DeepXCam: CNNによるリアルタイム モバイル画像認識・変換アプリ群

丹野 良介[1,a)]　柳井 啓司[1,b)]

1. はじめに

CNN の応用としてモバイル上での画像認識が考えられるが，高演算性及びメモリ容量などがボトルネックとなり実装が困難であった．しかしながら，モバイルデバイスの高性能化に伴い高い演算能力を必要とするアプリの実行が可能なってきたこと，また，高速化の工夫及び省メモリ化により Deep Learning を用いたリアルタイム画像認識システムがモバイル上でも実現できうる段階まできた．

そこで，本デモでは，モバイルデバイス上で CNN を用いたリアルタイム画像認識・変換アプリの実演を行う．画像認識については，データセットさえあればどんな物でも高速高精度に認識出来うる認識システムを紹介する．また，変換アプリについては，“Style-Transfer” という画像の画風を変換する CNN の応用をモバイル上で実現したシステムを紹介する．

2. 基本認識 CNN アーキテクチャ

物体認識で用いる基本的な CNN アーキテクチャは，一般的には AlexNet[7]，Network-In-Network(NIN)[8]，GoogLeNet[11]，VGG-16[5] を用いる．しかし，AlexNet 及び VGG-16 は冗長なパラメータをもつ全結合層を 3 層含むネットワークであるなど，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 NIN のモデルを認識エンジンに利用している．また，GoogLeNet は 5x5, 3x3, 1x1 の畳込み層，及び，プーリング層から成る “Inception modules” という複雑な CNN アーキテクチャで構成されていることから，モバイルデバイスに効果的なパラレル実装が難しいことから，今回は NIN を採用した (図 1)．

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 CNN 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．

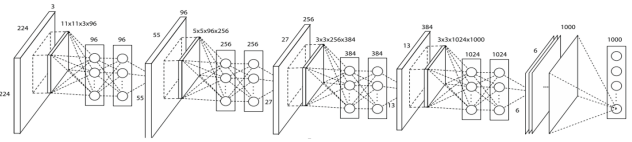

図 1 Network-in-Network. ([8] から引用). 全結合がなく，畳み込み層のみで構成されている．実際の実装では，batch normalization [2] をすべてのレイヤーの ReLU 関数の直前に追加している．

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，そのモデルを各認識対象データセットで fine-tuning している．学習には Caffe[3] を用いた．

Caffe で学習したモデルは，独自に開発した C 言語コードジェネレータ Caffe2C によって，iOS/Android 環境で動作可能な C 言語コードに変換し，それをモバイル CNN 画像認識エンジンとして利用した．

3. 画像認識アプリ

本章では，画像認識アプリについて，各アプリの個別説明を行う．DeepFoodCam のみ Android/iOS 両方，他は iOS のみの実装である．

3.1 DeepFoodCam Ver.2

非食事食事 100 種類に非食事を追加した 101 種類食事認識アプリである．アプリによる認識例を図 2 として以下に示す．学習には UEC-FOOD100[9] データセットの食事画像 1 万枚及び Twitter から収集した非食事画像 1 万枚の合計 2 万枚で学習した．また，認識率を表 3 として以下に示す．

DeepFoodCam Ver.2 では岡元らの研究 [13] で開発した DeepFoodCam の認識エンジン部分に改良を行っている．主な改良点は，BLAS/NEON の利用による認識速度の高速化である．以前のバージョンでは 1 回の認識に 800ms 程度掛かっていたが，表 1 に示すように大幅な高速化を実現した．本アプリのみ Android 版と iOS 版の両方が存在し，

[1] 電気通信大学 大学院情報理工学研究科 情報学専攻
[a)] tanno-r@mm.inf.uec.ac.jp
[b)] yanai@mm.inf.uec.ac.jp

他のアプリは iOS 版のみを実装している．

CNN の演算では，行列積 (Generic Matrix Multiplication, GEMM) が多用されるため，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．一方，Android では BLAS の実装として OpenBLAS を利用している．

さらに Android では OpenBLAS が高速化に役に立たなかったため，NEON 命令を用いた独自の GEMM ルーチンも作成した．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．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，iOS 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することができる．各認識時間を表 1 として以下に示す．また，ネットワークに NIN を用いることで，マルチスケールによる任意画像サイズを入力画像とすることができ，認識時間を大幅に減少させることが可能である．表 2 として結果を示す．

表 1 認識時間 [ms]

	NEON	BLAS	デバイス	BLAS 実装
iOS	255	78	iPhone SE	Accelerate
Android	251	1652	GALAXY Note 3	OpenBLAS

表 2 任意画像サイズの入力による認識時間 [ms] の変化．160x160 に縮小した場合の top-1 での認識精度低下は 10%以内，top-5 以内での認識精度低下は 5%以内．

入力画像サイズ	227x227	200x200	180x180	160x160
iPad Pro	66.6	49.7	44	32.6
iPhone SE	77.6	56	50.2	37.2

表 3 食事 101 種類の認識率

認識対象	top-1	top-5
食事 101 種類	78.8%	95.2%

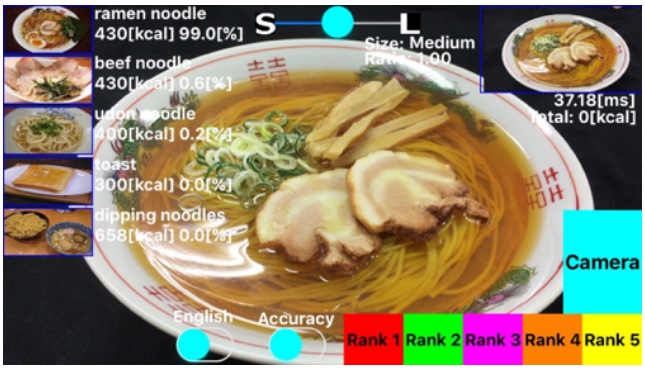

図 2 “Ramen” の認識画面

3.2 Deep2000Cam

ILSVRC1000 種類と食事に関連した 1000 種類の ImageNet 画像 2000 クラスを認識可能なアプリである．アプリによる認識例を図 3 として以下に示す．また，認識率を表 4 として以下に示す．

表 4 一般物体 2000 種類の認識率

認識対象	top-1	top-5
一般物体 2000 種類	39.8%	65.0%

図 3 “Keyboard” の認識画面

3.3 DeepBirdCam

鳥 200 種類を認識可能なアプリである．アプリによる認識例を図 4 として以下に示す．学習には California Institute of Technology が提供する Caltech-UCSD Birds 200 データセット [12] の 6033 枚を用いた．テストには各画像の 25%を用いた．また，認識率を表 5 として以下に示す．

表 5 鳥 200 種類の認識率

認識対象	top-1	top-5
鳥 200 種類	55.8%	80.2%

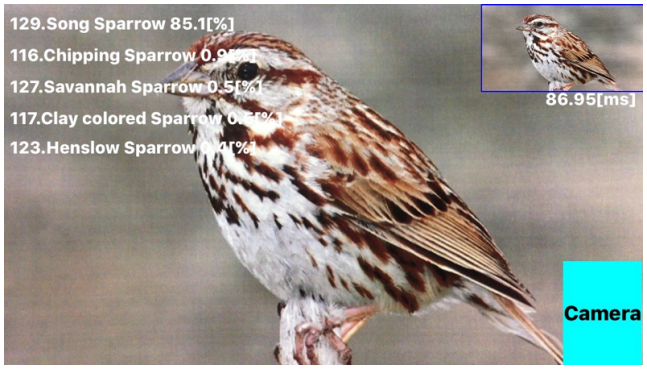

図 4 “Song sparrow” の認識画面

3.4 DeepDogCam

犬 100 種類を認識可能なアプリである．アプリによる認識例を図 5 として以下に示す．学習には 1 クラスあたり 150 枚以上の画像が存在する Stanford Dogs Dataset[6] を用いた．テストには各画像の 25%を用いた．また，認識率を表 6 として以下に示す．

表 6 犬 100 種類の認識率

認識対象	top-1	top-5
犬 100 種類種類	69.0%	91.6%

図 5 “Maltese dog” の認識画面

3.5 DeepFlowerCam

花 17 種類を認識可能なアプリである．アプリによる認識例を図 6 として以下に示す．学習には 1 クラスあたり 80 枚の画像が存在する University of Oxford が提供する 17 Category Flower Dataset[10] を用いた．テストには各画像の 25%を用いた．また，認識率を表 7 として以下に示す．

表 7 花 17 種類の認識率

認識対象	top-1	top-5
花 17 種類	93.5%	99.1%

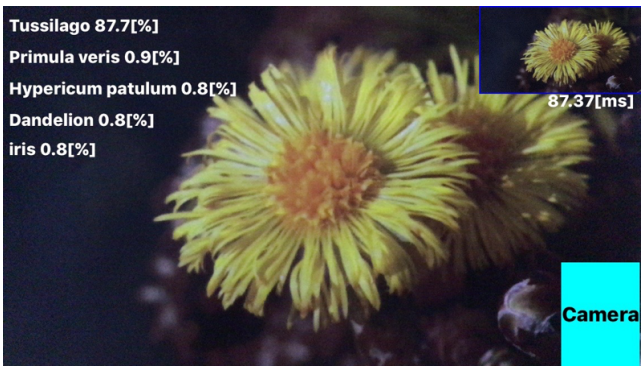

図 6 “Tussilago” の認識画面

3.6 DeepOmeletCam

図 7 にあるオムレツ 5 種類を認識可能なアプリである．アプリによる認識例を図 8 として以下に示す．学習には Web から収集した画像を用い，「文字」「絵」「模様」「ソース」は各 1 万枚，「プレーン」は 2400 枚を学習画像とし，テストには各画像 1500 枚 (プレーンのみ 171 枚) を用いた．また，認識率を表 8 として以下に示す．

図 7 オムライス認識の内訳．(上段左: 絵, 上段右: ソース, 下段左: プレーン, 下段真中: 模様, 下段右: 文字)

表 8 オムライス 5 種類の認識率

認識対象	top-1	top-5
オムライス 5 種類	84.4%	-

図 8 “絵付きオムレツ” の認識画面

4. 画像変換アプリ: DeepTransCam

画像のコンテンツ (形状) を保持したまま，スタイル (テクスチャ) のみを変化させることによって，例えば写真を絵画風に変換することができる Neural Style Transfer (スタイル転送) 手法 [1] が最近，注文を集めている (図 9)．この手法は，back-propagation を繰り返し行って変換された画像の生成を行うため，1 枚の画像変換に数分間程度掛かるという難点があった．それに対して，事前に特定のスタイルの変換を feed-forward で行う CNN を学習しておくことで，ほぼリアルタイムで画像を変換可能にする手法が提案された [4]．ただし，この手法では，事前に学習した単一のスタイルのみを転送することしかできなかった．

図 9 Style transfer の例.

そこで，本研究では，ネットワークを拡張し，複数種類のスタイルを任意の重みで合成して，リアルタイムで合成したスタイルを転送する手法を新たに考案し，それをモバイルアプリとして実装した．図 9 に示す例は，13 種類のスタイルの重みをランダムで決め，スタイル転送を 8 回行った例である．

スタイル転送では，画像から画像を生成するため，認識用の NIN とは異なる，図 10 に示す style transfer network を利用した．Convolutional layer のみで構成されるため，入力画像の大きさは任意である．一般には，最後の 3 レイヤーは convolutional layer の逆演算を行う deconvolutional layer を用いることが多いが，ここではモバイルでの高速 GEMM 演算をそのまま利用できるように，図 11 に示す unpooling を行って feature maps を縦横それぞれ 2 倍に拡大してから convolution を行うことで，stride 0.5 の convolution を実現し，deconvolutional layer の代わりとした．一般には，これらの操作は等価であるとされている．

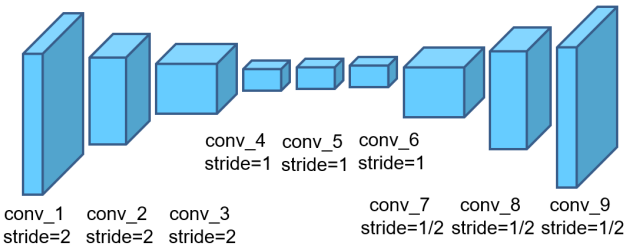

図 10 Transfer network. Super resolution や coloring で用いられるネットワークと基本的に同じである.

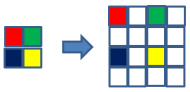

図 11 Unpooling 操作.

参考文献

[1] Gatys, L. A., Ecker, A. S. and Bethge, M.: A Neural Algorithm of Artistic Style, *arXiv:1508.06576* (2015).

[2] Ioffe, S. and Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, pp. 448–456 (2015).

[3] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S. and Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding, *Proc. of arXiv:1408.5093* (2014).

[4] Johnson, J., Alahi, A. and Fei-Fei, L.: Perceptual Losses for Real-Time Style Transfer and Super-Resolution, *arXiv:1603.08155* (2015).

[5] K. Simonyan, A. Vedaldi, A. Z.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, *Proc. of International Conference on Learning Representations* (2014).

[6] Khosla, A., Jayadevaprakash, N., Yao, B. and Fei-Fei, L.: Novel Dataset for Fine-Grained Image Categorization, *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition"* (2011).

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

[8] Lin, M., Chen, Q. and Yan, S.: Network In Network, *Proc. of International Conference on Learning Represenation Conference Track* (2014).

[9] Matsuda, Y., Hoashi, H. and Yanai, K.: Recognition of Multiple-Food Images by Detecting Candidate Regions, *Proc. of IEEE International Conference on Multimedia and Expo (ICME)* (2012).

[10] Nilsback, M.-E. and Zisserman, A.: Automated Flower Classification over a Large Number of Classes, *Proc. of the Indian Conference on Computer Vision, Graphics and Image Processing* (2008).

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A.: Going deeper with convolutions, *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 1–9 (2015).

[12] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S. and Perona, P.: Caltech-UCSD Birds 200, Technical report, California Institute of Technology (2010).

[13] 岡元晃一，柳井啓司：DeepFoodCam: DCNN による 101 種類食事認識アプリ，画像の認識・理解シンポジウム (MIRU) (2015).