3P1-11in2

スマートフォン上での Deep Learning による画像認識

Image recognition using Deep Learning on the smartphone

丹野 良介　　柳井 啓司
Ryosuke Tanno　　Keiji Yanai

電気通信大学大学院 情報理工学研究科 情報学専攻
Department of Informatics, The University of Electro-Communications

Deep Learning is becoming the de facto standard method in generic object recognition at the cost of a large number of parameters and computational capacity. However, because mobile devices such as smartphones have limited resources in terms of both computational power and memory, it is not easy to implement Deep Learning on mobile devices. In this paper, we propose a deep learning based image recognition system running on a consumer smartphone employing novel implementation techniques for acceleration and memory saving. As a result, we achieved 77ms on iPhone 6s and 251ms on Galaxy Note 3 for 101-class food recognition of a 224×224 image.

1. はじめに

近年，モバイル端末の高機能化により，高い演算能力を必要とするアプリの実行が可能となってきた．また，パターン認識分野では Deep Learning と呼ばれる多層のニューラルネットワークを用いた機械学習の手法の一つが，従来手法より極めて高い性能を見せるなど大きな注目を集めている．Deep Learning の応用としてモバイルデバイス上での画像認識が考えられるが，モバイル上に実装するには計算量がボトルネックとなる．その多くの計算量は各層の中でも，特に，畳込み層における畳込み演算の計算量が大部分を占めることから，畳込み演算を高速化することが，モバイル上で Deep Learning を実装する上で重要となる．また，モバイル特有の問題としてメモリ容量に限りがあるなどの制約も発生する．

そこで，本研究ではモバイルに実装するために，高速化の工夫と省メモリ化により，スマートフォン上での高速・高精度な画像認識システムを実現した．図 1 に本研究で開発したアプリのデモ画面を示す．

2. 関連研究

2.1 DCNN の高速化の研究

DCNN をリソースが制限されるモバイルコンピューティング上で使うために DCNN の高速化は必須であり，この分野は盛んに研究が行われている．例えば，処理に大部分の時間を要する畳込み層の計算の工夫に関する研究として [1]，[2]，[3] などがある．また，学習時のパラメーターにかかる重みに着目した高速化の研究として [4]，[5] などがある．

連絡先: 丹野良介，電気通信大学大学院 情報理工学研究科 情報学専攻，tanno-r@mm.inf.uec.ac.jp

図 1: システムのデモ画面

2.1.1 畳込み層の計算の工夫に関する研究

Gong ら [1] は Deep Learning をモバイル端末でも利用できるように，Vector 量子化によるパラメーター圧縮を行っている．1000 種類カテゴリ分類で，僅か 1%の損失に抑えて 16〜24 倍の圧縮を達成する研究成果を残している．

Liu ら [2] はスパース分解による DNN のパラメーター圧縮を行っている．また，CPU 上での効率的なスパース行列演算アルゴリズムとして Sparse Convolutional Neural Networks(SCNN) を提案し，物体検出に SCNN モデルを適用することで大幅な高速化を実現している．

Jaderberg ら [3] は低ランク近似やフィルタ近似で，文字認識タスクにおいて精度低下 1%未満で 4.5 倍の高速化を実現している．

2.1.2 学習時のパラメーター重みに着目した高速化の研究

Courbariaux ら [4] は順方向及び逆方向伝搬時における重みを −1 or 1 に二値化することで，本来必要な演算の 2/3 を除去し，トレーニング時間を 3 倍にするという新しい高速化手法として “BinaryConnect” を提案している．

Lin ら [5] は畳込み層の乗算をビットシフトに置き換える手法として “quantized back propagation” を提案し，先行研究である “BinaryConnect” を上回る性能を見せた．

2.2 物体認識アプリケーションの研究

本研究では画像認識システムとして，モバイル食事画像認識システムを開発した．本研究のような食事画像の認識に関する研究としては [6] がある．

河野の研究 [6] では認識手法に HOG 特徴量と色特徴の 2 つの局所特徴量を Fisher Vector で表現し，線形 SVM で分類する認識エンジンでスマートフォン上でのリアルタイム高精度物体認識システムを実現している．図 2 が「FoodCam」であり，食事画像を認識して，カロリー付きの結果を表示する機能や，食事の量を右下のスライダーで調整することでカロリーの量を調整することもできる．また，食事記録をシステム内に登録することで，食事管理支援の機能も備わっている．

本研究では，河野 [6] の認識エンジンを Deep Learning に変更し，畳込み層の高速化の工夫により，従来手法である Fisher Vector+ 線形 SVM による認識エンジンと同程度の認識速度で，より高精度な物体認識を実現している．認識性能については後述する．

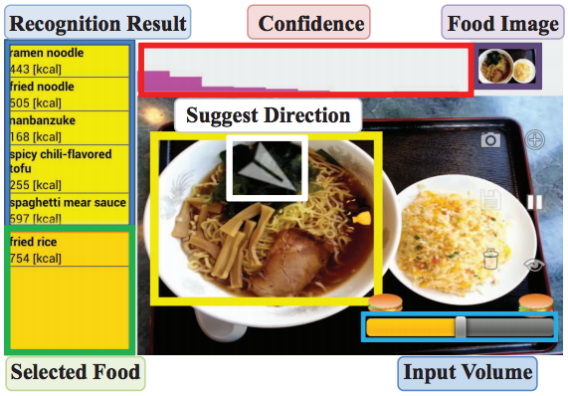

図 2: システム構成画面

3. 画像認識システム

本アプリの主な特徴として次が挙げられる．

- 認識はアプリ内完結 (サーバ不要)
- マルチスレッド，高速フレームワーク利用による高速化
- マルチスケールによる任意画像サイズに対応
- 必要メモリの大幅な削減

また，101 種類認識については，標準的な 224×224 の画像を入力とした場合，

- 認識時間は 80ms 台の高速認識
- 上位 5 位以内で 93.5%の高精度認識

という特徴がある．そこで本章では認識システムに用いている認識エンジンの説明を行う．

3.1 DCNN の学習

DCNN には一般的には AlexNet [7] を用いるが，モバイルに実装するにはアプリ容量に限りがあるなど問題が生じる．そこで，全結合層をもたない Network-In-Network(NIN) [8] のモデルを認識エンジンに利用している．

これにより AlexNet では約 6000 万もの大規模パラメーター数が必要だったところを，NIN を利用することで約 750 万のパラメーター数で済むなど，約 87.5%のパラメーター数の圧縮が可能である．NIN のモデルを利用することで大幅なパラメーター数の削減を実現しているが，認識性能については，1000 種類認識において AlexNet と同程度の認識性能を維持しており，モバイル実装を考慮すると，NIN のネットワークアーキテクチャを利用することが妥当であると考えられる．

NIN のモデルを利用して，ILSVRC1000 種類と食事に関連した 1000 種類の ImageNet 画像 2000 クラス，計 210 万枚で pre-train を行い，そのモデルを UEC-FOOD100 の食事 100 クラスと，主に Twitter から収集した非食事画像 1 万枚，計 101 クラスで fine-tuning している．学習には最も有名な Deep Learning Framework である Caffe [9] を用いた．

3.2 高速化手法

3.2.1 Im2col

画像の 3 次元配列を行列のような 2 次元配列に変換することができれば，畳込み演算が行列積の計算に落としこむことができる．それを可能にするのが im2col である．im2col とは image-to-column の略であり，図 3[*1] にあるように，画像に畳込むフィルタのカーネルサイズと同じ大きさのパッチに画像を切り分け，画像のパッチを列行列に変換する操作のことである．この操作により，畳込み層における畳込み演算を図 4[*2] の行列積での計算に変換することができる．

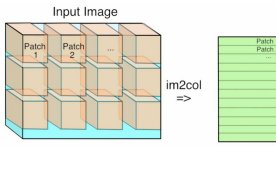

図 3: im2col

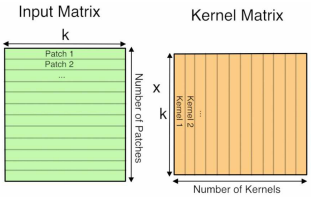

図 4: 畳込み演算の行列積

3.2.2 Basic Linear Algebra Subprograms(BLAS)

im2col 操作により Deep Learning における畳込み層の畳込み演算を，行列積で計算可能な形に変換した後は，行列積をいかに

*1 http://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/ から引用

*2 http://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/ から引用

高速に計算するかが高速化の鍵となる．DeepFoodCam(BLAS)ではこの行列積の計算に，線形代数演算ライブラリである BLAS の GEneric Matrix Multiplication(GEMM) 関数を用いている．

BLAS とは線形代数演算で用いられる基本的な演算を API 化したものであり，BLAS の演算性能によって以下の 3 つの演算に分類される．

Level1 BLAS

ベクトルの内積やベクトルの定数倍の加算などの演算を行う関数群である．

$$\boldsymbol{y} \leftarrow \alpha\boldsymbol{x} + \boldsymbol{y} \tag{1}$$

Level2 BLAS

行列とベクトルの積などの演算を行う関数群である．

$$\boldsymbol{y} \leftarrow \alpha\boldsymbol{A}\boldsymbol{x} + \beta\boldsymbol{y} \tag{2}$$

Level3 BLAS

行列と行列の積などの演算を行う関数群である．

$$\boldsymbol{C} \leftarrow \alpha\boldsymbol{A}\boldsymbol{B} + \beta\boldsymbol{C} \tag{3}$$

iOS では BLAS の実装としてデバイスに高度に最適化された Accelerate Framework を利用した．このフレームワークは内部で GPU を利用している可能性があるが，詳細は Apple が公表していないため不明である．一方，Android では BLAS の実装として OpenBLAS を利用している．両者の違いは実装先のアーキテクチャに BLAS が最適化されているか否かにある．

3.2.3 NEON

NEON とは ARM プロセッサの Single Instruction Multiple Data(SIMD) 命令セットであり，一の命令で複数の処理を可能にするベクトル処理機構のことである．畳込み演算を主に以下の式 4 及び式 5 の NEON 命令を使って高速化している．

ベクタ乗算

2 つのベクタの対応する要素を乗算して，デスティネーションベクタに結果を返す NEON 命令である．

$$V_r[i] := V_a[i] \times V_b[i] \tag{4}$$

ベクタ積和

2 つのベクタの対応する要素を乗算して、結果をデスティネーションベクタの要素に累積する NEON 命令である．

$$V_r[i] := V_a[i] + V_b[i] \times V_c[i] \tag{5}$$

式 4 及び式 5 の NEON 命令により，同時に 4 つの 32bit 単精度浮動小数点を演算させることができる．また，マルチプロセッサにより，iPhone 6s では 2 コア同時実行の合計 8 演算を同時に実行することができる．Android ではコア数が QuadCore の 4 コアであることが一般的なので，合計 16 演算を同時実行でき，畳込み演算を高速化している．

4. 実験

モバイル上に実装した画像認識システムを用いて画像認識時間の計測実験を行い，認識エンジンの認識時間の評価を行った．また，認識エンジンの認識精度も評価した．

4.1 実験の設定

BLAS 及び NEON により高速化を図った認識エンジンを iOS 及び Android 両デバイス上に実装し，認識時間の計測を行った．実験の設定を一覧表示したものを表 1 として以下に示す．尚，本実験では認識対象を食事 101 種類とした．

表 1: 認識時間の計測実験の設定一覧

実験番号	フレームワーク
(1)	iOS(BLAS)
(2)	iOS(NEON)
(3)	Android(BLAS)
(4)	Android(NEON)

4.2 評価用デバイス

認識時間の計測には次の 2 台を使用した．

- iPhone 6s(CPU A9 1.84GHz RAM2GB DualCore)
- GALAXY Note 3(2.3GHz RAM3GB QuadCore Android5.0)

4.3 実験結果

iPhone 6s 及び GALAXY Note 3 を用いて，表 1 にある各実験の設定毎に計測を 20 回行い，平均の認識時間 [ms] を求めた結果を表 2 として以下に示す．表 1 から iPhone 6s 上では認識時間 78ms と高速に認識出来ていることがわかる．一方，Android では BLAS の方が NEON 命令よりも遅いという結果が得られた．これは BLAS が最適化されているか否かが原因であると考えられる．

表 2: 実験結果

実験番号	認識時間	フレームワーク
(1)	78ms	iOS(BLAS)
(2)	255ms	iOS(NEON)
(3)	1652ms	Android(BLAS)
(4)	251ms	Android(NEON)

4.4 認識性能

表 3 に認識エンジンの認識性能を示す．

表 3: 食事 101 種類と一般物体 2000 種類の認識性能

認識対象	1 位	5 位以内
食事 101 種類	74.5%	93.5%
一般物体 2000 種類	39.8%	65.0%

5. おわりに

5.1 まとめ

Deep Learning をモバイルに実装する上でボトルネックとなるのが計算量であり，モバイルに実装されているような CPU では Deep Learning に要する計算量を処理するのに時間がかかりすぎてしまう．その多くの計算量は各層の中でも特に畳込み層における畳込み演算の計算量が大部分を占めることから，畳込み演算を高速化することが，モバイルデバイス上で Deep Learning による画像認識システムを構築する上で重要となっている．

そこで本研究では，モバイルデバイス上で Deep Learning による画像認識システムを構築し，画像認識にかかる時間を実験により評価した．実験の結果，iOS では BLAS を使って高速化した認識エンジンが iPhone 6s 上で平均 78ms という認識時間の結果が得られた．一方，Android では NEON 命令を用いた高速化認識エンジンが GALAXY Note 3 上で平均 251ms という結果が得られた．iOS の場合，BLAS の実装に用いたフレームワークがデバイスに最適化されていること，また，内部で GPU を利用している可能性があることから，iOS の BLAS 実装が Android と比較して高速な結果が得られた理由であると考えられる．

今回，実装として入力画像サイズを 224×224 で固定した画像を入力としているが，入力画像サイズを小さくすることで，さらに高速化することも可能である．

5.2 今後の課題

現在，普及しているスマートフォンなどのモバイルデバイス上には GPU が搭載されていることが一般的となっている．GPU は CPU よりもベクトル演算や並列処理に適していることから，様々な汎用演算を高速化するために利用されている．その為，今後は CPU だけではなく，モバイル GPU も利用した CPU と GPU による並列演算を行うことで認識エンジンの更なる高速化をしたいと考えている．何故なら，単に物体を認識するのみならず，領域分割 (セグメンテーション) や動画像のリアルタイム画像処理など，より複雑な処理をシステムに実装するには更なる高速化が必要不可欠となってくるためである．また，現段階では学習モデルに Network In Network(NIN) を採用しているが，更に認識エンジンを高速化することができれば，GoogLeNet や VGG など，NIN よりも複雑な学習モデルを利用することができ，更なる精度向上も期待できる．よって，認識エンジンの更なる高速化は必須であると考える．

認識エンジンの応用も重要であり，本研究で開発した画像認識システムはモバイル端末内で認識は完結していることから，サーバが不要という特徴がある．よって，各種 IT 機器への組込みも用意であることから，様々な応用が可能である．また，本システムの発展形として，スマートフォン上でのカロリー量推定が考えられる．今後は，2 次元画像から 3 次元形状を復元することにより，食べ物のカロリー量の推定に関する研究も課題としたい．

参考文献

[1] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. In *Proc. of International Conference on Learning Representations*, 2015.

[2] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2015.

[3] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proc. of arXiv: 1405.3866*, 2014.

[4] M. Courbariaux, Y. Bengio, and J. P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 2015.

[5] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio. Neural networks with few multiplications. In *Proc. of arXiv:1510.03009*, 2015.

[6] Y. Kawano and K. Yanai. Real-time mobile food recognition system. In *Proc. of CVPR International Workshop on Mobile Vision (IWMV)*, 2013.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[8] M. Lin, Q. Chen, and S. Yan. Network in network. In *Proc. of International Conference on Learning Representations*, 2014.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. of arXiv:1408.5093*, 2014. `http://caffe.berkeleyvision.org/`.