Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 8, Number 2, pages 116–121, July 2019

An Efficient Convolutional Neural Network Computation using AVX-512 Instructions

Hiroki Kataoka[1], Kohei Yamashita[1], Koji Nakano[1], Yasuaki Ito[1], Akihiko Kasagi[2], and Tsuguchika Tabaru[2]
[1] Department of Information Engineering，Hiroshima University
1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan
[2] Fujitsu Laboratories Ltd. 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa
211-8588, Japan

***Abstract*—Recently, Convolutional Neural Networks (CNNs) are widely used for image processing. Since the computation cost is high, it is necessary to accelerate the computation. Therefore, in this paper, we propose an efficient implementation using Intel AVX-512 instructions on the multicore CPUs. AVX-512 instructions suppose 512-bit vector operations, in which 16 32-bit floating point number operations can be executed simultaneously. In this implementation, to reduce the computation, we use an idea of the fused filter that combines a convolutional layer and its following pooling layer. As a result, we achieve a speed-up factor of 1.62 over an existing library implementation using Intel Math Kernel Library for Deep Neural Networks.**

***Index Terms*—Deep Learning, Neural Networks, Convolution, Average Pooling**

I. はじめに

画像処理や自然言語プロセスといった分野では，データの特徴の抽出に畳み込みニューラルネットワーク (CNN) と呼ばれる，主に畳み込み層とプーリング層の 2 つの層からなる計算モデルが使用されている．近年では，畳み込み層をより深く連結させた深層学習という手法が著しい成果を上げており，その計算の高速化を容易に可能とする為のソフトウェアアクセラレータの開発が現在でも盛んに行われている．

特に，GPU(Graphics Processing Unit) の性能の向上や，機械学習向けのフレームワークが NVIDIA GPU 向けのソフトウェアアクセラレータを取り入れたことにより，多くのユーザが GPU の計算リソースを用い始め，CNN の開発がより容易に可能となった．しかし，GPU デバイスメモリの容量には制限がかかる等の理由により，パラメータを膨大に持つ CNN などでは依然として CPU を用いた畳み込み計算を要求される場合が存在する．CPU におけるソフトウェアアクセラレータとしては，ベクトル演算ユニットを搭載するコアを複数搭載した CPU を用いて複数の計算を並列に処理することによる高スループット計算に関する研究・応用が盛んである．

そこで本論文では SIMD 演算器を各コアに搭載した Intel CPU を用いたマルチチャネル畳み込みの効率的な並列実装の手法を提案する．本研究では Intel Core i9-7980XE を用い，Intel CPU に搭載されている 512bit の SIMD 演算器を利用するための拡張命令セットである AVX-512 を利用したベクトル演算を用いて畳み込み及びプーリングの効率的な並列実装を行った．そして，Intel CPU 向けのアクセラレータである Intel Math Kernel Library for Deep Neural Networks(MKL-DNN) と畳み込み及びプーリングの計算の実行時間を比較した結果，提案手法は最大で 1.62 倍もの高速化を達成した．

II. Convoluiton Neural Network(CNN)

Convoluiton Neural Network(CNN) は画像処理や自然言語処理といった分野で用いられる，学習と推論の 2 つのプロセスからなる計算モデルである．CNN は主に 2 つの層からなり，一つは複数回のマルチチャネルの入力に対し，畳み込み計算を行う畳み込み層で，もう一つは計算量削減のためのサンプリングを行うプーリング層である．CNN では入力に対して畳み込み層とプーリング層の計算を複数回繰り返すことでデータの特徴を抽出することが可能であり，特に近年は高性能な CPU や GPU により以前よりも豊富な計算リソースの確保が容易になったため，畳み込み層をより深く連結させた深層学習という手法が著しい成果をあげている [1], [2]．例えば，krizhevsky らが作成した AlexNet は画像の認識精度を競うコンペティションである ImageNet Large Scale Visual Recognition Challenge (ILSVRC) において 2012 年当時に於いて最高精度であるという記録を残している [3].

また，ユーザが CPU や GPU を始めとする計算リソースの性能を最大限引き出すことが出来，より多くの層をもつ CNN の計算の高速化を容易に可能とする為のソフトウェアアクセラレータの開発が現在でも盛んに行われている [4], [5].

特に，最近の GPU の性能の向上及び，TensorFlow のような機械学習に用いられるフレームワークが NVIDIA GPU 向けのソフトウェアアクセラレータである cuDNN を取り入れた事により，多くのユーザが GPU の計算リソースを用いるようになり，CNN の開発がより容易に可能となっている．

しかし, GPU のデバイスメモリの容量には制限がかかることなどの理由から, パラメータを膨大に持つ CNN などでは依然として CPU を用いた畳み込み計算を要求される場合が存在する [6]. CPU におけるソフトウェアアクセラレータの一つとして Intel CPU 向けのアクセラレータである Intel Math Kernel Library for Deep Neural Networks(MKL-DNN) がある．これは Intel CPU のマルチコアや SIMD 演算器，キャッシュメモリなどの計算リソースをユーザが簡単に利用できるように開発された機械学習用のプリミティブ関数を提供するオープンソースのライブラリである. 最近では TensorFlow などの機械学習用フレームワークのアクセラレータとしても利用されている [7]. このように深層学習の計算高速化のための様々なソフトウェアが開発されてきている．

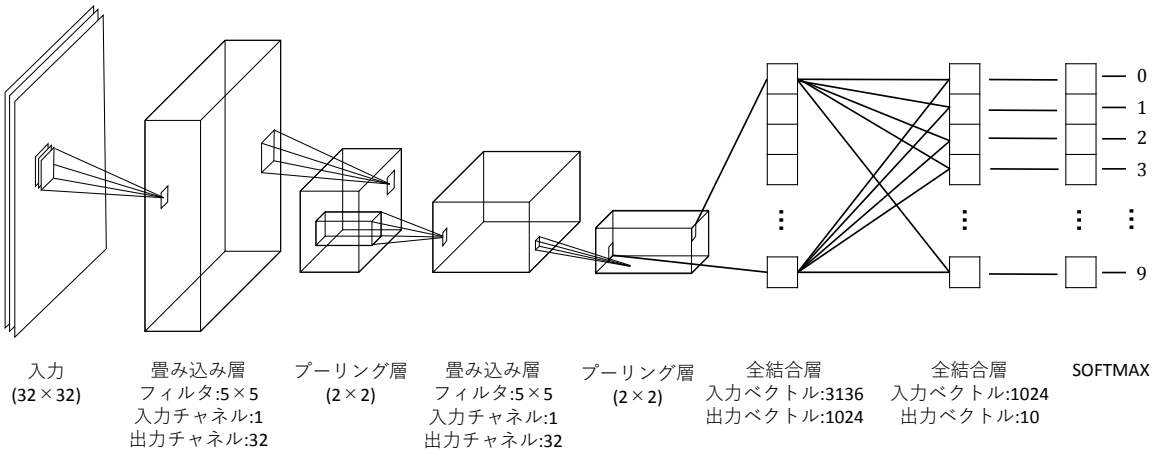


Fig. 1. MNIST 手書き文字認識で利用される CNN アーキテクチャ

現在では CNN アーキテクチャについて様々なアーキテクチャが設計されており，例えば MNIST 手書き文字認識でよく用いられる CNN アーキテクチャは図 1 に示すものである．この CNN アーキテクチャは畳み込み層と全結合層を 2 層ずつ持っており，近年用いられる CNN アーキテクチャとしては比較的層が浅く，単純なものとなっている．

各層で行われる計算について説明する．まず畳み込み層では，図 2 に示すように，各入力チャネルに対して畳み込み計算を実行し，その出力結果の合計を出力結果とする. 出力チャネル数は畳み込みカーネル (フィルタ) のセット数と同じ個数になる．

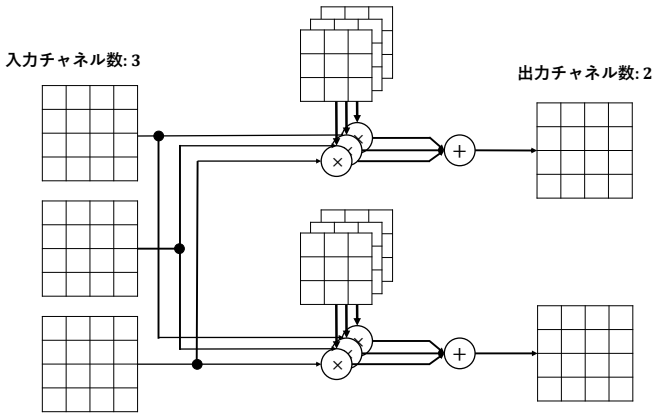
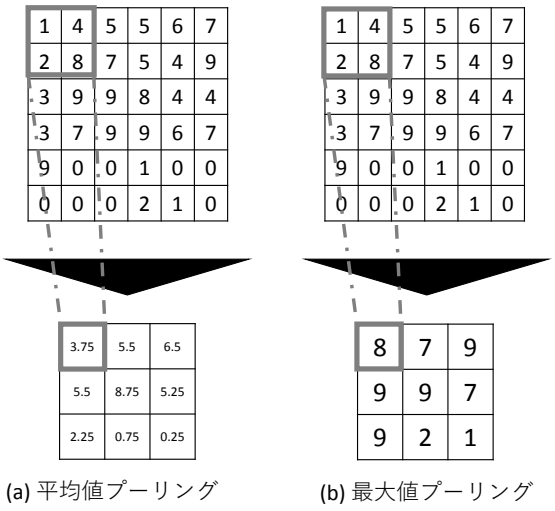


Fig. 2. 畳み込み層

次に，プーリング層では畳み込み層での計算量を削減するために入力をサンプリングする層として働く．このプーリング層でよく使われるアルゴリズムとして平均値プーリングと最大値プーリングがある. 図 3 の (a) は平均値プーリングの例を示しており，(b) は最大値プーリングの例を示している. 平均値プーリングでは，プーリングサイズ p のとき $p \times p$ 領域の平均値を出力の 1 要素とし，最大値プーリングでは最大値を出力の 1 要素とする．



(a) 平均値プーリング　(b) 最大値プーリング

Fig. 3. プーリング層

次に，全結合層の例を図 4 に示す. 全結合層では入力ベクトルの左からパラメータ行列を掛けることで出力を得る．

先述の通り，図 1 の CNN は比較的小さい CNN であるにも関わらず，畳み込み層だけでも推論実行におよそ 1.07×10^7 回の積和演算が必要であり，これは CNN 全体の計算量の 75%以上を占める. AlexNet のようなさらに深い CNN では 1.0×10^8 回以上の積和演算が必要となる [2]．そこで本研究では Intel のマルチコア CPU を用いてより高速に CNN の推論計算を実行するための手法を提案する．

III. マルチチャンネル畳み込みと平均値プーリング

本節では Convolutional Neural Network(CNN) と CNN で繰り返し計算されるマルチチャネル畳み込みと平均値プーリングの計算方法について説明する [8].

はじめにマルチチャネル畳み込みについて説明をする．図 2 に示されるマルチチャネル畳み込みの計算を一般化すると次式のように定義される [8].

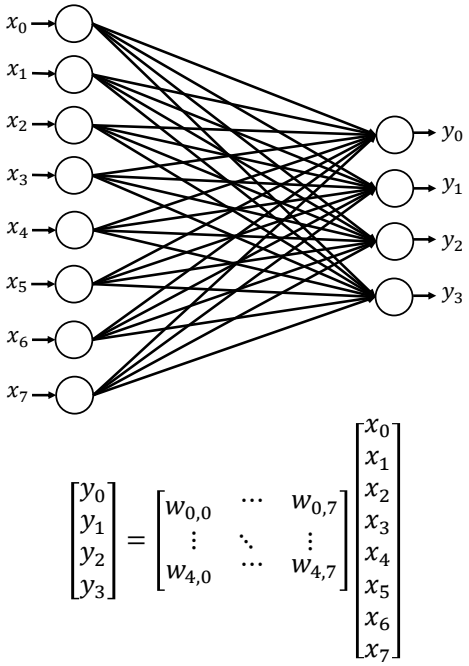


Fig. 4. 全結合層

$$O^m(x,y) = \sum_{ch=1}^{C} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} I^{ch}(x+i, y+i) \times F^{ch,m}(i,j),$$
$$\text{where } x \in [0, W_x - K + 1), y \in [0, W_y - K + 1),$$
$$m \in [1, M].$$

上式において，I は入力される特徴画像の集合，F は畳み込みカーネル (フィルタ) の集合を表す．O は I, F により生成される出力画像の集合である．x, y は画像の二次元座標 x, y に加算されるオフセットであり，その上限値はカーネルサイズ K により決定される．ch は入力チャネルを示しており，$[1, C]$(C は入力チャネル数) の範囲の値を取る．畳み込み結果は ch の次元方向に加算される．m は出力チャネル数を表す．このと $K \times K$ カーネルが $M \times C$ 枚あることになる．$C = 1$ のときの畳み込みはシングルチャネル畳み込みと呼ばれ，$C > 1$ のときマルチチャネル畳み込みと呼ばれる．また，マルチチャネル畳み込みの総積和演算回数は $M \times C \times (W_x - K + 1) \times (W_y - K + 1) \times K^2$ となる．

次に平均値プーリングについて説明をする．図 3 に示される，平均値プーリングの計算を一般化すると次式のように定義される [8].

$$O(x,y) = \frac{1}{P^2} \sum_{i=0}^{P-1} \sum_{j=0}^{P-1} I(x \times P + i, y \times P + i)$$
$$\text{where } x \in [0, W_x), y \in [0, W_y),$$

上式の P はプーリングのサイズを示し，他の変数は先述の畳み込みの場合の定義と同様である. このプーリングの操作は，畳み込み演算が主に画像の特徴量を抽出する目的で行われる一方で，CNN の後続の層の計算量削減が主な目的である．プーリング層を通すことで 1 チャネルあたりの特徴画像のサイズは $\frac{1}{P^2}$ となる．このとき，畳み込み層の積和演算数も $\frac{1}{P^2}$ になる．プーリング層のアルゴリズムとして，他にも最大値プーリングや最小値プーリングなどが存在するが，本研究では平均値プーリングのみを扱う．

IV. CPU アーキテクチャ

本節では実験に用いる CPU のアーキテクチャについて説明する．本研究で使用する CPU である Intel Core i9-7980XE を対象としてアーキテクチャを詳述する [9].

Intel Core i9-7980XE は 1 つのチップ上に 18 コア搭載しており，各コアはパイプラインレベルの並列化技術であるハイパースレッディングを利用することで最大 36 スレッドの並列スレッドを動作させることが可能である．しかし，シングルコアによる実行と比較すると複数コア利用時は CPU の最大動作クロックが低下する．シングルコアによる実行時は最大動作クロック数は 4.2GHz，18 コア利用時の最大動作クロック数は 3.4GHz となる.

また，Skylake-X アーキテクチャの Intel CPU は 512bit の計算を同時実行可能な SIMD 演算器を搭載しており，CPU の各コアには 2 基の積和演算用の計算ユニットである Fused Multiply Add(FMA) 演算器が搭載されている．FMA 演算器では，図 I に示すように，3 つの zmm レジスタと呼ばれる 512bitSIMD レジスタにメモリから 512bit 分の値をロードした後，２つの SIMD レジスタの各要素の積を計算しもう一つのレジスタの値を加算することができる．このときメモリからロードされる値の型が 32bit 数であれば 16 要素，64bit 数あれば 8 要素同時に計算可能である．したがって，32bit 浮動小数点数について，Intel Core i9-7980XE は最大 2×2(基)$\times 16$(要素)$\times 18$(コア)$\times 3.5$[GHz] = 3.9TFLOPs の演算性能を持ち，64bit 浮動小数点数に対しては最大 2×2(基)$\times 8$(要素)$\times 18$(コア)$\times 3.5$[GHz]$=$ 2.0TFLOPs の演算性能を持つことがわかる.

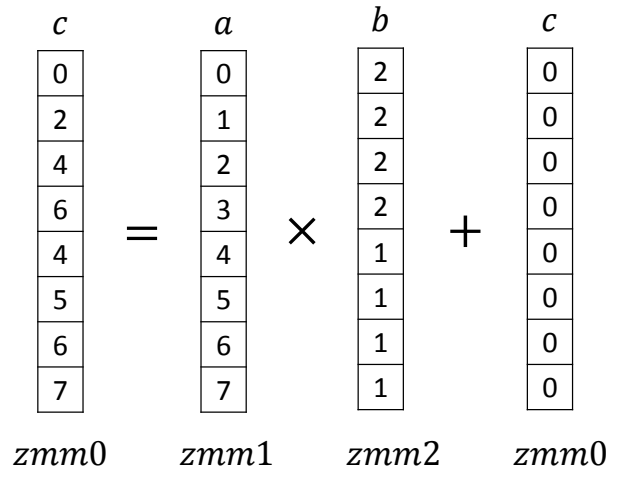


Fig. 5. SIMD 演算器による積和演算の例

Intel Core i9-7980XE では命令セットアーキテクチャとして x86-64 アーキテクチャを使用しており，Intel CPU に搭

載される 512bit の SIMD 演算器を利用するための拡張命令セットである AVX-512 が利用可能となっている．AVX-512 拡張命令を発行することで Intel CPU の 512bit SIMD 演算器を利用したベクトル演算を使用できる．また，AVX-512 では vaddps，vmulps といったベクトル演算命令のオペランドにスカラー値をとることができるように組込みブロードキャスト命令が含まれている．すなわち，スカラー値を明示的にベクトルレジスタへブロードキャストする命令を発行しなくてもオペランドに直接スカラー値を書くことでスカラーとベクトルの演算が可能である．これにより，マルチチャネル畳み込みで行われる明示的なブロードキャスト命令数の削減が可能になる．

V. INTEL MKL-DNN によるマルチチャネル畳み込み

本節では，Intel MKL-DNN に実装されるマルチチャネル畳み込みの計算方法について説明する [5]．MKL-DNN では Intel CPU の SIMD 演算器の性能を最大限に発揮するためにメモリ配置を工夫している．通常，CNN では NCHW か NHWC のメモリ配置が用いられる．各メモリ配置の例が図 II に示されている．NCHW ではメモリアドレスの連続方向にラスタスキャン順に画像データが保持されており，NHWC ではメモリアドレスの連続方向がチャネル方向優先順で画像データが保持されている．例えば，Tensorflow ではデフォルトのメモリ配置として NHWC が採用されている．

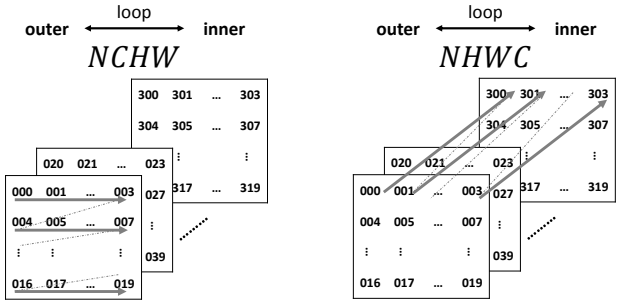


Fig. 6. 一般的な CNN における入力のメモリレイアウト

一方で，Intel MKL-DNN では Intel CPU の SIMD 演算器の性能とマルチコアのスケーラビリティを最大限発揮するために図 III に示すような $nChw16c$ と呼ばれるメモリ配置を用いる．このメモリ配置を利用すればチャネル方向の計算並列性を最大限利用しつつ，ベクトルレジスタのメモリアクセスが単純になるためキャッシュヒット率が向上しメモリ効率が向上する．

そして，$nChw16c$ のメモリ配置に対して実際に畳み込み計算を行う方法を図 8 に示す．まず入力の 1 要素をベクトルレジスタにブロードキャストする．そして畳み込みカーネルのデータをベクトルレジスタにベクトルロードしてから，それらの積を別のベクトルレジスタに累積する．そして 16 個の入出力チャネル分の累積を求めた後出力を格納するメモリにベクトルレジスタの値を加算する．この操作を繰り返すことで畳み込み計算をマルチコアスケーラビリティを確保しつつベクトル演算器とキャッシュメモリを効率的に利用して畳み込み計算を行うことができる．

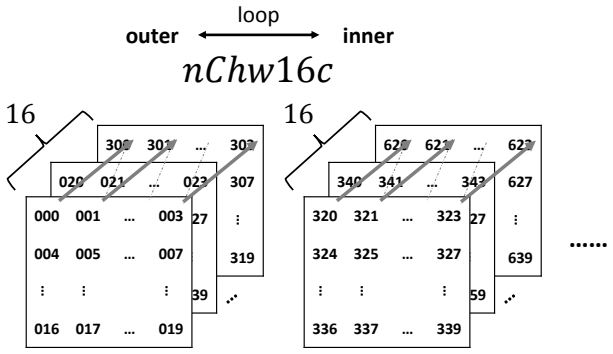


Fig. 7. MKL-DNN における入力のメモリレイアウト

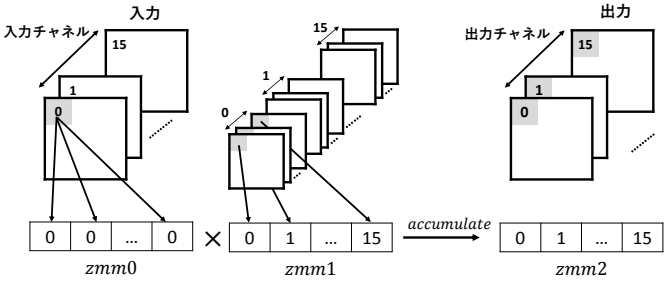


Fig. 8. MKL-DNN における畳み込み計算

VI. 提案手法

マルチチャネル畳み込みと平均値プーリングを組み合わせた提案手法について説明する．先述の Intel MKL-DNN による畳み込み層におけるマルチチャネル畳み込みの計算は Intel CPU のマルチコアスケーラビリティと SIMD 演算器を効率的に活用している．提案手法では多くの場合の CNN アーキテクチャについて畳み込み層の後にはプーリング層が配置されることを考慮して，畳み込み層とプーリング層 (平均値プーリング) の計算を 1 つの層で実行する新しい層を提案する．この新しい層では畳み込み層とプーリング層を個別に実行する場合と比較してメモリアクセス回数が減少するため効率的である．また，提案手法においても $nChw16c$ のメモリ配置を使用する．以下では，この提案手法について詳説する．

図 9 に示すように，通常の CNN では畳み込み層の計算を別の出力用メモリに書き出し，その出力を次のプーリング層に渡すことで連鎖的に計算を行う．すなわち，畳み込み層とプーリング層の計算を実行するにあたり，畳み込み層への入力，畳み込みの結果を保持する中間出力，そしてプーリング層の結果の出力，それぞれのメモリを用意しなければならない．また，中間出力へのメモリ書き出しおよび中間出力からのメモリ読出し命令も発生する．そこで本研究では，中間出力のメモリ読み書きを削減する畳み込みとプーリングの計算を一括で行う層を提案する．

本研究で提案する層では畳み込みの計算と平均値プーリングの際に必要な平均値を求める計算をすべてベクトルレジスタ上で行う．計算の様子を図 10 に示す．

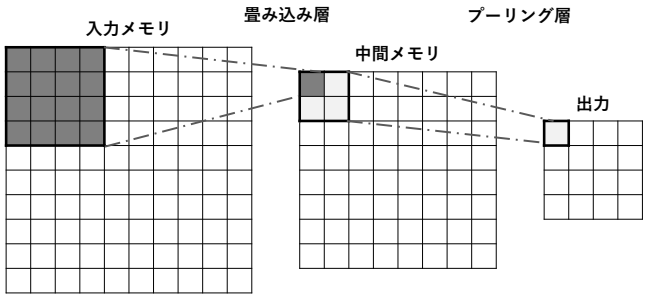


Fig. 9. 通常の畳込み層とプーリング層

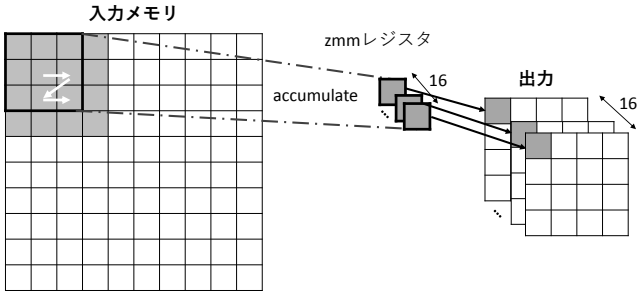


Fig. 10. 提案アルゴリズムによる畳込み層とプーリング層

本研究の提案手法では畳み込み計算がベクトルレジスタに入力値とカーネル値の積を累積していくという方法で行われることを利用する．即ち，畳み込みと平均値計算ではベクトル加算を実行するが加算命令が可換であるため，平均値プーリング層の出力結果を求めるために必要な畳み込み層の出力結果の要素の和をベクトルレジスタ上で計算ができる．そこで提案する新しい層では，中間出力メモリへベクトルレジスタの値を加算せず，ベクトルレジスタの値をクリアせず平均値を求めるために必要な畳み込み結果すべての値をベクトルレジスタへ加算する．そしてベクトルレジスタ上での加算が完了したら畳み込みとプーリング全体の計算結果を出力するメモリへと書き出しを行う．これにより，中間出力のためのメモリ空間を用意する必要がなくなり，メモリ読出しと書き込み命令の回数も削減できる．さらに，予めカーネルの値をプーリングサイズ $p \times p$ で除算しておくことで平均値計算の際の乗除算命令を削減できる．

VII. 性能評価

本節では MKL-DNN の畳み込み層およびプーリング層の計算に対して提案手法が優れていることを示す．実行環境は Intel Core i9-7980XE を用いて，コンパイラには gcc のバージョン 8.2.0 を使用した．コンパイラオプションは-march=native, -fopenmp, -O3, ftree-vectorize を使用し，マルチスレッド実装のために OpenMP5.0 による並列化を行った [10]．使用した CPU アーキテクチャは Skylake-X であり AVX-512 をサポートしている．

表 I, 表 II, 表 III に実行結果を示す．各表において，「入力並べ替え」は入力を $nChw16c$ のメモリ配置に並べ替えるために必要な時間を示している．そして，MKL-DNN では畳み込み層とプーリング層 (3×3 平均値プーリング) を実行し，提案手法では畳み込みとプーリングを一括で実行する．実行結果より入力サイズ 34×34，カーネルサイズ 3×3 の場合について，提案手法は 1.46 から 1.62 倍の高速化を達成したことが分かる．さらに，MKL-DNN の実行結果を見てみると，プーリング層の実行時間は殆ど無視できる程度であり，実行時間の大半を畳み込み層が占めていることが分かる．そして，提案手法における畳み込みとプーリングの操作にかかる時間は，MKL-DNN において畳み込み計算をするためにかかる時間よりも少なくなっている．これは，畳み込み層では画像の縮小を行わないために中間出力のメモリ書き出しのデータ量が提案手法の畳み込みとプーリング操作の出力をメモリに書き出すデータ量よりも非常に多いからだと考えられる．これより，提案手法はメモリへの読み書きデータ量の削減により実行時間が大きく短縮されたと考えられる．

TABLE I
入力サイズ 34×34，カーネルサイズ 3×3, バッチ数 64 の場合の提案手法と MKL-DNN の実行時間比較 (ミリ秒)

		入力チャネル数, 出力チャネル数		
		128, 128	256, 256	512, 512
MKL-DNN	入力並び替え	2.30	4.66	9.82
	畳み込み層	7.64	15.15	29.41
	プーリング層	0.33	0.28	0.28
	全体	10.26	20.08	39.51
提案手法	入力並び替え	1.81	4.24	9.19
	畳み込み&プーリング	4.53	9.03	17.90
	全体	6.34	13.27	27.09
	vs.MKL-DNN	1.62	1.51	1.46

TABLE II
入力サイズ 66×66，カーネルサイズ 3×3, バッチ数 64 の場合の提案手法と MKL-DNN の実行時間比較 (ミリ秒)

		入力チャネル数, 出力チャネル数		
		128, 128	256, 256	512, 512
MKL-DNN	入力並び替え	9.40	17.51	38.81
	畳み込み層	29.83	54.27	90.63
	プーリング層	1.43	1.46	1.48
	全体	40.66	73.24	130.92
提案手法	入力並び替え	8.49	17.01	34.24
	畳み込み&プーリング	17.85	35.44	70.48
	全体	26.34	52.45	104.73
	vs.MKL-DNN	1.54	1.40	1.25

TABLE III
入力サイズ 130×130，カーネルサイズ 3×3, バッチ数 64 の場合の提案手法と MKL-DNN の実行時間比較 (ミリ秒)

		入力チャネル数, 出力チャネル数		
		128, 128	256, 256	512, 512
MKL-DNN	入力並び替え	35.89	63.81	174.33
	畳み込み層	92.53	174.96	259.94
	プーリング層	6.80	6.59	6.44
	全体	129.85	245.36	440.71
提案手法	入力並び替え	32.88	65.11	129.82
	畳み込み&プーリング	69.65	138.52	277.68
	全体	102.53	203.64	407.49
	vs.MKL-DNN	1.27	1.20	1.08

VIII. まとめ

本研究では，CPU を用いた CNN における畳み込み層とプーリング層の計算の効率的な計算手法を提案した．畳み込み層とプーリング層の計算効率的に行う方法として，畳み込みとプーリングを一つの層として扱うことで中間メモリに対する読み書きを削減することで実行速度の高速化を達成した．入力サイズ 34×34，カーネルサイズ 3×3 の場合について，提案手法は 1.62 から 1.68 倍の高速化を達成した．本研究の実験により提案手法は MKL-DNN で畳み込み層のみを実行する場合よりも，メモリの読み書き数が減るために高速に実行可能であることがわかった．多くの CNN のフレームワークの畳み込み計算高速化のために Intel MKL-DNN が利用されており，本研究ではその MKL-DNN に対してマルチチャネル畳み込みと平均値プーリングの計算に対して提案手法がより有効であることを示した．

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, ”Deep learning,” nature, vol. 521, no. 7553, p.436, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ”Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097-1105.

[3] ”IMAGENET,” http://www.image-net.org/.

[4] ”NVIDIA cuDNN,” https://developer.nvidia.com/cudnn.

[5] ”Intel(R) Math Kernel Library for Deep Neural Networks (Intel(R) MKL-DNN) 0.17 Performance library for Deep Learning,” https://github.com/intel/mkl-dnn.

[6] D. Budden, A. Matveev, S. Santurkar, S. R. Chaudhuri, and N. Shavit, ”Deep tensor convolution on multicores,” *arXiv preprint arXiv:1611.06565*, 2016.

[7] ”Tensoflow,” https://www.tensorflow.org/.

[8] Q. Chang, M. Onishi, and T. Maruyama, ”Fast convolution Kernels on Pascal GPU with High Memory Efficiency,” in *Proceedings of the High Performance Computing Symposium*, ser. HPC ’18. San Diego, CA, USA: Society for Computer Simulation International, 2018, pp. 3:1-3:12. [Online]. Available:http://dl.acm.org/citation.cfm?id=3213069.3213072.

[9] ”Intel Core i9-7980XE Extreme Edition Processor,” https://ark.intel.com/products/126699/Intel-Core-i9-7980XE-Extreme-Edition-Processor-24-75M-Cache-up-to-4-20-GHz-.

[10] OpenMP.org, ”OpenMP Application Program Interface,” http://www.openmp.org.