情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 3　88–99 (May 2011)

手書き文字認識向けニューラルネットワークモデルのGPUによる高速化

池田孝利[†1]　宮本弘之[†2]
伊野文彦[†1]　萩原兼一[†1]

ネオコグニトロンは手書き文字認識を行うニューラルネットモデルである．変形や位置ずれに強い認識が可能だが，計算コストが高く，高速化が求められている．GPUを用いた並列計算により高速化が期待できる反面，その計算には3次元のコンボリューションが含まれており，多くのメモリアクセスを要するといった問題がある．本研究では，ネオコグニトロンの処理時間の95%以上を占めるS層応答計算をGPUを用いて高速化し，性能を検証した．グローバルメモリへの書き込みを最小化し，共有メモリの利用効率を高めた高速化では，CPU実装に対して約8〜20倍高速化した．

GPU Acceleration of Neural Network Model for Handwritten Digit Recognition

Takatoshi Ikeda,[†1] Hiroyuki Miyamoto,[†2]
Fumihiko Ino[†1] and Kenichi Hagihara[†1]

Neocognitoron is a neural network model for handwritten digit recognition. Although Neocognitoron is robust to shifts and deformations but it asks very high calculation cost, therefore an improvement in the speed is required. While the improvement in the speed is expectable by parallel computing, there is a problem which calculation of 3D convolution requires many memory accesses. In this research, we accelerate the S layer response calculation of Neocognitoron using the GPU, which occupies more than 95% of processing time. We also found that the speedup over a CPU implementation ranges from a factor of 8 to that of 20 by minimizing write to global memory and by increasing the efficiency of shared memory.

1. はじめに

ネオコグニトロンは，手書き文字認識のための階層型ニューラルネットワークモデルである[1)]．認識対象とする手書き文字の変形や位置のずれに強く，高い認識性能を持つが，細胞ごとの応答計算コストが高く，ネットワークの規模も大きいため，計算コストが高い．ネオコグニトロンのより幅広い応用を可能とするためにいっそうの高速化が求められている．近年，57×57 画素の画像を約100ミリ秒で認識する成果[2)]があるが，たとえば，カメラからの画像に対する実時間文字認識[3)]に応用するには，一般的なビデオカメラの速度である毎秒30フレームの入力画像に対して，十分な速度が達成されていない．パイプライン処理などにより文字認識処理以外の実行時間が文字認識の実行時間に対して隠蔽可能であるとき，1文字あたりの計算時間を4.1ミリ秒以下にできると，電話番号，郵便番号など数字8桁程度を実時間認識する用途に応用できる可能性がある．

一方，急速に性能向上するGPU（Graphics Processing Unit）を多様な問題の高速化に用いる研究が多くの成果をあげている[4)–7)]．ネオコグニトロンの処理時間の大部分は，S層応答計算が占めており，高い並列演算性能を持つGPUを用いることにより高速化が期待できる．しかし，S層応答計算は，多くの3次元畳み込み計算を含み，大きなレイテンシをともなう多くのメモリアクセスが必要となる．この多数のメモリアクセスがGPU実装の性能を低下させてしまうという問題がある．畳み込み計算をアトミック浮動小数点演算が使えないPE（Processing Element）により並列計算する場合，メモリに対する排他的な加算ができないため，並列化手法によっては積算時に追加のコードおよびメモリアクセスが必要になる．GT200世代以前のGPUは，アトミック浮動小数点演算が使えないため，この点が問題になる．

本研究では，ネオコグニトロンのより幅広い応用を可能とするため，GPUを用いてネオコグニトロンのS層応答計算を高速化した．GPUを用いてネオコグニトロンのS層応答計算を高速化し，毎秒30フレームの入力画像に対し数字8桁の認識速度である1文字あたりの計算時間4.1ミリ秒以下を目指した．S層応答計算が含む3次元畳み込み計算は，他の

†1 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University
†2 九州工業大学大学院生命体工学研究科脳情報専攻
Department of Brain Science and Engineering, Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology

階層型ニューラルネットワークをはじめ多くのアプリケーションで使われている．S 層応答計算の高速化はこれらのアプリケーションの高速化にも有用である．GPU は GT200 世代を対象とし，アトミック浮動小数点演算が使えないことによる問題に対処し，グローバルメモリへの書き込み回数を最小化できるスレッド設計，興奮性細胞から S 層への入力に関する 3 次元畳み込み計算と抑制性細胞の応答計算のための畳み込み計算を同時に計算することで大きなレイテンシをともなうグローバルメモリへのアクセスを削減するブロック設計，およびグローバルメモリからの読み込み回数を削減するため共有メモリの利用および効率を高める手法により高速化，その性能を 2 種類の CPU 実装と比較，評価した．

以降では，まず 2 章で関連研究について紹介し，それらと本研究との比較をする．次に 3 章でネオコグニトロンについて述べ，4 章で GPU プログラミング環境 CUDA（Compute Unified Device Architecture）[8] について述べる．その後，5 章でネオコグニトロンの GPU による高速化について述べ，6 章で性能を評価し，7 章で本稿をまとめる．

2. 関連研究および本研究との比較

本章では，関連研究を紹介し，それらと本研究との比較をする．ネオコグニトロンの高い並列度に着目し，並列計算機に実装し高速化した研究および，類似した研究には以下の W_1〜W_5 がある．また，それぞれの実装と本研究との比較を**表 1** に示す．

W_1　伊藤ら[9] によるネオコグニトロンの NCUBE への実装

表 1　関連研究の並列実装との比較
Table 1　Comparison with parallel implementation of related work.

実装	認識対象（画像サイズ）	実装ハードウェア（認識速度）	分割単位	実行単位
W_1	文字（19 × 19）	NCUBE（8 文字/秒）	細胞面	PE
W_2	画像（17 × 17）	トランスピュータ（3 秒/文字）	画像の 2 次元分割	PE
W_3	動画像（27 × 20）	FPGA（毎秒 9 フレーム）	細胞（S 細胞，C 細胞）	デジタル回路
W_4	画像（57 × 57）	GPU（118 ミリ秒）	細胞面 細胞	ブロック スレッド
W_5	文字（29 × 29）	GPU（1.6 ミリ秒）	機能地図 細胞	ブロック スレッド
本研究	文字（65 × 65）	GPU（3.4 ミリ秒）	細胞面串刺し細胞群 細胞	ブロック スレッド

NCUBE は，CPU と同程度の計算資源をローカルに持つ独立した 512 個の MIMD PE が，ハイパーキューブ構造のネットワークで接続された疎結合並列計算機である．ネオコグニトロンについては，3 章で述べるが，各機能を担う細胞層が多段に接続された構成である．細胞層は複数の細胞面から構成され，これらの細胞面を PE へ分割している．1 細胞面の計算を 1PE で実行し，1 文字あたり 38PE を使用する実装である．この実装および後述する W_2 はとともに MIMD 並列計算機への実装である．これに対して，本研究では，実装対象である GPU の PE が SIMD に近いため，粒度の小さな単位で分割実装している点で異なる．

W_2　大森ら[10] のトランスピュータへの実装
4 つの MIMD PE（トランスピュータ）から構成されるシステムへの実装である．PE 間の通信量に着目し，処理対象画像を MIMD PE に分割する実装である．W_1 と同様，通信リンクを介した MIMD PE への実装であるが，W_1 とはデータ分割の方法が異なり，画像を 2 次元分割し，4PE で処理している．

W_3　木村ら[11] によるネオコグニトロン型動画像識別モデルのハードウェア実装
ネオコグニトロンを拡張し，動画像に対応したモデルを FPGA に実装し高速化する研究である．S 細胞，C 細胞をそれぞれデジタル回路で実現したことにより，それぞれ 1 細胞あたり 1.79 マイクロ秒，1.67 マイクロ秒で処理し，毎秒 9 フレームの実時間処理に必要な速度である，U_{S1} 層の処理時間 12.6 ミリ秒を実現している．

W_4　Poli ら[2] によるネオコグニトロンの GPU への実装
顔画像認識のための 3 ステージ（6 層）のネオコグニトロンを CUDA を用いて GPU に実装し高速化している．CUDA については，4 章で述べるが，ネオコグニトロンの細胞面，細胞をそれぞれ CUDA のブロック，スレッドに対応させた実装である．この研究のように，画像を認識対象とした場合には，計算が省略できる細胞の比率が多くならないため問題とならないが，本研究のように文字を認識対象とした場合には，計算を省略できる細胞の比率が高くなり，CPU 実装と比較して高速化な実装が難しくなるという問題がある．この詳細については，3.3 節で述べる．

W_5　Billconan ら[12] によるニューラルネットワークの GPU 実装
この研究は，“A Neural Network on GPU” として “THE CODE PROJECT に” 公開されている．本研究と同じく手書き文字認識を目的とするため，ニューラルネットワークの構造には類似点がある．ニューラルネットワークの機能地図，細胞をそれぞれ CUDA のブロック，スレッドに割り当てている点で，W_4 と同じ分割手法であ

り，本研究では異なった分割手法を用いる．5 階層構造ニューラルネットワークを使用しており，本研究の 10 層ネオコグニトロンに対して大幅に小規模である．入出力を含む実行時間は，CPU の実行時間 16 ミリ秒の 10 倍高速としているため，表 1 では 1.6 ミリ秒とした．SIMD に似た GPU の性能を発揮しやすくするために，ニューラルネットワークの構造を単純にし，メモリアクセスや計算を単純化している．文字を認識対象としているものの，計算を省略できる細胞の判定は実装されておらず，すべての細胞について計算を実行する実装になっている点でも本研究とは異なる．

本研究で対象としたネオコグニトロンは，高い認識性能のための複雑なネットワーク構造を持ち，文字を認識対象とする際に多く起こる不要な計算を省略することで高速化するための条件分岐を多く含む．GPU への実装の研究 W_4，W_5 では，このように条件分岐を多く含んだ実装の研究がなされていない．本研究では，高い認識性能のための複雑なネットワーク構造を簡略化せず，従来研究で問題になっていないメモリアクセスに関する高速化手法を適用するため，従来研究とは異なる分割手法により高速化を達成することを目指した．

3. ネオコグニトロン

ネオコグニトロンは，Fukushima[1] により提案された，パターン認識のためのニューラルネットモデルである．特に位置ずれやひずみをともなう手書き文字に対して高い認識率を発揮する．

3.1 ネオコグニトロンのネットワーク構造

ネオコグニトロンのネットワーク構成を**図 1** に示す．図 1 に示すように，神経細胞を模倣する細胞を 2 次元に配置した細胞面（cell-plane），複数の細胞面からなる細胞層（cell-layer）が階層化した構造である．細胞には興奮性細胞と抑制性細胞があり，図 1 は興奮性細胞のみを図示した．興奮性細胞（Excitatory cells）と抑制性 V 細胞（Inhibitory V-cells）を含む U_{Cl-1} 層と U_{Sl}（$1 \leq l \leq 4$）層の細胞の接続を**図 2** に示す．細胞層はそれぞれコントラスト抽出（contrast extraction），輪郭抽出（edge extraction）など特徴抽出に特化した機能を担う．図中の U_0，U_{S1}〜U_{S4}，U_{C0}（U_G）〜U_{C4} はそれぞれ細胞層であり，認識対象となる手書き文字画像は入力層（input layer）U_0 から入力される．細胞の応答は，各層間のネットワークを経て伝わり，最後に認識層（recognition layer）U_{C4} に認識結果を提示する．それぞれの細胞面中の細胞は前層の（層ごとに異なる）半径 A_S の範囲にある細胞の応答を受け取り集積する．また，U_{Sl}，U_{Cl}（$1 \leq l \leq 4$）は対になっており，この対をそれぞれステージ Sl（$1 \leq l \leq 4$）と呼ぶ．U_{Cl} 層は，直前の U_{Sl} 層の結果をぼかす機能を持つ．

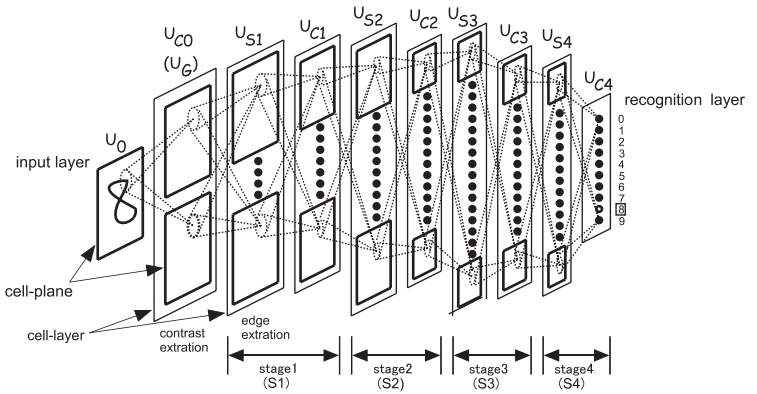

図 1　ネオコグニトロンのネットワーク構成
Fig. 1　Structure of neocognitron network.

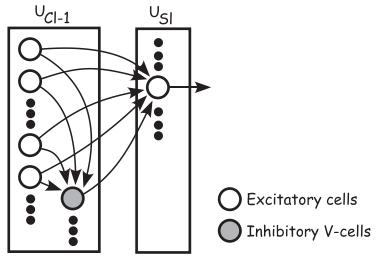

図 2　興奮性細胞と抑制性細胞（V 細胞）の接続
Fig. 2　Connections of excitatory cells and inhibitory V-cells.

これにより，細胞面が 2 次元の等方性を持つことと合わせ，文字の位置のずれや変形に強い認識を可能としている．

表 2 にそれぞれの細胞層の構成および各細胞の前層細胞との接続数（$4{A_S}^2$）を示す．表は興奮性細胞のみ示す．入力接続数は細胞面の周辺部ではより小さな値となるが，細胞面の中央付近の値で示す．表中の一部の細胞構成は学習段階で自動的に決定されるものである．ネットワークの前方（S1）では細胞面は広く面数は少ない，後方（S4）ほど細胞面は狭く面数は多いという特徴がある．

表 2　細胞層の細胞構成
Table 2　Size of each cell-layer.

細胞層	細胞構成	入力接続数
U_0	65×65	
U_{C0}（U_G）	$71 \times 71 \times 2$	7×7
U_{S1}	$68 \times 68 \times 16$	6×6
U_{C1}	$37 \times 37 \times 16$	6×6
U_{S2}	$38 \times 38 \times 26$	6×6
U_{C2}	$21 \times 21 \times 26$	6×6
U_{S3}	$22 \times 22 \times 78$	6×6
U_{C3}	$13 \times 13 \times 78$	8×8
U_{S4}	$5 \times 5 \times 70$	9×9
U_{C4}	$1 \times 1 \times 10$	5×5

3.2　S 層応答計算

ネオコグニトロンにおいて処理時間の大部分を占めるのは，U_{S1}〜U_{S4}（S 層）の細胞に対する応答計算である．それぞれの S 層は表 2 に示すように，細胞構成が大きく異なる．

ステージ $S1$〜$S4$ の細胞層 U_{Sl}，k 番目の細胞面，位置 $\boldsymbol{n}$ の細胞 $u_{Sl}(\boldsymbol{n}, k)$ への応答を次式に示す．

$$u_{Sl}(\boldsymbol{n}, k) = \frac{\theta_l}{1-\theta_l}\varphi\left[\frac{1+\sum_{\kappa=1}^{K_{Cl-1}}\sum_{|\nu|<A_{Sl}} a_{Sl}(\nu,\kappa,k)u_{Cl-1}(\boldsymbol{n}+\nu,\kappa)}{1+\theta_l b_{Sl}(k)v_l(\boldsymbol{n})}\right]-1, \quad (1)$$

$$v_l(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}}\sum_{|\nu|<A_{Sl}} c_{Sl}(\nu)\{u_{Cl-1}(\boldsymbol{n}+\nu,\kappa)\}^2}, \quad (2)$$

$$\varphi(x) = \begin{cases} x \ (x \geq 0) \\ 0 \ (x < 0)\ . \end{cases} \quad (3)$$

ここで，u_{Cl-1} は前層 U_{Cl-1} からの入力細胞，K_{Cl-1} は前層の細胞面数，a，b，c，θ は係数配列および係数である．式 (1) は，u_{Sl} が多くの興奮性細胞 u_{Cl-1}，および抑制性細胞 v_l からの入力に応答することを表している（図 2）．式 (2) は，抑制性細胞 v_l が多くの興奮性細胞 u_{Cl-1} からの入力に応答していることを表している．

また，$u_{Sl}(\boldsymbol{n}, k)$ は，3 次元畳み込み $v_l(\boldsymbol{n})$ および前層からの入力細胞 u_{Cl-1} と係数配列 a_{Sl} の 3 次元畳み込みの商の非線形関数であることが示されている．この計算のため，ネオコグニトロンの S 層応答計算は計算量が多く，また多くのメモリアクセスを必要とする．それらにより，GPU の処理速度が低下してしまう可能性がある．

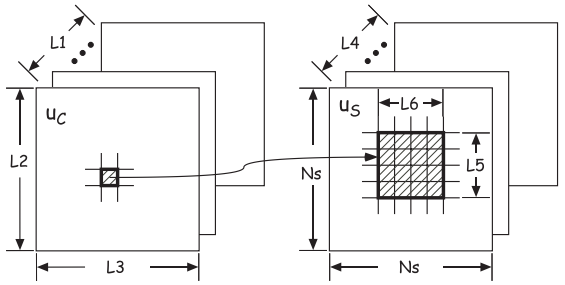

図 3　CPU 実装のスキャッタ構造
Fig. 3　Scatter structure of CPU implementation F.

3.3　S 層応答計算における C 層細胞の影響

式 (1) では，項 $u_{Cl-1}(\boldsymbol{n}+\nu,\kappa)$ が 0 のとき，その積も 0 である．したがって，$u_{Sl}(\boldsymbol{n}, k)$ の応答計算では，同細胞に接続された入力細胞であっても値が 0 の細胞との積和計算は省略できる．その場合，計算量を削減できる．そのためにはそれぞれの u_{Cl-1} に対して条件分岐が必要となる．CPU では浮動小数点演算と比べ条件分岐を高速に実行できるため高速化できる可能性がある．しかし，GPU ではアーキテクチャの制限により並列実行がストールするため，逆に性能が低下してしまう可能性がある．

3.4　ネオコグニトロンの CPU 実装

本研究では，まず，Fukushima の実装[13] をもとに CPU 実装を作成した．この実装は，入力層から出力層へと伝達の流れに従って応答計算をする．

CPU 実装では，3.2 節で述べた S 層応答計算手続きが，認識処理の実行時間の 95%を占めるホットスポットであり，それぞれステージ S1〜S4 の応答計算のため 4 回呼び出される．

CPU 実装の S 層応答計算手続きでは，最初に式 (2) により，5 重ループでステージ l のすべての抑制性細胞 v_l の応答計算を行う．次に 3 重ループ（L1，L2，L3）により 3 次元の U_C 層の興奮性入力細胞から 1 つの細胞 $u_C(kc, ic, jc)$ を選び，次にその細胞が接続する複数の出力細胞から 3 重ループ（L4，L5，L6）により次々に細胞 $u_S(ks, is, js)$ を選ぶ，そのつど，係数との積和を計算し $u_S(ks, is, js)$ へ積算する構造（スキャッタ構造）である（**図 3**）．このため，1 つの $u_S(ks, is, js)$ に対して，読み出し，加算，書き込みの一連の操作が表 2 に示した入力接続数回繰り返される．3.3 節で述べた計算量の削減は，L1，L2，L3

により入力細胞 $u_C(kc, ic, jc)$ を選んだ後，条件判断を行う．$u_C(kc, ic, jc)$ が 0 のとき，内側のループ（L4，L5，L6）は実行しない．

4. GPU プログラミング環境 CUDA

本章では，本研究で使用した，GPU プログラミング環境 CUDA [8] について述べる．CUDA を使用したプログラミングについては，CUDA Programming Guide [14] などのドキュメントに詳細に解説されている．また，GPU メーカのサイト [8),15)] では，多くの情報を得ることができる．CUDA を使用した開発では，3D グラフィクスに関する知識は必須ではないが，GPU の性能を引き出すためには，依然として GPU のアーキテクチャに関する深い理解が必要である．

4.1 CUDA における GPU 並列プログラミング

CUDA では，GPU をシングル命令マルチスレッド（SIMT: Single Instruction Multi-Thread）プロセッサとして扱う．スレッド（thread）は，32 スレッドを 1 単位（1 warp と呼ぶ [14)]）として実行を制御され，最大 512 個までのスレッド・グループを 1 ブロック（block），ブロック・グループをグリッド（grid）とする論理階層構造である．本研究で使用した GPU GTX 280 は，Streaming Processor（SP）を 8 個搭載した 30 個の Streaming Multiprocessor（SM）で構成され，合計 240 個の SP を持つ．各スレッドは SP で実行される．プログラムにおいて使用可能なメモリおよびその特徴を**表 3** に示す．サイズは GTX 280 のものである．

CUDA による GPU 並列プログラミングには，以下のような特徴がある．

F1　グローバルメモリは容量が大きく，全スレッドから読み書きできるが，アクセスには 400〜600 サイクル [16)] のレイテンシがともなう．

F2　グローバルメモリへの複数スレッドからの同時アクセスには，アクセスのパターンによりレイテンシが異なる．Coalesced と呼ばれる [14)] メモリアクセスパターンであれば，最大のスループットとなる．

F3　グローバルメモリは，テクスチャ・マッピング用の機能を持つ，テクスチャメモリとしてアクセスする領域ではキャッシュが働く．

F4　共有メモリは，ブロック内で共有できる小容量のメモリである．レジスタ並みの低いレイテンシでアクセスできる．スレッド間でのバンク・コンフリクトがない場合には，概略グローバルメモリの数百倍小さな [16)] レイテンシである．共有メモリは複数のバンクで構成されていて，同一ワープ内の複数スレッドが同時に同じバンクにアクセスすると，レイテンシをともなうバンク・コンフリクト [14)] が起こる可能性がある．

F5　レジスタは，レイテンシが小さいが，ブロックごとに 16,384 本までしか使うことができない．

表 3　GPU メモリの特徴（GT200）
Table 3　Features of GPU memory (GT200).

メモリ	キャッシュ	GPU からのアクセス	スコープ	サイズ
レジスタ	なし	読み書き	スレッド内	ブロックごとに 16,384 本
共有メモリ	なし	読み書き	ブロック内	SM ごとに 16 KB
グローバルメモリ	なし	読み書き	全スレッド	1 GB
定数メモリ	あり	読み出し	全スレッド	64 KB
テクスチャメモリ	あり	読み出し	全スレッド	

4.2 CUDA のデザインパターン

CUDA で，GPU の性能をより引き出すため，広く知られているデザインパターン [14),16),17)] には次のものがある．

D1　特徴 **F5** により，スレッドあたりのレジスタ使用量が増えると，同時に実行できるスレッド数が減り，ブロックあたりのスレッド数を増やすと，スレッドあたりの使用可能なレジスタ量が減ってしまう．レジスタおよびスレッドの割り当て方によっては，いずれかのリソースが十分に使われない状態となる．したがって，レジスタおよびスレッドの割り当て方を調整し，ハードウェア・リソースの稼働率を上げることで，性能向上の可能性がある．

D2　特徴 **F1** のため，グローバルメモリへのアクセスは最小に抑える．メモリアクセスを減らすのは CPU でも重要である．GPU ではグローバルメモリへのアクセスはレジスタやバンク・コンフリクトのない場合の共有メモリへのアクセスに比べて数百倍低速 [16)] なため，特に重要である．

D3　特徴 **F1**，**F2** のため，グローバルメモリへのアクセスは，Coalesced アクセスとする．

D4　特徴 **F1**，**F4** のため，グローバルメモリから共有メモリへいったんコピーし，共有メモリをアクセスすることにより，性能向上の可能性がある．

5. ネオコグニトロンの GPU による高速化

本章では，ネオコグニトロンの GPU による高速化について述べる．

5.1 ネオコグニトロンの GPU 実装における問題点

本研究では，CPU 実装の処理時間の 95%以上を占める S 層応答計算を GPU を用いた高

速化の対象とした，S 層応答計算の GPU 実装では，以下のような問題点がある．

(1)　スキャッタ構造では計算結果の出力に多くの読み書きが必要．3.4 節で述べように，応答信号の伝達に従いスキャッタ構造とした場合，計算結果の書き込みに積算が必要となり，1 出力細胞に対して入力接続数回の読み，書きを実行することになる．並列スレッドにより並列処理する場合，本研究で使用した GPU のようにアトミック浮動小数点演算が使えない場合，さらに多くの読み書きを用いて積算する必要がある．

(2)　メモリアクセスが多い．3.2 節に示したように S 層応答計算は 3 次元畳み込み計算を含むため，計算量が多いだけでなく多次元配列へのメモリアクセスを多く含む．GPU はメモリアクセスにレイテンシがともない，特にグローバルメモリへのアクセスのレイテンシは大きく，速度低下の原因となる．

(3)　構造が複雑．表 2 および 3.1 節に示したように細胞層の細胞構成，および入力接続数が統一されておらず，入力接続数も細胞面の中央部と周辺部で異なるなど，プログラムが複雑になる要因が多い．このためコードが長くなり高速化効果が小さくなる．

ネオコグニトロンの認識性能の高さは (3) によるものであるため，本研究では，(1) および (2) をメモリアクセスの削減，特にグローバルメモリへの書き込みを最小化した実装により解決，高速化を目指した．以下の節では，適用した高速化手法について述べる．

5.2　グローバルメモリへの書き込みを最小化するギャザ構造スレッド（T1）

3.4 節では，CPU 実装 F が 1 つの入力から複数の出力に伝播するスキャッタ構造であり，U_C の各細胞についての読み出し，加算，書き込みの一連の操作が多数回繰り返されることを示した．デザインパターン **D2** のため，1 つの出力細胞 $u_S(ks, is, js)$ のすべての入力細胞 $u_C(kc, ic, jc)$ の応答積算を計算カーネルとし，1 スレッドで担当するギャザ構造とする．この概要を**図 4** に示す．図では，U_C のサイズは $N_C \cdot Nc \cdot Kc$，U_S のサイズは $Ns \cdot Ns \cdot Ks$，入力接続数は $Na \cdot Na$ として示す．各スレッドは，担当する $u_S(ks, is, js)$ それぞれに対して 1 回だけ書き込む．この，スキャッタ構造からギャザ構造への変換による並列スレッド化で，グローバルメモリへの書き込みを最小化できる．また，複数スレッドから同一のアドレスへの読み書きが起こらないため，本研究で使用した GPU が浮動小数点演算に対応したアトミック命令を持たず，$u_S(ks, is, js)$ の積算を正しく実行することができない点にも対応できる．また，**D3** のため，計算カーネルの最後で同期した後に書き込む．スキャッタ構造では，$u_S(ks, is, js)$ の積算のため $Ks \cdot Ns \cdot Ns \cdot Na \cdot Na$ 回の読み出しと書き込みが必要であるが，ギャザ構造では，$Ks \cdot Ns \cdot Ns$ 回の書き込みのみで済む．これによりグローバルメモリへの書き込みの最小化が可能である．実際のスキャッタ構造の実装では，3.4 節に述べたループ省略のため，$u_S(ks, is, js)$ 積算回数はもっと少なく，逆に初期値の書き込みおよび積算の後の係数演算のためにそれぞれ $Ks \cdot Ns \cdot Ns$ 回多くの書き込みと読み出しが実行される．6 章で述べる実験条件での実測では，$u_S(ks, is, js)$ のためのグローバルメモリの読み出しおよび書き込み回数は，スキャッタ構造の実装では，1 文字の認識あたり平均それぞれ約 200 万回および約 190 万回，ギャザ構造の実装では，0 回および約 14 万回であった．スキャッタ構造からギャザ構造にしたことにより，$u_S(ks, is, js)$ ためのグローバルメモリへのアクセスは約 96%削減できている．

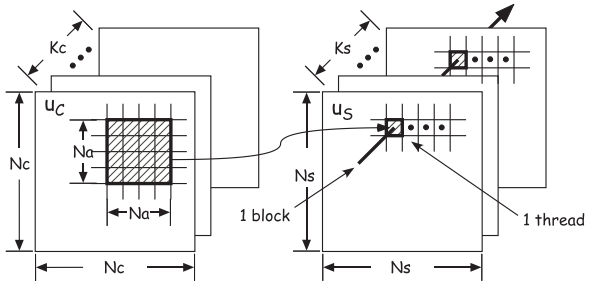

図 4　GPU 実装 T のギャザ構造
Fig. 4　Gather structure of GPU implementation T.

CPU でもスキャッタ型からギャザ型に転換し，高速化する可能性があるが，本研究のような畳み込み計算を行う場合，低速化してしまう可能性がある．ギャザ型の実装では出力変数へのアクセス回数を最小化できるが，逆に入力側変数のアクセス回数は増加する．また，本研究のように，入力側の変数により条件判断する場合，スキャッタ型からギャザ型への転換にともない内側ループと外側ループが逆転し，条件判断を実行する回数が増加するため低速化の要因となる．このため，積算の入力にともなう処理コストに対し，出力変数へのメモリアクセスのコストが十分に大きくないと，出力変数へのメモリアクセス回数削減による高速化効果が打ち消され，低速化してしまう可能性がある．

5.3　グローバルメモリへのアクセスを削減する串刺しブロック構造（T2）

S 層応答計算式 (1) の計算では，u_{Sl} 細胞への入力となる抑制性細胞 v_l の応答計算式 (2) の計算が必要であり，U_S 中の位置 $(\boldsymbol{n}, k)$ の S 層応答計算で必要な抑制性細胞からの入力 $v(\boldsymbol{n})$ は，細胞面の位置 k に依存しないことに着目し，図 4 に示すように，U_S を処理するスレッドを串刺し状にまとめ同一ブロックとした．k によらない各 U_S 面内の位置 $\boldsymbol{n}$ を同一ブロック内で計算する．このブロック構成では，計算した抑制性細胞の応答 $v(\boldsymbol{n})$ を他

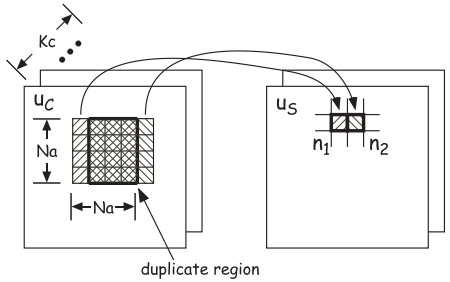

図 5　隣り合う 2 つの細胞に接続する入力細胞
Fig. 5　Input cells of two adjacent cells.

のブロックで参照する必要がなくなるため，他のブロックからも参照できるように $v(\boldsymbol{n})$ をグローバルメモリへ書かかなくてよい．これにより，T2 適用前に対して U_S 細胞層あたり $Ns \cdot Ns$ 回の書き込みを削減する．前節に述べた $u_S(ks, is, js)$ と同様，実際のスキャッタ構造の実装では，$v(\boldsymbol{n})$ のアクセス回数は多少異なるが，6 章で述べる実験の条件での実測では，$v(\boldsymbol{n})$ のためのグローバルメモリの読み出しおよび書き込み回数は，1 文字の認識あたりどちらも約 11 万回であった．これが T2 の適用により 0 回となる．

また，位置 $(\boldsymbol{n}, k)$ の S 層応答計算（式 (1)）と，その計算で必要となる抑制性細胞の応答 $v(\boldsymbol{n})$ の計算（式 (2)）では同じ u_{Cl-1} を必要とすることに着目し，式 (1) の分子，および分母である式 (2) を同時に計算する．実際の CPU 実装では，式 (2) の計算のための u_{Cl-1} アクセスは，1 文字の認識あたり 5.6 万回であった．T2 により，これを 0 回とすることができる．

5.4　スレッド/ブロック構成（T3）

各スレッドでは，U_S 層の 1 つの位置 $\boldsymbol{n}$ について応答計算する．したがって，1 グリッドを $Ns \cdot Ns$ ブロックとし，U_S 層のすべての u_S を 1 グリッドで計算する構成とした．

5.5　共有メモリの利用効率を高める高速化（T4）

図 5 に示すように，水平方向に隣り合う 2 つの計算対象 $\boldsymbol{n_1}$，$\boldsymbol{n_2}$ があるとき，$\boldsymbol{n_1}$ に接続する $Na \cdot Na \cdot Kc$ の入力細胞と $\boldsymbol{n_2}$ に接続する $Na \cdot Na \cdot Kc$ の入力細胞には $Na(Na-1)Kc$ の重複がある．このことに着目し，$\boldsymbol{n_1}$ および $\boldsymbol{n_2}$ のいずれかに接続する $Na(Na+1)Kc$ の入力細胞をグローバルメモリから共有メモリにコピーしてから計算することで，共有メモリ中の $Na(Na-1)Kc$ は 2 つの位置の計算で参照され，共有メモリの利用効率を高めることができる．この高速化のため，1 ブロックで計算に使用する入力細胞は共有メモリを使用する（**D4**）．スレッド/ブロックが Ks を超える場合，図 4 水平方向に同一ブロックで計算する細胞を追加するものとする．

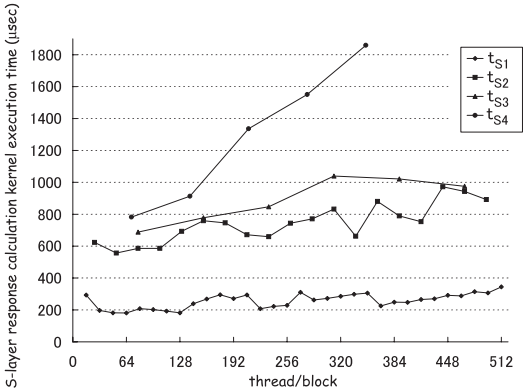

図 6　スレッド/ブロックと計算時間
Fig. 6　Thread/block and calculation time.

また，デザインパターン **D1** の適用による性能向上のため，スレッドあたりに消費するメモリリソースを可能な限り削減する．これにより，各ステージごとに可能な限りスレッド/ブロックを増やすことが可能となる．GPU では，スレッド/ブロックの増加に従い，単調に実行速度が向上するとは限らないため，実行可能な範囲ですべてのスレッド/ブロックの組合せを試し最適なスレッド/ブロックをステージごと決定する．**図 6** に示すように，スレッド/ブロックを増加させ計算時間が最小となる最適値を決定することができる．

5.6　入力細胞の選択を高速化する逆引き索引による高速化（T5）

計算カーネルでは，計算対象とする位置 $\boldsymbol{n}$ に接続する入力細胞の選択が必要である．入力細胞の範囲は，計算によって求めることができるものの，細胞面の周辺部分では，細胞面の中央部分とは異なる小さな半径が用いられている．高速化のため，この計算を避けあらかじめ求めた逆引き索引を用意し，定数メモリに配置する．

5.7　デザインパターンを適用する際の困難と解決

D1 および D3 を適用するため，スレッド/ブロックは 32 の倍数とすべき[18] とされている．スレッドが 32 個単位で管理されていること，および Coalesced アクセスのためにはスレッド/ブロックの 32 の倍数化は重要であると考えられる．本研究も，当初スレッド/ブロックを 32 の倍数とするような実装に取り組んだ．しかし，ネオコグニトロンは，表 2 に示すよ

うに細胞構成，入力接続数といった構成パラメータのいずれも 32 の倍数になっていない点で問題がある．スレッド/ブロックを 32 の倍数とする実装（GPU 実装 B）では，Coalesced アクセスは容易になるもののネオコグニトロンの構成との差異に対応するためにプログラムが複雑化する，配列をパディングするためにデータ領域が非効率化するといった，高速化の面からは望ましくない事象が生じた．GPU 実装 B では，CPU 実装（後述する CPU 実装 F）に対して，ステージ S1〜S4 で最大約 3.3〜6.3 倍を達成したが，ネオコグニトロンの構成に準じたスレッド/ブロック構成とした GPU 実装 T では，全ステージ（S1〜S4）で GPU 実装 B を上回る約 8〜20 倍の高速化を達成している．

ネオコグニトロンのように問題の構成が 32 の倍数と合わない場合，GPU のアーキテクチャを優先し，スレッド/ブロックを 32 の倍数化した実装と，問題の構成を優先するスレッド/ブロック構成の実装の両方を評価する必要がある．

6. 性能評価

本章では，高速化手法を適用した実験結果を示し，性能を評価する．実験に用いた PC および GPU の主な仕様を**表 4** に示す．使用した CPU は 2 コアであるが，1 コアを使用する実装とした．入力する認識対象文字は，Fukushima のデータ 400 文字を用いた．CPU 実装の計算時間は，400 文字に対する 800 回の文字認識において S 層応答計算手続きの実行時間を計測し，800 分の 1 とした．GPU 実装の計算時間は，CPU 実装内の S 層応答計算手続きの呼び出し元で，GPU に実装した S 層応答計算カーネルを呼び出してから実行が終了するまでの時間を計測し，その他の条件は CPU の場合と同じである．CPU 実装の時間計測には Linux のシステムコールである gettimeofday() を用い，GPU 実装の時間計測には，CUDA で高精度にパフォーマンスを測定する方法とされている[19] cudaEventElapsedTime()，および CPU 同様 gettimeofday() の両方を用いた．

表 4　実験環境
Table 4　Experimental environments.

CPU G33 chipset	Core2 Duo 2.33 GHz（4 MB L2） FSB 10.6 GB/s Memory Bandwidth 17 GB/s PCI-e x16 8 GB/s
Main memory	1 GB DDR3 dual channel
GPU	nVIDIA GTX 280 CUDA Core 240 Graphics Clock 602 MHz Processor Clock 1296 MHz Memory Clock 1107 MHz Shader Memory 1 GB Memory Bandwidth 141.7 GB/s
OS	Linux

6.1 GPU 実装の性能評価

ネオコグニトロンの S 層応答計算を，5 章で述べた手法により GPU を用いて高速化した実験結果を**表 5** に示す．表では，手法 T1 のみを適用し全体を 1 グリッド，256 スレッド/ブロックで計算する GPU 実装 T1，手法 T1〜T3，T1〜T4，T1〜T5 をそれぞれ適用した GPU 実装の実行結果を GPU 実装 T3，T4，T5 とし，t は S 層応答計算の各ステージ（S1〜S4）における計算カーネルの実行時間（マイクロ秒），GPU 実装 T3，T4，T5 の GPU 実装 T1 に対する高速化比を t_{T1}/t として示す．

表 5 から，GPU 実装 T3 は，実装 T1 に対してステージ S2〜S4 で約 1.7〜2.7 倍高速，S1 では低速である．手法 T2 の串刺しブロック構造による高速化効果は S2〜S4 では現れている．S1 は S2〜S4 に対して特に細胞層の細胞面数（Ks）が 26〜78 であるのに対し，S1 では 16 と低い（表 2）．手法 T2 は細胞面数 Ks をスレッド/ブロックとしているため，Ks が高速化効果に大きく影響する．このため，S1 では手法 T1 の高速化効果以上に T2 の効果が得られていないと考えられる．Ks が小さい場合にスレッド/ブロックを増加することで，高速化効果を高める手法およびその効果は後述する．

GPU 実装 T4 は，それぞれのステージについてスレッド/ブロックの最適値を決める．各ステージごとのスレッド/ブロックと計算時間 t の関係を図 6 に示す．手法 T4 では，スレッド/ブロックは Ks の倍数である．S1，S2 では計算時間 t を最小化するスレッド/ブロックの最適値を示すピークがグラフに現れており，それぞれ 64，54 である．S3，S4 ではスレッド/ブロックが最小値 78，70 のとき t の最小となっている．スレッド/ブロックの変更による処理速度の変化は，GPU のスレッド制御，メモリアクセスにともなうレイテンシといっ

表 5　GPU 実装の S 層応答計算カーネル実行結果
Table 5　The result of S-layer response calculation kernel execution of GPU implementations.

	t（マイクロ秒）				t_{T1}/t			
	S1	S2	S3	S4	S1	S2	S3	S4
GPU 実装 T1	346	1,340	1,980	2,220	-	-	-	-
GPU 実装 T3	388	769	721	1,270	0.891	1.74	2.74	1.75
GPU 実装 T4	177	573	667	847	1.96	2.34	2.97	2.62
GPU 実装 T5	168	547	627	736	2.06	2.45	3.16	3.02

た複数の要因が関係しており，振舞いは単純でない．したがって，最適なスレッド/ブロックを決定するためには実測が必要である．

GPU 実装 T4 は，実装 T1 に対してすべてのステージについて約 2.0〜3.0 倍高速である．また，実装 T2 に対しても全ステージについて高速化である．手法 T4 により共有メモリを活用し，U_C に関するグローバルメモリアクセスの削減およびスレッド/ブロック割当てを改善したことによる高速化効果が現れている．同一ブロックのスレッドを増加させたことで，共有メモリの U_C を共用する効率も高くなることも，グローバルメモリのアクセス削減による高速化に寄与している．GPU 実装 T3 では Ks の小さな S1 において実装 T1 より低速であったが，手法 T4 の適用により，適用前に 16 であった S1 でのスレッド/ブロックは 64 となり，高い高速化効果が得られている．

GPU 実装 T5 は，それぞれ実装 T4 に比べ数%高速である．手法 5 の適用による逆引き索引によるアドレス計算時間の削減では，大幅な高速化は得られないものの，逆引き索引を使用する前と比べ，実行命令数を確実に削減できるため，削減した実行命令数に相当する高速化が期待できる手法である．計算カーネルの総実行命令数がより小さければより高い効果が期待できる．S 層応答計算の計算カーネルは，5.1 節に述べた理由により長いため，高い効果は期待できない．

6.2 CPU 実装に対する性能評価

高速化手法の CPU 実装に対する性能評価のため，3.4 節で述べた実装を CPU 実装 F とした．CPU 実装 F は，3.3 節で述べた C 層細胞の条件による計算量削減を含む実装である．この条件分岐による計算量削減の効果を考慮し比較するため，この計算量削減をしない実装を CPU 実装 S とした．また，ギャザ型実装（手法 T1）は CPU 実装に対しても高速化の可能性があるため，CPU 実装 F に手法 T1，T2，T5 を適用し CPU 実装 G とした．CPU 実装 F，CPU 実装 S，および CPU 実装 G の実験結果を**表 6** に示す．表では CPU 実装の S 層応答の計算時間，メモリバンド幅，浮動小数点演算速度および CPU 実装 S に対する CPU 実装 F の高速化比を t（ミリ秒），M（MB/s），F（MFLOPS）t_S/t_F として示す．M は応答計算において実行したメモリアクセスを，F は応答計算において実行した浮動小数点演算をカウントしたものをそれぞれ計算時間 t で割って得たものである．すべての手法 T1〜T5 を適用した GPU 実装 T と CPU 実装との比較を**表 7** に示す．t_S/t は CPU 実装 S に対する高速化比，t_F/t は CPU 実装 F に対する高速化比 t_G/t は CPU 実装 G に対する高速化比である．

手法 T4 は，データアクセスにおける局所性に着目したものであり，CPU ではこれに近い高速化手法として，キャッシュ・ブロッキング（タイリング）[20] がある．CPU 実装 F および CPU 実装 G にこの手法適用しその効果を確認した結果，CPU 実装 F では S1〜S4 で 1.5%低速化から 3.4%高速化，CPU 実装 G では 8.3〜3.8%低速化した．顕著な高速化効果が現れないのは，キャッシュ・ブロッキングによる高速化効果が手法適用のため追加したループのオーバヘッドにより相殺されている可能性があるが，アクセスする配列はたかだか数百 KB（表 2）であり，CPU の L2 キャッシュ・サイズ 4 MB（表 4）に対して十分に小さいことから，キャッシュ・ブロッキング適用前にもデータ・キャッシュが十分に機能しており，メモリアクセスのレイテンシは小さい状態と推定できる．

CPU 実装 G は CPU 実装 F より遅い．5.2 節で述べたように，ギャザ型への転換によって起こる C 層細胞の条件判断の実行回数の増加は低速化の要因である．本実験では，CPU では積算の入力のメモリアクセスのコストよりも出力のメモリアクセスのコストに大きな差がない．このため低速化したと考えられる．

表 7 には，GPU 実装 T は，CPU 実装 S に対して約 45〜230 倍，CPU 実装 F に対して約 8〜20 倍，CPU 実装 G に対して約 63〜210 倍の高速化が示されており，本手法による GPU を用いた高速化により，CPU 実装に対して高速化が達成されている．ただし，CPU 実装は 1 コアを使用した実装であるため，2 コアを使用した場合にはこの半分程度である可能性がある．

CPU 実装 F の全ステージおよび S1〜S4 において，L4〜L6 の計算が省略されず実行された比率を C_{ALL}，C_{S1}〜C_{S4} として**表 8** に示す．表 8 および表 6 から，CPU 実装 F は，

表 6　CPU 実装の S 層応答計算手続き実行結果
Table 6　Computation time at S-layer response calculation procedure by CPU implementations.

	t（ミリ秒）				M（MB/s）				F（MFLOPS）			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
CPU S	29.7	116	144	32.9	586	555	532	543	179	194	145	129
CPU F	3.33	6.91	9.98	6.01	600	551	529	570	181	189	147	206
CPU G	23.0	72.5	131	46.0								
t_S/t_F	8.9	17	14	5.5								

表 7　GPU 実装と CPU 実装の比較
Table 7　Comparison of GPU and CPU implementations.

	t_S/t				t_F/t				t_G/t			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
GPU T	177	213	231	44.9	19.9	12.7	15.9	8.2	137	133	209	62.7

表 8　S1〜S4 における，C_{uc} 真の比率
Table 8　True ratio of C_{uc}.

	C_{ALL}	C_{S1}	C_{S2}	C_{S3}	C_{S4}
True ratio of C_{uc}	5.7%	7.8%	4.6%	5.2%	6.3%

CPU 実装 S に対して，ほぼ，C 層細胞の条件による計算の削減に応じた高速化がみられる．メモリバンド幅および浮動小数点演算速度は S1〜S3 でほぼ同じ結果を示しているが，S4 の浮動小数点演算速度では CPU 実装 S が遅く，CPU 実装 F が約 1.6 倍速い．これらから，CPU 実装 F を GPU を用いた高速化手法の評価に用いても問題ないと考えられる．

表 6 には，CPU 実装 F の最大メモリバンド幅および浮動小数点演算速度は約 600 MB/s および 210 MFLOPS であり，CPU の最大性能である 17 GB/s および 4.66 GFLOPS より大幅に低いことが示されている．前述のようにメモリアクセスのレイテンシは十分小さい状態であると推定できるため，CPU 実装 F の S 層応答計算では，メモリアクセスおよび浮動小数点演算を除くループ，条件分岐などのフロー制御および，整数演算が実行時間の多くを占めていると考えられる．3 次元畳み込みでは，多くのメモリアクセスをともなうことを 3.2 節で述べた．U_{S1}〜U_{S4}，U_{C1}〜U_{C4} はいずれも 3 次元配列であるため，配列メモリアクセスには配列要素のメモリアドレスの計算が付随し，多く実行される．

6.3　GPU 実装のボトルネック

GPU 実装 T は，プロファイラ cudaprof[21] により収集したイベントカウントでは，命令スループットは約 0.78，グローバルメモリの総スループットは約 18.0 GB/s，実行命令数は 1.79×10^9 個，および warp シリアライズは 4.77×10^7 回である．命令スループットは，単位時間あたりに実行できる最大の命令数に対して実際に実行した命令数の比である．1 のとき最大であるが，実行できる最大の命令数は 1 命令発行を基準としているため，多重命令発行が多く実行されると，1 を超えることがある．メモリのスループットは GPU の最大性能である 141.7 GB/s に対して低いが，メモリのレイテンシが実行速度に与えている影響は，命令スループットの値からたかだか約 22%であり，命令数に対し約 2.7%で warp シリアライズを起こしていることからも，命令ボトルネックの状態である．したがって，命令数の削減による効果は，総命令数が多い場合には相対的に効果が小さいことからも，GPU 実装 T の性能を大幅に改善するには命令数を大幅に削減する必要がある．

6.4　総合性能の評価

CPU 実装 F を用いた 1 文字あたりの総計算時間は 27.3 ミリ秒である．GPU 実装 T における CPU と GPU 間のデータ転送時間は 0.27 ミリ秒であり，GPU 実装 T を用いた 1 文字あたりの総計算時間は，このデータ転送時間，および GPU による S 層応答計算時間以外の CPU の計算時間を含み 3.4 ミリ秒である．これらから，CPU 実装 F，GPU 実装 T それぞれが毎秒 30 フレームの入力画像に対して処理可能な文字数は，1 文字および 9 文字相当である．

本実験では，入力画像が文字領域に限定されるため CPU と GPU の間の転送データサイズは小さく，総計算時間の 8%程度である．今後，転送する画像領域を大きくし転送時間が長くなると，CPU と GPU 間のデータ転送がボトルネックとなることが考えられる．その場合，データ転送の非同期コンカレント実行[14] を活用し，データ転送と計算処理を並列に実行することでボトルネックを解消できると考えられる．

本実験では 400 文字を用いたが，認識率の立場からは，この文字数では十分とはいえない，しかし，それらの文字に対するネオコグニトロン実装の性能を確認し，研究の方向性を決めるのには十分であると考えている．数字より密度の高いデータに対しては，条件判断によって省略する計算量が減少するため計算時間は長くなるが，計算を省略することによる高速化効果が減少するために，CPU 実装 F に対する GPU 実装 T の高速化比は高くなることが予測できる．また，計算を実行する比率が増加するため，メモリバンド幅および演算スループットは高くなると考えられる．さらに密度が高い画像認識を対象とするのであれば，W_5 の実装のように，条件判断をせず，命令数が増加する要因を避けた実装の方が高速に処理できると考えられる．

7. ま と め

本研究では，ネオコグニトロンの S 層応答計算を GPU を用いて高速化し，S 層応答計算の CPU 実装に対して，それぞれ性能を評価した．実験では，サイズの異なる 4 つのステージについて，CPU 実装に対して約 8〜20 倍高速化しており，本研究で高速化のために適用した以下の手法による高速化効果が確認できた．

- グローバルメモリへの書き込み回数を最小化するギャザ構造スレッド
- 興奮性細胞から S 層への入力と抑制性細胞の応答計算を同時に計算することでグローバルメモリへのアクセスを削減するブロック構造
- グローバルメモリからの読み込み回数を削減するため共有メモリの利用および効率を高める手法
- 入力細胞の選択を高速化する逆引き索引による高速化

また，1 文字あたりの総計算時間は 27.3 ミリ秒から 3.4 ミリ秒に短縮させた．これによ

り，本研究の目標である毎秒 30 フレームのカメラ画像に対し，数字 8 桁の認識に必要な速度が達成されている．GPU を用いた高速化の実現は重要な目標であったが，CPU のホットスポットを GPU へオフロードすることで，CPU では他の処理が実行できる点でも有用である．

以下は今後の課題である．

- コア数やキャッシュ容量が増加した GPU および新たな機能を持つ GPU により可能となる新たな実装の検討．
- 1 台の PC に複数の GPU を搭載することなどによる，多文字の同時処理．
- ステージ S1〜S4 の S 層応答計算をパイプライン実行することによる，複数文字の同時並列処理．

謝辞　本研究の一部は，科学研究費補助金基盤研究（A）（20240002）および大阪大学グローバル COE プログラム「予測医学基盤」の補助による．また，有益なご意見をいただいた査読者の方々に感謝いたします．

参 考 文 献

1) Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics*, Vol.36, No.4, pp.193–202 (1980).
2) Poli, G., Saito, J.H., Mari, J.F. and Zorzan, M.Z.R.: Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture, *Proc. 20th Int'l Symp. Computer Architecture and High Performance Computing* (*SBAC-PAD'08*), pp.81–88 (2008).
3) Iwamura, M., Tsuji, T., Horimatsu, A. and Kise, K.: Real-Time Camera-Based Recognition of Characters and Pictograms, *Proc. 10th Int'l Conf. Document Analysis and Recognition* (*ICDAR'09*), pp.76–80 (2009).
4) Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A. and Purcell, T.J.: A Survey of General-Purpose Computation on Graphics Hardware, *Computer Graphics Forum*, Vol.26, No.1, pp.80–113 (2007).
5) GPGPU: General-Purpose Computation Using Graphics Hardware (2007). http://www.gpgpu.org/
6) Fatahalian, K., Sugerman, J. and Hanrahan, P.: Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication, *Proc. 19th SIGGRAPH/EUROGRAPHICS Workshop Graphics Hardware* (*GH'04*), pp.133–137 (2004).
7) Stock, M.J. and Gharakhani, A.: Toward efficient GPU-accelerated N-body simulations, *Proc. 46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2008-608, pp.76–80 (2008).
8) NVIDIA Corporation: NVIDIA GPU Computing Developer Home Page (2010). http://developer.nvidia.com/object/gpucomputing.html
9) 伊藤崇之，福島邦彦，三宅　誠：並列計算機による神経回路モデルの実現手法の検討—ネオコグニトロンの NCUBE 上での実現，信学論，Vol.J73-D-I, No.1, pp.1–8 (1990).
10) 大森隆司，佐藤克巳：並列計算機システムでのネオコグニトロンの実現，情報処理学会研究報告，Vol.1991, No.64, pp.67–74 (1991).
11) 木村　要，高橋　聖，中村英夫：ネオコグニトロン型動画像識別モデルのハードウェア実装に関する検討，信学技法，NC2003-111, pp.19–24 (2004).
12) Billconan and Kavinguy: A Neural Network on GPU (2008). http://www.codeproject.com/KB/graphics/GPUNN.aspx
13) Fukushima, K.: Neocognitron for handwritten digit recognition, *Neurocomputing*, Vol.51, pp.161–180 (2003).
14) NVIDIA Corporation: *CUDA Programming Guide Version 2.3.1* (2009). http://developer.nvidia.com/cuda/
15) NVIDIA Corporation: NVIDIA GPU Computing Developer Home Page. http://developer.nvidia.com/object/gpucomputing.html
16) NVIDIA Corporation: *NVIDIA CUDA C Programming Best Practices Guide CUDA Toolkit 2.3* (2009). http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVID%IA_CUDA_BestPracticesGuide_2.3.pdf
17) Ryoo, S., Rodrigues, C.I., Baghsorkhi, S.S., Stone, S.S., Kirk, D.B. and Hwu, W.W.: Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA, *Proc. 13th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming* (*PPoPP'08*), pp.73–82 (2008).
18) NVIDIA Corporation: *NVIDIA CUDA C Programming Best Practices Guide CUDA Toolkit 2.3*, chapter 4.4 Thread and Block Heuristics [16] (2009). http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVID%IA_CUDA_BestPracticesGuide_2.3.pdf
19) NVIDIA Corporation: *NVIDIA CUDA C Programming Best Practices Guide CUDA Toolkit 2.3*, chapter 2 Performance Metrics [16] (2009). http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVID%IA_CUDA_BestPracticesGuide_2.3.pdf
20) Wolfe, M.: *High Performance Compilers for Parallel Computing*, chapter 10 OPTIMIZING FOR LOCALITY, Addison Wesley (1995).
21) NVIDIA Corporation: *NVIDIA CUDA Visual Profiler Version 2.3* (2009). http://developer.nvidia.com/object/cuda_2_3_downloads.html

(平成 22 年 10 月 1 日受付)
(平成 23 年 1 月 23 日採録)

池田　孝利（学生会員）

平成元年大阪大学基礎工学部生物工学科卒業．現在，同大学院情報科学研究科博士課程後期在学中．並列処理全般に興味を持つ．

宮本　弘之

昭和 60 年大阪大学基礎工学部生物工学科卒業．昭和 62 年同大学院基礎工学研究科博士前期課程修了．住友電工を経て平成 6 年同大学院同研究科博士後期課程中退．同年 ATR 人間情報通信研究所滞在研究員．平成 8 年科学技術振興事業団川人学習動態脳プロジェクト研究員．平成 13 年九州工業大学大学院生命体工学研究科助教授．現在，同大学院准教授．博士（工学）．ニューラルネットワークとロボティクスに関する研究に従事．

伊野　文彦（正会員）

平成 10 年大阪大学基礎工学部情報工学科卒業．平成 12 年同大学院基礎工学研究科修士課程修了．平成 14 年同大学院同研究科博士課程中退．同年同大学助手．現在，同大学准教授．博士（情報科学）．高性能計算に関する研究に従事．

萩原　兼一（正会員）

昭和 49 年大阪大学基礎工学部情報工学科卒業．昭和 54 年同大学院基礎工学研究科博士課程修了．工学博士．同大学助手，講師，助教授を経て，平成 5 年奈良先端科学技術大学院大学教授．平成 6 年より大阪大学教授．平成 4〜5 年文部省在外研究員（米国メリーランド大学）．現在，並列処理の基礎および応用に興味を持っている．