# DeepFoodCam: A DCNN-based Real-time Mobile Food Recognition System

Ryosuke Tanno, Koichi Okamoto, Keiji Yanai The University of Electro-communications, Tokyo

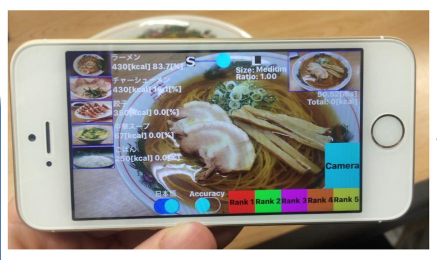
## 1. Objective

***Features of DeepFoodCam***

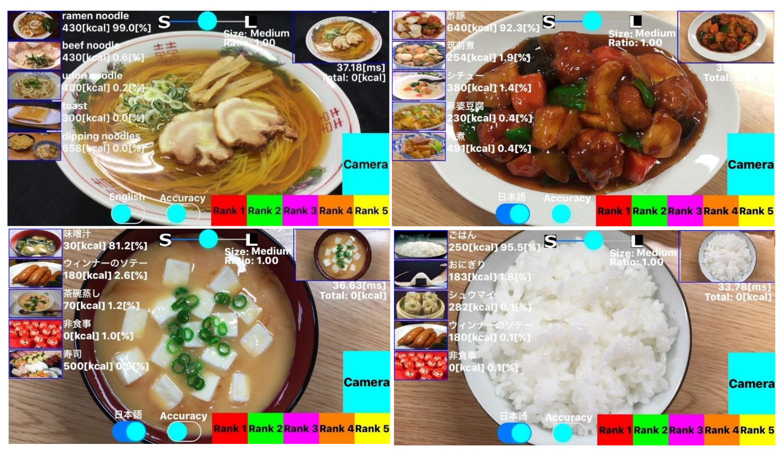
- all processing complete within iPhone (server not required)
- speeding up by multi-threading and fast framework
- recognizing any size of images by multi-scale CNN
- significant reduction in memory
- built-in easy to various mobile devices

***Example: 101 class recognition***

- recognition time: 26.2ms(iPhone7Plus)
- top-5 accuracy: 91.5%



***Holding your smartphone to the target !!***



## 2. Proposal Contents

***Anyone can make high-speed, high-precision object recognition and conversion iOS app***

～Flow of Making Mobile App～

1. Prepare a training image data
2. Train a CNN model by Caffe (or DIGITS)
3. Generate a C source code by Caffe2C automatically
4. Prepare a GUI code of mobile app
5. Generate CNN-based image recognition app by compiling the generated C code and the GUI code

Developed in our Lab

- **Caffe2C / Chainer2C**
  - converter to convert the parameter files to the C language code that can run on the mobile devices
- **Fast recognition (conversion) engine**
  - speeding up by multi-threading and fast framework
  - any image size corresponding by multi-scale
  - adjust the trade-off between accuracy and processing time by changing image size

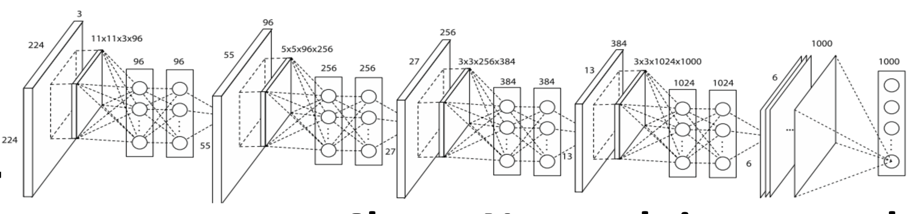
***If there is even training data, you can be created in any recognition app!!***

## 3. Recognition Engine

- ***Training DCNN***
  - We use **Network-In-Network(NIN)**[2] considering mobile implementation
  - Less parameters than Alexnet

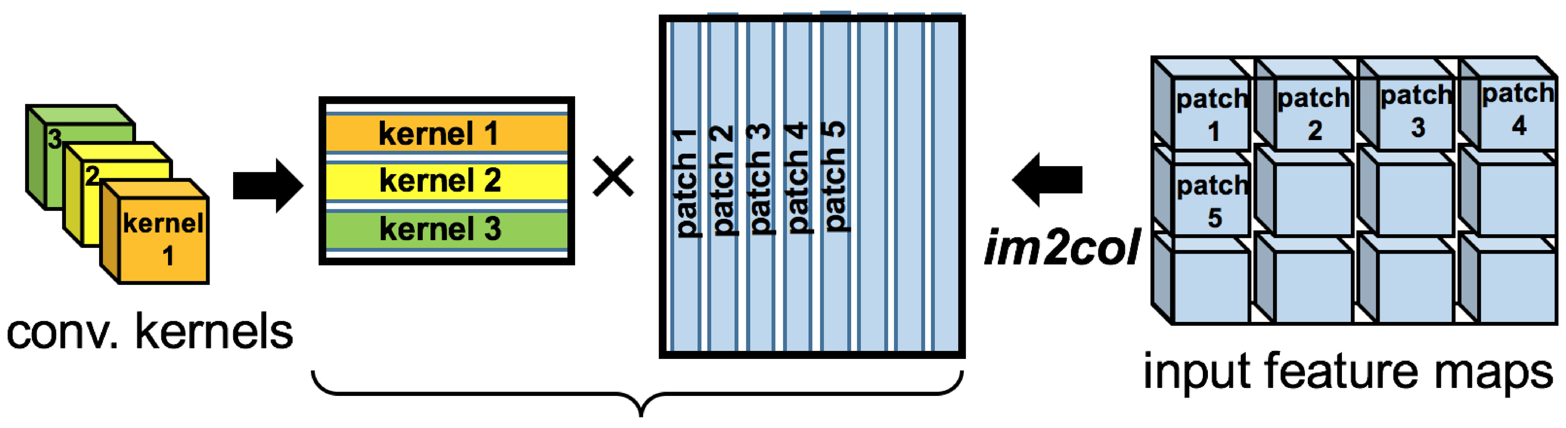
| | Param | Memory | Top-5 |
|---|---|---|---|
| AlexNet | 62Million | 248MB | 95.1% |
| NIN(4L+BN) | 5.5Million | 22MB | 95.2% |
| NIN(5L+BN) | 15.8Million | 63MB | 96.2% |

***Network In Network*** [2]
- only Conv layers
- no FC layers
- relatively smaller than the other architectures
- any image size correspondence



**4layer Network-in-network**

- ***Pre-trained CNNs with ImageNet 2000 category images***(totally 2.1 million images)
- ***Speeding up Conv layers ⇒ Speeding up GEMM***
  - Computation of conv layers is decomposed into "im2col" operation and matrix multiplications
  - BLAS(iOS: Accelerate Framework, Android: OpenBLAS)
  - We use the NEON instruction set which can execute four multiplications and accumulating calculations at once.
  - iOS: 2Core*4 = 8 calculation, Android: 4Core*4 = 16 calculationzz

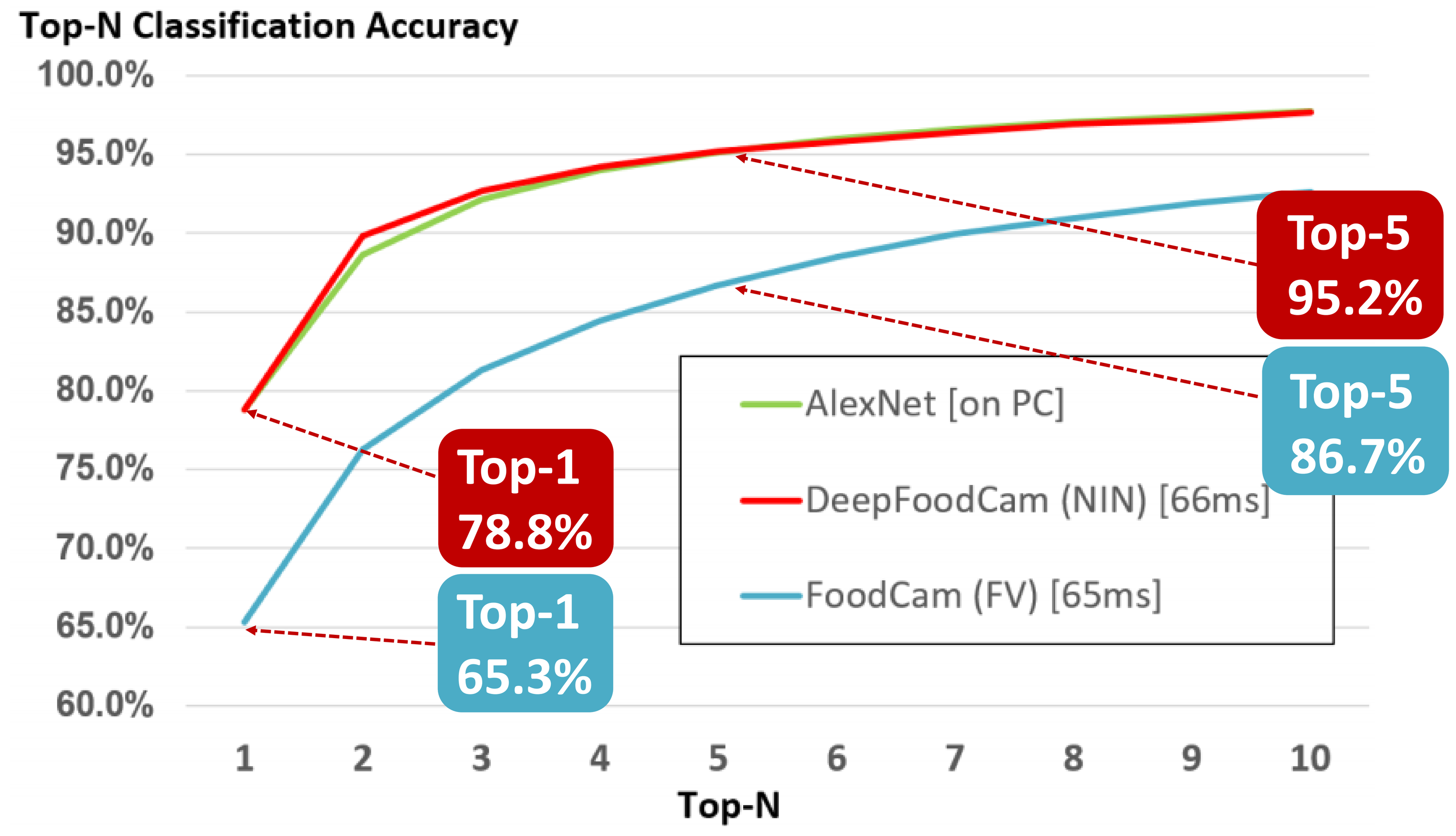



***GEMM: generic matrix multiplication*** (=conv. layer computation)

## 4. Accuracy and Recognition Time

- ***We use multi-scale network-in-networks (NIN)[2]***
  - Users can adjust the trade-off between recognition time and accuracy.
- ***We implemented multi-threaded mobile applications on both iOS and Android***
  - Employing either NEON SIMD instructions or the BLAS library for fast computation of convolutional layers

***food 101 class recognition performance (5 - fold CV)***




***Trade-Off between Accuracy and Recognition Time***

| Input Image Size | 227x227 | 200x200 | 180x180 | 160x160 |
|---|---|---|---|---|
| iPhone 7 Plus | 55.7[ms] | 42.1[ms] | 35.5[ms] | 26.2[ms] |
| iPad Pro | 66.6[ms] | 49.7[ms] | 44.0[ms] | 32.6[ms] |
| iPhone SE | 77.6[ms] | 56.0[ms] | 50.2[ms] | 37.2[ms] |
| Accuracy (top-5) | 95.2% | 95.1% | 94.1% | 91.5% |

*We achieved **real real-time !!***

## 5. Characteristic analysis of iOS and Android

- ***We revealed that BLAS is better for iOS, while NEON is better for Android, and that reducing the size of an input image by resizing is very effective for speedup of DCNN-based recognition.***
  - Please refer to [1] about the details.

| | NIN(NEON) | NIN(BLAS) |
|---|---|---|
| iPad Pro | **66.0**[ms] | 221.4[ms] |
| iPhone SE | **79.9**[ms] | 251.8[ms] |
| Galaxy Note 3 | 1652[ms] | **251.1**[ms] |

*Optimal speeding-up approach is different in the iOS / Android*

## 6. Conclusion

- ***Stand-alone DCNN-based mobile image recognition***
  - No need of a recognition server and communication.
  - Built-it trained DCNN model with UECFOOD-100
  - Implemented as iOS/Android app.
  - Released as iOS app on Apple Store (Please search "**DeepFoodCam**") as Android app (APK) on **http://foodcam.mobi/**
- ***Excellent performance with reasonable speed and model size***
  - UECFOOD100 : 78.8% (top-1) 95.2% (top-5) in 66.6 [ms] with 5.5M weights (22MB)
  - Employing Network-in-Network
  - Adding batch normalization and additional layers
- ***Multi-scale recognition***
  - User can choose the balance between speed and accuracy
  - Ex. iPhone 7 Plus: 26.2[ms] for 160x160 images ⇔55.7[ms] for 227x227 images

## Multiple Style Transfer and Object Recognition App

***·Food Rec App (both iOS/Android)***

Our Project page

**http://foodcam.mobi/**

Please search "***DeepFoodCam***"

***·Multi Style Transfer (only iOS)***

Please search

***"RealTimeMultiStyleTransfer"***

## Reference

[1] K. Yanai, R. Tanno, and K. Okamoto.: Efficient Mobile Implementation of A CNN-based Object Recognition System, Proc. of ACM International Conference Multimedia, 2016.
[2] M. Lin, Q. Chen, and S. Yan.: Network In Network, Proc. of International Conference on Learning Represenation Conference Track, 2014.