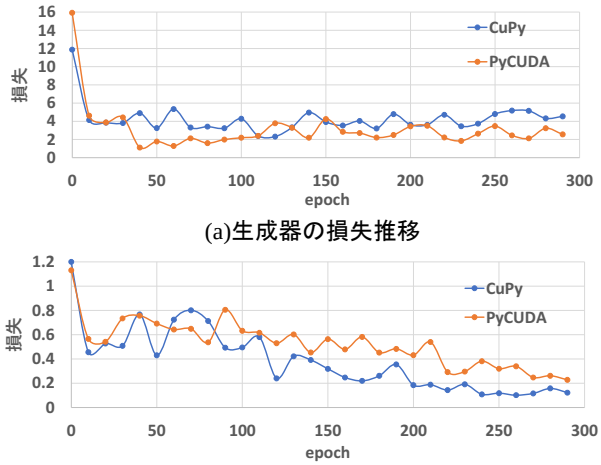
5.1　PyCUDA 実装の DCGAN の性能評価

CuPy，NumPy，PyCUDA 実行時間は，表 2 に示す通りである．PyCUDA では使用する変数を開始時に一度だけに GPU に転送するため，0.463[秒] の転送時間が余分にかかるが，300 エポックの実行では影響が小さい．

1 エポックによる実行で PyCUDA による実行時間は，9.11[秒] であった．NumPy による実行は 101.63[秒] であり，それに比べて 11.15[倍] の速度向上を得ることができた．CuPy 実行が 15.69[秒] であり，それに比べて 1.72[倍] の速度向上を得ることができた．また，1 バッチの平均処理時間を見ても速度が向上していることが分かる．

5.2　DCGAN の損失関数の評価

生成器の損失は，本物と判定してほしいため本物と判定すれば損失は小さくなる．図 2(a) に示す通り，CuPy も PyCUDA も誤差はあるものの同じように損失が小さくなっている．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．偽物が 0，本物が 1 と判定するため間違えると損失が大きくなる．図 2(b) に示す通り，CuPy も PyCUDA も誤差はあるものの同じように損失が小さくなっている．



(a)生成器の損失推移

(b)識別器の損失推移

図 2　DCGAN の損失関数の推移.

損失の推移を見ることで生成された画像の精度や偽物画像生成が成功していることが分かる．また，この二つ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．

6　おわりに

本稿では，DCGAN における PyCUDA 実行による高速化を提案した．DCGAN を PyCUDA 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，NumPy 実行と比べて 11.15[倍]，CuPy 実行と比べて 1.72[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] 我妻幸長．はじめてのディープランニング 2，SB クリエイティブ，2020.

[2] Arman Roohi, Shadi Sheikhfaal, Shaahin Angizi, Deliang Fan, Ronald F DeMara. ApGAN: Approximate GAN for Robust Low Energy Learning From Imprecise Components, IEEE Transactions on Computers (Volume: 69, Issue: 3, 01 March 2020).

[3] 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.

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

[5] 富田雄大，吉田明正．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] t-tetsuya.　Python で始める CUDA プログラミング，Kindle 版，2019.