FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks

anonymous

Abstract

We present a mobile application made to recognize the food items of multi-object meal from a single image in real-time, and then return the nutrition facts with a components and approximate amounts. Our work is organized in two parts. First, we build a deep convolutional neural network merging with YOLO, the state-of-the-art detection strategy, to achieve simultaneous multi-object recognition and localization with nearly 80% mean average precision. Second, we adapt our model into a mobile application with extending function for nutrition analysis. After inferring and decoding the model output in the app side, we present detection results with both bounding box and class label in either real-time or local mode. Our model is well-suited for mobile devices with negligible inference time and small memory requirements with deep learning algorithm.

1 Introduction

With people paying increasing attention to tracking meal items and nutrition contents in daily life for weight loss or medical purposes, self-managed food monitoring has become one of the most practical application of computer vision. However, most current mobile apps (CarbandMove, MyFitnessPal, FatSecret, etc) require manual data entry, which is tedious and time consuming. Cordeiro *et. al* [1] conducted a survey in food journaling in 2015. They find that even if 80% (117/141) food journalers reported archiving data with mobile apps, most users do not use such apps for very long.

To avoid manual data recording, some recent work [2] use computer vision techniques to reason about meals, but it only works in laboratory conditions with well-separated food items and the number of categories is small. Google proposed a system called Im2Calories [3] with a mobile app that can recognize a food object in an image, and then predict its nutritional content based on volume estimation. However, they rely on the menu database and restaurant database collected from a specific area, and can only recognize a single food object in an image. They generate segmentation mask instead a bounding box, but this part is presented only in laboratory environment with few categories.

The ideal system in food journaling is a real-time automatic system including localization, recognition, and precise volume estimation based on solely image input. However, even though there is a large number of prior work [4] [5] in this area, food journaling is still widely considered difficult. The challenges stem from causes such as: the vast range of intraclass variations, the components complexity in many foods, e.g. sandwich, and the huge amount of categories and regional specialty.

In this paper, we propose an approach, which utilizes several state-of-the-art deep learning techniques, trained to detect[1] multiple food items in real-time. Our system is tailored to run on a conventional mobile phone with built-in camera, and present with nutrition facts for each food item served in an unit amount.

We rely on our prior work producing a deep convolutional neural network (DCNN) model built upon Mobilenet [6], adapting with a state-of-the-art *one-stage* detection framework, YOLOv2 [7] to generate bounding box and class label simultaneously. The network structure is small and computationally inexpensive thanks to the leverage of depth-wise separable convolution [8] and YOLOv2 strategy. Next we fit our deep learning model into a mobile application by using the TensorFlow Java API [9], and decode our model output with short inference time with real-time user image input. Finally, we present detection results with the nutrition analysis in the app.

This paper is structured as follows. Section 2 explains in detail how depth-wise separable convolution achieves less computation cost and parameter reduction. Section 3 describes how we process the training, the DCNN architecture and the YOLO strategy. Section 4 illustrates the mobile application prototype with the user interface outline. Section 5 closes with a discussion and a summary of the future work.

2 Background

2.1 Depthwise Separable Convolution

Our DCNN structure is built upon MobileNet, which is designed for mobile and embedded vision applications. MobileNet is based on a streamlined architecture that utilizes depth-wise separable convolutions [8] to construct lightweight deep neural networks. A depth-wise separable convolution block factorizes a standard convolution into a depth-wise convolution and a 1x1 point-wise convolution. Compared to a standard

[1] Following common terminology, we denote by **detect** a combination of localize and recognize.

convolution layer that filters and combines input feature maps into a new set of outputs in one step, depthwise separable convolution splits this process into two layers, with a separate layers for filtering and combining.

By expressing standard convolution as a two-step process, we achieve substantial reduction in both the parameter count and the computation cost.

More precisely, in a standard convolution layer, we denote by D_F the spatial dimension of the input feature map, D_K the filter width and height (assumed to be square), M the number of input channels and N the number of output channels. We also assume `SAME` padding so that the output feature map has the same spatial dimensions as the input.

Then, the parameter number ratio between depthwise separable convolution and standard convolution is:

$$\frac{D_k \cdot D_k \cdot M + M \cdot N}{M \cdot N \cdot \ D_k \cdot D_k} = \frac{1}{N} + \frac{1}{D_k^2}$$

Likewise, the computation cost ratio between depthwise separable convolution and standard convolution:

$$\frac{D_k \cdot D_k \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet uses depth-wise separable convolutions with 3x3 filter size to require 8 to 9 times fewer computations than the standard convolutions, with only a small reduction in accuracy [6].

3 Food Detection

In this section, we first describe how to prepare the training instances, outline the principle of YOLO, then illustrate the overall DCNN architecture and present the mean average precision (mAP) result validated on UECFood100 [10] and UECFood256 [11] benchmarks. These two datasets contain around 40K images (mostly Japanese food items) in total with 100 and 256 categories, respectively.

3.1 Data Augmentation

Data augmentation represents the process of artificially increasing the number of training instances [12]. Previous research demonstrates that data augmentation can act as a regularizer in preventing overfitting in neural networks [13]. Our DCNN model aims to be applied in mobile devices with camera input in real-time, which can vary in illumination, viewpoints, backgrounds, etc. Thus we utilize several data augmentation methods to improve the robustness and alleviate overfitting.

In particular, half of the training instances are processed with one of the following data augmentation methods: blur, horizontal flip, Gaussian noise or color shift. Although the training time is increased by 5 to 6 times due to data augmentation, our model is more robust and less overfitting, which is critical and beneficial for real-time image processing.

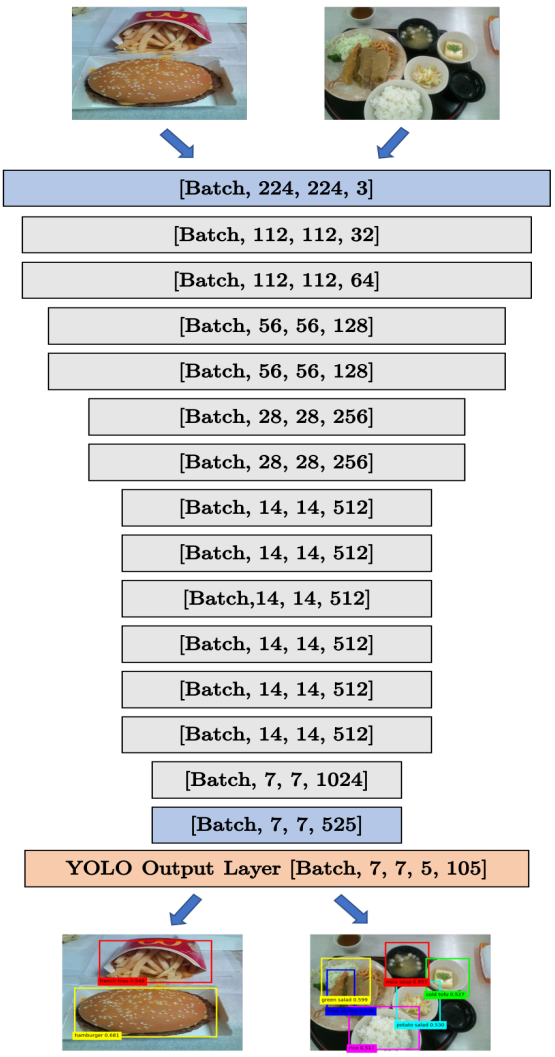


Figure 1: MobileNet-YOLO Architecture. Each grey triangle represents a convolution block. Blue triangle represents standard convolution layer. *Batch* is the number of training instances in each iteration.

3.2 YOLOv2 Strategy

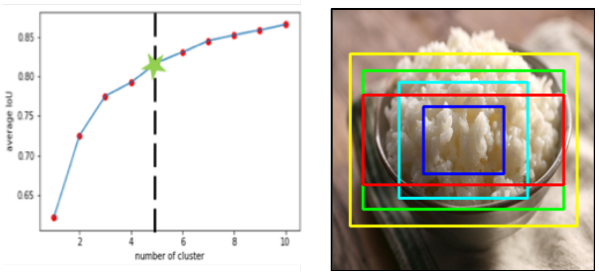
After downsizing the input resolution from (224, 224) to (7, 7) by 13 depth-wise separable convolution blocks in MobileNet, we add one more convolution layer reshaping the output size to be compatible with the YOLO output layer.

The main idea here is to divide each input image into (S, S) grid, and predict N bounding boxes in each cell with the shape $[b_x, b_y, t_w, t_h]$ for a bounding box along with a box confidence score for each box. In this case, t_w, t_h are offsets for the N prior anchor boxes generated

by clustering.

To find out the suitable number of clusters and the prior shapes of bounding boxes predicted in each cell, we use the k-means cluster algorithm [14] to generate prior anchors' shapes with respect to the cluster number under the Intersection over Union (IoU) metric. IoU is simply computed as the ratio of the area where the two boxes overlap over the total union area of the two boxes. We compute IoU for each KNN-generated anchor with the ground truth box, then we compute the average IoU under different cluster numbers from 1 to 10 to find the balance between number of anchors with model complexity as shown in Fig. 2(a).

Even if more anchors lead to higher average IoU, the complexity of model will also increase thereby giving rise to higher computation cost with a little benefit in performance. We find that k=5 gives a good trade-off for average IoU vs. model complexity, and we elaborate the five k-means generated prior anchors in Fig. 2(b).



(a) (b)

Figure 2: (a)K-means clustering box dimensions on UECFood100. (b)Prior anchors illustration in an example image.

3.3 DCNN Architecture

All convolution layers are followed by a batch-norm [15] and ReLU nonlinearity [16] with an exception of the final fully connected (FC) layer that has no nonlinearity, and feeds into a YOLO output layer. Counting depth-wise and point-wise convolutions as separate layers, our MobileNet-YOLO architecture has 30 layers with 3.5 million parameters in total. The overall scheme is shown in Fig. 1 with two example inputs with results. The input resolution for training is (224, 224).

For training, we use Adam [17] optimizer with $lr = 1e-4$, reduce factor $= 0.5$, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$, batch size $= 16$, with 11K iterations in total.

We evaluate our model on UECFood100 [10] and UECFood256 [11] sets with validation on around 10K food images. We achieve satisfying food detection performance with **76.36%** mAP on UECFood100 and **75.05%** on UECFood256 with `IoU=0.5` on validation set, hence our model is well-suited to be fit into mobile devices.

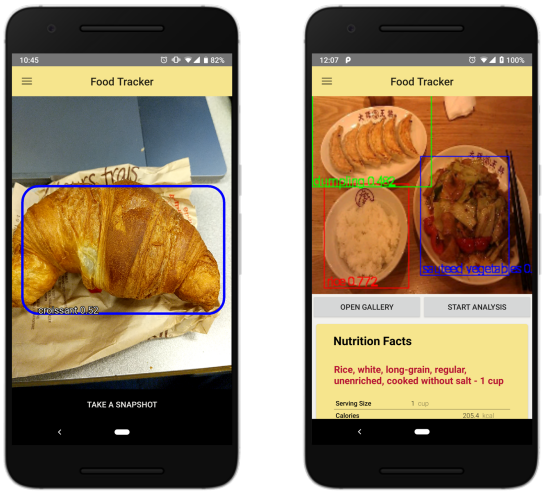


Figure 3: **Left:** real-time food detection example with results. **Right:** local mode food detection and nutrition analysis, user can select an picture for processing from local gallery.

4 Mobile Application

We convert our trained model from `.h5` file to `.pb` file so that our model can make prediction by using TensorFlow Java library [9].

For each inference, we decode the output and select the bounding box with confidence score higher than 0.4, locate boxes by b_x, b_y and adjust them based on the prior anchors calculated in Section 3.3 with the predicted offsets t_w, t_h. Moreover, we perform non-maximal suppression [18] to get rid of duplicate boxes by checking if there are overlapping with predicted boxes with more than 30% area.

We do not apply depth/volume estimation [19] in this work. Considering that our datasets contain mostly Asian food, which is usually served in a container (similar to western fast food), we assume, that the amount of each detected food item is one serving. Our nutrition analysis function is inspired by NutriVision [20], which utilizes one of the biggest global nutrition databases, Nutritionix [21], to analyze the nutrition content. Nutritionix contains more than 700K food items ranging from common foods, restaurant dishes and grocery foods. The deficiency of NutriVision is it requires manual data entry. We call the Nutritionix API based on our trained DCNN model's output, and return the one serving food item nutrition facts on the user interface, as shown in Fig. 4.

There are two modes in our mobile app design as shown in the left and right side of Fig. 3. One is the real-time mode to fast infer the camera input per frame and display the detection result including bounding box, class label and confidence score in the screen.

Figure 4: Example of a multi-object nutrition analysis user interface based on the food detection results.

Second is the local mode where user can select a local image to do nutrition analysis along with food detection.

We do not rely on any remote server to do the computation required for image inference. The inference process is conducted solely in the mobile device. We test our mobile app on a Google Pixel 2 with Android 9.0 system and a OnePlus 5 with Android 8.0 system in both real-time and local mode. Thanks to our small neural network design and efficient detection strategy, our average wall clock time[2] per image is **75ms** and average CPU time[3] per image is **15ms** in real time (Table. 1), thereby there is almost no delay observed by users.

5 Discussion and Future Work

The process of automatically detecting food objects and extracting nutrition contents is very complicated, especially when it comes to real life application. In particular, it requires solving various problems, such as: fine-grained recognition to distinguish subtly different forms of food, instance segmentation and counting, mask generation, depth/volume estimation from a single image. The existing state-of-the-art work focuses specifically on one of the sub-problems of food detection with computer vision techniques [22] [23] [24]. However, they all focus on a single task with strict environmental conditions or external assistance so that they are still far away from the holy grail of the automated food journaling systems.

[2]Wall clock time is the actual time taken by a computer to complete a task. It is the sum of CPU time, I/O time, and the communication channel delay.

[3]CPU time measures only the time during which the processor is actively working on a certain task.

Table 1: App performance profile.

Inference time	CPU time	15ms
	Wall clock time	75ms
DCNN model size		8.1MB
Runtime memory		242.2MB

Mobile application development is one of the most promising areas because of wide popularity and usability of smartphones. On the other hand, the limitation of computational resources of mobile devices makes it difficult to apply deep learning techniques. As for food detection and nutrition analysis with computer vision, many research conducted in laboratory environment making use of the power with multiple graphic cards, fewer of them have been generalized to mobile devices, still among those even fewer have been made publicly available.

For all these aforementioned problems, our work has tackled some of these, but it is clear that there is much more work to do. We believe that multi-task learning in this area is essential as the process from image to nutrition is involved with many computer vision problems. Moreover, one of the shortage of current computer vision algorithms is most of them are supervised learning thus all require training set with ground truth annotations, which result in some limitations on the testing categories, especially for food-related issue with countless food categories around the world.

In research to follow, we will be working on mask generation with respect to volume estimation, and investigate how to better leverage deep learning into mobile devices taking the computational resource limitation into consideration. There are many details to handle and many interesting problems from the point of view of computer vision research. Nevertheless, we believe that even a partial solution to these problems could be of great value to the society. Further, we will release the mobile app shortly in Google Play.

References

[1] E. Thomaz E. Bales A. K. Jagannathan G. D. Abowd F. Cordeiro, D. Epstein and J. Fogarty. Barriers and negative nudges: Exploring challenges in food journaling. *SIGCHI*, 2015. 1

[2] Weiyu Zhang, Qian Yu, Behjat Siddiquie, Ajay Divakaran, and Harpreet Sawhney. snap-n-eat: Food recognition and nutrition estimation on a smartphone.

Journal of Diabetes Science and Technology, 9(3):525–533, 2015. PMID: 25901024. 1

[3] A. Myers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. Murphy. Im2calories: Towards an automated mobile vision food diary. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1233–1241, Dec 2015. 1

[4] Ziad Ahmad, Marc Bosch, Nitin Khanna, Deborah A. Kerr, Carol J. Boushey, Fengqing Zhu, and Edward J. Delp. A mobile food record for integrated dietary assessment. In *Proceedings of the 2Nd International Workshop on Multimedia Assisted Dietary Management*, pages 53–62, New York, NY, USA, 2016. ACM. 1

[5] C. J. Boushey, M. Spoden, F. M. Zhu, E. J. Delp, and D. A. Kerr. New mobile methods for dietary assessment: review of image-assisted and image-based dietary assessment methods. *Proceedings of the Nutrition Society*, 76(3):283294, 2017. 1

[6] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 1, 2

[7] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. 1

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. 1

[9] TensorFlow Java API. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/java. 1, 3

[10] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2012. 2, 3

[11] Y. Kawano and K. Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014. 2, 3

[12] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016. 2

[13] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003. 2

[14] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 3

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 3

[16] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress. 3

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 3

[18] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 3

[19] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *CoRR*, abs/1411.6387, 2014. 3

[20] NutriVision. https://play.google.com/store/apps/details?id=edu.ucuccs.nutrivision&hl=en. 3

[21] Nutritionix API. https://www.nutritionix.com/database. 3

[22] Ya Lu, Dario Allegra, Marios Anthimopoulos, Filippo Stanco, Giovanni Maria Farinella, and Stavroula Mougiakakou. A multi-task learning approach for meal assessment. In *Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management*, CEA/MADiMa '18, pages 46–52, New York, NY, USA, 2018. ACM. 4

[23] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-slice residual networks for food recognition. *CoRR*, abs/1612.06543, 2016. 4

[24] T. Ege and K. Yanai. Simultaneous estimation of food categories and calories with multi-task cnn. In *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pages 198–201, May 2017. 4