B-018

GPU を用いた機械学習計算システムの高速化
Accelerating machine learning computation systems using GPUs

住本 貴明[1)]　　増田 信之[1)]
Takaaki Sumimoto　Nobuyuki Masuda

1　まえがき

近年，コンピュータの性能向上に伴い，大規模かつ高精度な数値計算の需要が高まっている．このような計算を行う際，近年ではグラフィック処理用の半導体チップである GPU を数値計算に応用する GPGPU(GeneralPurpose Computing on GPU) という技術が広く用いられている．GPU は CPU と比較して数百倍のコアを有するため，これらを用いて並列計算を行うことで CPU よりも高速に数値計算を行うことができる．数値計算の活用が期待されている状況は様々であり，その中でも近年注力されている機械学習の一種であるディープラーニング及びニューラルネットワークは CPU のみで構築をおこなうと処理が長くなるという課題点があるため，GPGPU 用いた処理での高速化が期待されている．

本研究では，NVIDIA 社が提供する開発環境である CUDA を用いてニューラルネットワークの訓練部分の高速化を行い，処理時間の比較・検討を行った．なお，ニューラルネットワークの訓練として用いるデータベースは MNIST データベースを利用し 60000 枚の画像を基に推論を行うニューラルネットワークの構築をおこなった．

2　CUDA

2.1　プログラミングモデル

CUDA プログラミングの流れを図 1 に示す．CUDA のプログラムはホスト (CPU) 側で動作するプログラムとデバイス (GPU) 側で動作させるプログラム (カーネル関数) にそれぞれ分かれており，実行する際にはホスト側デバイス側に必要なメモリを確保し，計算を行う際にホスト側からデバイス側にデータを転送し処理を行い，結果をホスト側に戻すというアルゴリズムをとっている．並列処理を行う際には CPU 側でカーネル関数を実行する際にスレッド数，ブロック数を指定しカーネル関数を起動することでデバイス側でどの程度の並列計算を行うのかを決定することができる．そのため，開発者は逐次的にコードを記述するだけで GPU プログラミングを行うことができる．

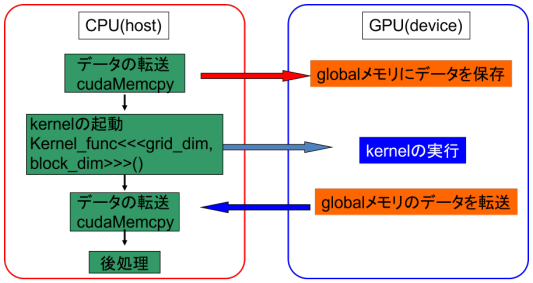

図 1　CUDA プログラミングモデル

2.2　スレッドの構成

図 2 に CUDA のスレッド構造を示す．ホスト側からカーネルを実行すると，デバイス側で指定されたスレッド数，ブロック数が生成され，カーネル関数に従い処理が並列におこなわれる．
グリッド内のスレッドはすべて同じグローバルメモリ空間を共有する．また，ブロックは互いに協調して動作できるスレッドのグループであり，ブロックに属するスレッド間の同期や，ブロック内で共有されるメモリ (シェアードメモリ) 等の機能を利用できる．[1]

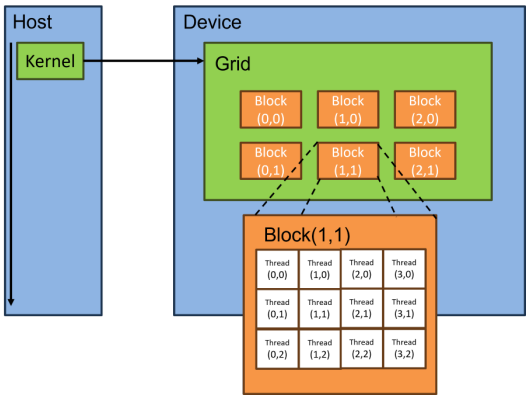

図 2　CUDA のスレッド構造

3　ディープラーニング

ディープラーニングはニューラルネットワークの構造を拡張したものであり，脳内の神経細胞 (ニューロン) のネットワーク構造を模した数学モデルである．ディープラーニングの構造を図 3 に示す．構造として入力層，隠れ層，出力層の 3 つに分かれ，相互に接続し推論を行う．

この隠れ層にあたる部分を多層的に展開することでディープラーニングはデータに含まれる特徴を段階的により深く学習することが可能になる．

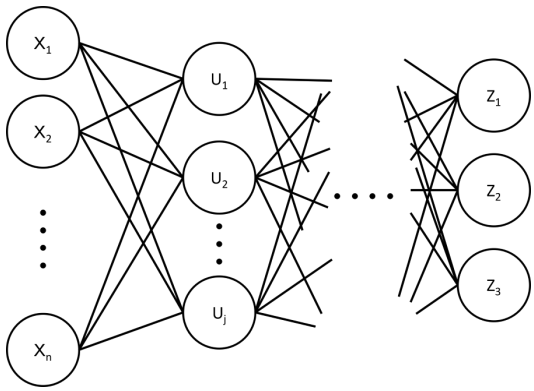

図 3　ディープラーニング構造

1)　東京理科大学　先進工学研究科

3.1 順伝播

入力層から出力層への方向を順方向として入力と変数を掛け合わせ予測値を計算する手法が順伝播である．隣り合う 2 層において，前の層のノードが n 個の時の j 番目の出力を考える．入力を x，次の層を u，各ノードにかける重み，バイアスをそれぞれ w，b とすると順伝播の式 (1) は以下のようにあらわされる．[2]

$$u_j = \sum_{i=0}^{n} w_{ji} x_i + b_j \tag{1}$$

得られた値を活性化関数を用いて 0，1 間の値で表せるように出力を変換する．活性化関数は式 (2) で表されるシグモイド関数を用いた．

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

3.2 誤差逆伝播

得られた出力と正解ラベルとの誤差を求めその誤差を微分の連鎖律に基づいて出力側から入力側に逆伝播することで勾配を計算し，重みやバイアスを更新する手法である．

入力に対する二乗誤差の合計は E で表され，損失関数と呼ばれる．この損失関数が最小になるように重みとバイアスを更新し，学習の精度を高めていく．[2] 学習率を α，出力を y, 損失関数である二乗誤差を E とすると重み w，バイアス b は以下のように更新される．

$$w_{new} = w_{old} - \alpha \frac{\partial E}{\partial w_{old}} \tag{3}$$

$$b_{new} = b_{old} - \alpha \frac{\partial E}{\partial b_{old}} \tag{4}$$

4 GPU による高速化

4.1 MNIST データベース

本研究では，ニューラルネットワークの構築を高速化させるにあたって MNIST データベースを使用した．MNIST(Modified National Institute of Standards and Technology) データベースとは，米国商務省配下の研究所が構築した手書き数字画像の大規模なデータベースである．[3] このデータベースには 28×28 ピクセルのグレースケール画像が訓練用に 60000 枚，評価用に 10000 枚用意されている．これらの画像は 0 から 9 までの手書き数字であり，各画像につき 1 つの数字が書かれている．

4.2 プログラム実装概要

本研究では，ニューラルネットワークを用いて手書き数字画像を分類するプログラムを実装する．訓練用の画像 60000 枚を用いてニューラルネットワークの構築をおこなった．その際，最も処理時間のかかる順伝播と逆伝播の計算処理を GPU 上でおこない，その処理時間を CPU のみで行った場合，GPU のブロック，スレッドを変更した場合をそれぞれ計測，比較を行うことにした．また，出力は 10 とし，そのうちの該当する番号にのみ 1 が出力される One-Hot エンコーディングを使用した．

4.3 実装環境

本シミュレーションで使用した GPU 搭載 PC の実装環境を表 1 に示す．

表 1 実装環境

OS	Ubuntu20.04
CPU	Intel Core i7-11700K 3.60GHz
RAM	31GB
GPU	NVIDIA GeForce RTX3070
CUDA	10.1

測定に用いたプログラムは CUDA を用いて順伝播と逆伝播の部分を GPU 上に実装したものと，C++で実装したもの用意し，それぞれ実行した場合について測定を行う．

4.4 評価

画像の各ピクセルを 1 つの入力とし，入力層のノードを 784，隠れ層を 2 つ用意し，それぞれのノードを 100,50，出力層のノードを 10 とし，60000 枚の手書き数字データを訓練に用いて分類をおこなうニューラルネットワークを構築し，その処理時間の計測をおこなう．
また，構築したニューラルネットワークの精度を確認するため，評価用の 10000 枚の手書き数字画像を入力とし順伝播を行い，その正答率を計算する．

C++を用いて CPU 上で構築したニューラルネットワークでは，その正答率は 68%ほどであった．処理時間を 10 回計測し平均をとったところ，24.756 秒であった．CUDA を用いて GPU 上で構築したニューラルネットワークの処理時間や正答率とそれぞれの比較の詳細は発表時に詳細な結果を述べる．

5 まとめと今後の課題

本項では，ニューラルネットワークの構築における課題点である処理時間に着目し，その処理時間を CPU のみで実装した場合と GPU 上で処理時間が最もかかる順伝播と逆伝播の計算を行い実装した場合の処理時間を比較し検討を行った．

CUDA で実装した場合，計算を行う処理は高速化される．しかし，計算を行うために必要なデータのコピー部分の処理時間がボトルネックであることが予想される．そのため，データコピーの回数を少なくすることや，シェアードメモリなどを活用することでボトルネックとなっている処理の負担を軽減することが期待できる．

今後は CUDA 上で実装したプログラムを最新の GPU 機種である RTX4090 上に実装し，その処理時間や消費電力などの計測をおこない，高速化の検討を行う予定である．また，今回のプログラムにおいて，計算処理は float 型である単精度浮動小数点を用いて実装を行ったが，倍精度浮動小数点などほかの型での精度や実装時間の計測をし，その違いの検討を行う事も予定している．

参考文献

[1] John Cheng, Max Grossman, Ty McKercher, ”CUDA C プロフェッショナルプログラミング”，株式会社インプレス，p36,2018 年

[2] 藤田毅，滝口直樹，”C++で学ぶディープラーニング”，株式会社マイナビ出版，p51,p054，2017 年

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ”Gradient-based learning appliedto document recognition”, Proceedings of the IEEE, November 1998, vol.86 no.11,p.2278-2324