A Deep Learning Approach to Calorie Counting

Brendan Abraham, Jianyu Su
Department of Systems Engineering, University of Virginia, Charlottesville, VA 22904
[bea3ch, js9wv]@virginia.edu

Abstract

Manually logging food and calorie intake can be significantly time-consuming. In this paper, we present a far simpler approach to calorie tracking. We propose that calorie counts can be estimated from an image alone. That is, given a picture of a meal, our model can detect which foods are in the image and estimate the meal's aggregate calorie count. To test this hypothesis, we first trained a very deep convolutional networks for large-scale image recognition (VGG) [10] to classify over 221 different types of foods. We achieved over 75% MAP, demonstrating that food classification is highly effective with state of the art CNN architectures. Then, as a proof of concept, we trained a faster-rcnn type neural net on a 10-class food data-set. The model then performs a calorie lookup for its top class predictions and sums these counts to give a final calorie estimation. Our model achieved over 72% MAP for object detection and the calorie predictions were reasonably close to our test cases where total calorie counts were known. Our contributions to the field are twofold. First, we have created the first publicly available food-detection dataset that is adequately sized for deep learning. Second, we will show in future work that is is possible to predict calories by regressing on image features.

1. Introduction

Logging food and calorie intake has been shown to facilitate weight management [11]. Many smartphone apps allow the user to track calories by manually specifying the food types and portion sizes eaten at each meal. However, this process can be significantly time-consuming and cumbersome. We present a far simpler yet novel approach to calorie tracking in this paper. We believe that calorie counts can be estimated from an image alone. All the user needs to do is take a picture of their meal, and the system will return an estimated calorie count based on the food items it detects in the picture. This system possesses numerous advantages over manual calorie tracking. First, it is far more natural; between writing yelp reviews and snapchatting, photographing meals has become all too common for millennials and young adults. Second, it will drastically reduce the required time and burden of calorie-tracking. It is our hope that our system will promote users to be more conscious of their diets and ultimately become healthier eaters.

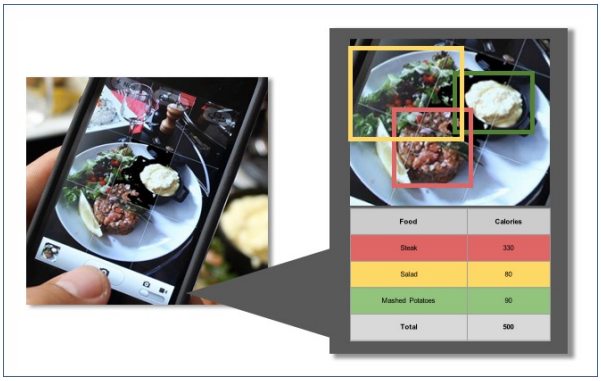


Figure 1: We envision a system where the user simply takes a photo of their meal and the app performs food detection and calorie prediction on this image

We formulate the problem in three stages that culminate in our ideal model. The first stage is image classification, in which we trained a VGG [10] via transfer learning to classify over 220 different types of foods. For training, we combined two large food datasets: food-101 and Vireo-172. For stage 2, we applied the knowledge learned from classification to train an object detector on a 10-class dataset with annotations that we labeled ourselves. This model's network can perform food classification and detection. However, we have to feed the predictions into a calorie lookup program that finds average calorie counts of the model's top labels and returns a weighted sum. Our calorie program scraped data from *myfitnesspall.com* and is described in detail below. The final stage of the project includes training the model to predict calorie counts by regressing over extracted image features. To our knowledge, this has never been done before. While we did not get this far in the project due to time constraints, we plan to pursue this avenue of research in the future.

2. Related Work

The field of object detection has exploded with innovation in the last five years, thanks to advances in deep learning. For a while, detection models plateaued in performance, and the only way to incerase performance was to drastically increase the complexity of these models. However, in 2014, that all changed with the advent of Girschick et al's R-CNN model, which bested the state of the art at the time by over 10% [3]. Given an input image, their model used Selective Search to generate 'box proposals' and fed these proposals into a convolutional neural network, performing classification on each proposal. While the model achieved impressive results, it was far too computationally expensive to be used in real time. [2] improved this architecture, naming their model Fast-RCNN. Unlike it's predecessor, the image only needed to pass through the CNN once, and it achieved similar accuracy. Now the selective search phase was the only remaining bottleneck in the model. In 2016, Ren et. al made a model that eliminated the need for selective search [8]. Creatively dubbed "Faster-RCNN", the model performed classifications of various shaped boxes within an NxN grid of the original image. This model achieves near-real time performance, performing object detection in a fraction of a second. Since then, a few other CNN based models have emerged such as YOLO (You Only Look Once), but we decided to use Faster-RCNN for this project.

While there have been major advances in object detection as a whole, very little work has been done in food detection and classification. Most work this domain has focused on classifying foods based on engineered features. Matsuda et al incorporate multiple visual features of food to detect food [7]. Similarly, Shroff et al extract features relating to the context, size, shape and texture of food images, and performed food detection on these features using a perceptron classifier [9]. However, the efficacy of these approaches relies on domain expertise and selection of features. That being said, a few researchers have taken deep learning approaches to food detection. Kawano et al feed deep convolutional features and visual features to a classifier to detect foods [6]. Similarly, Kagaya et al performed food detection with a deep convolutional neural network and compared their performance to traditional approaches using handcrafted features [5]. However, none of these works dealt with both food detection and calorie prediction.

The closest work to our proposal is a project by Microsoft research that predicts calorie counts of items on various restaurant menus [1]. In this paper, users photograph their meal at a restaurant, and the system maps them to menu items at that restaurant using both GPS lookups and image features. Then, the calorie count is found by performing an lookup in a calorie database. Our approach differs in numerous ways. First, they were only concerned with detecting items from certain restaurant menus; our method generalizes the problem to predict food types regardless of geographical location. Second, they used an SVM classifier and handcrafted features for food detection, while our model uses a CNN architecture. Finally, they did simple calorie lookups on pre-stored calories data with predicted labels. The main down sides of this approach are that the same dishes may differ substantially in calorie counts and the that the model cannot infer calories for unknown dishes. While we also perform calorie lookups in this paper as a proof of concept, our long term goal is to train a calorie regressor that learns to predict calories based on the extracted features from the CNN. While we also perform calorie lookups in this paper as a proof of concept, our long term goal is to train a calorie regressor that learns to predict calories based on the extracted features from the CNN. To our knowledge, we are the first to perform calorie-count regression on CNN feature maps.

3. Data Collection

The bulk of this project was spent gathering and curating the data for our models. For classification, we combined two previously available food datasets. For food detection, we manually labeled over 9,000 of these images, spanning 10 different foods. For calorie prediction, we scraped calorie data online for each of the 10 food categories. Each dataset is explained in detail below.

3.1. Classification Dataset

For classification, we combined Food-101 and Vireo-172, two previously available food datasets. Food-101 consists of 101 Western (European/American) foods and each class has 1000 samples. The Vireo-172 dataset was gathered by [4] and consists of 172 predominantly Chinese dishes. Each dish had anywhere from 200-1000 samples. After de-duplicating the datasets and removing overly-obscure foods (mainly from Vireo-172), we were left with 220 food classes and nearly 190,000 samples.

3.2. American Food Detection Dataset (AFDD)

While we had nearly 190k labeled food images, we needed annotations to perform object detection. However, as a two-man team, we had neither manpower nor resources to annotate such a behemoth of a dataset. Instead, we chose a subset of 10 foods from the combined data and decided to label those ourselves. We decided to use solely American foods because American dishes consist of a main course and sides which tend to be spread out and thus easy to detect. Most of the Chinese dishes were soup-based or consisted of mixed meats and vegetables. As such, they proved to be difficult for object detection. We used a free tool called Simple

Stage	Goal	Training Data	Model
1	Food Classification and Transfer Learning	Food-101 and Vireo-172	VGG16
2	Food Detection and Calorie look-up	AFDD	Faster-rcnn and Calorie Look-up
3	Food Detection and Calories prediction	Food and Calorie data	Our Proposed Model

Table 1: Summary of three stages

Image Annotator [1] to create the annotations for our images. The output of this process was a CSV file containing an image name, object id, and bounding box cordinaetes for each object we annotated. We then wrote a script to collapse and convert these annotations to an XML format that was compatible with the py-faster-rcnn architecture. In the end, we were left with 8,966 annotations, an average of 896 samples per class.

3.3. Calorie Dataset

Our final source of data was calorie counts for each food type. We got this data by crawling the myfitness.com [2] calorie database maintained by Under Armor. On this site, you can search for a food and it will return the most relevant list of food with nutritional info such as calories, serving size and grams of fat. We queried the database for each food category and extracted the top 500 samples. From these, we computed the mean and variance of calorie counts. We also computed these for the other nutrition metrics in case we decide to expand our model to predict multiple metrics. We used this data to form our calorie lookup table used in stage 2.

4. Models

4.1. VGG16

Architecture: VGG16 has 16 weight layers, including 13 convolutional layers and 3 fully connected layers. The input to this model is 224×224 pixel RGB images. The pixel values are normalized by subtracting the mean RGB value from each pixel, and some images are randomly flipped. The input image passes through convolutional layers, whose receptive filter is 3×3 pixels and stride and have a stride of 1. Each convolutional layer is followed by a ReLU layer to make its output non-linear. To achieve spatial pooling, max-pooling layers of 2×2 receptive filter and strides of 2 follow some of the Relu layers.

The output of this stack of convolutional layers will be fed into a stack of 3 fully-connected layers. The first two layers have 4096 channels while the last layer has 221 channels corresponding to 221 food categories. The first two layers are also followed by ReLU layers and dropout layers with dropout probability of 0.5 to overcome over-fitting.

Input Image Size: Images are re-scaled and cropped to 224×224 pixels. Specifically, we use random cropping to overcome over-fitting.

Loss Function: The loss function we used to train the model is cross entropy, which is given by:

$$\ell(y, \hat{y}) = -\sum_i y_i log(\hat{y}_i), \tag{1}$$

where y is the ground truth of form $\left[0,0,1,\ldots,0\right]$ and $\hat{y}_i$ is the predicted scores which is also a vector.

4.2. Faster-rcnn and Calorie Look-up

Faster-rcnn: Faster-rcnn uses the output of vgg16's convolutional layers as feature maps. This network contains a region proposal (RPN) module, which is a network having a 3×3 pixel receptive window. The RPN network slides on the feature map and outputs the coordinates of proposed region and its confidence in whether the region contains a object or not. The faster-rcnn network will clip the feature maps according to the top region proposals. Next, the clipped feature maps will be concatenated with its corresponding confidence scores and fed into a ROI pooling layer. Finally, the ROI pooling results pass through a stack of linear layers. The first two layers have 4096 channels and the outputs are coordinates of a proposed region with classification scores.

Input Image Size: Since the input and output layers are both fully-connected and convolutional, the model can handle images of any size.

Loss Function: The loss function we used is a linear combination of classification loss and box regression loss. It is defined by:

$$\begin{aligned} L(p_i, t_i) = &\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \\ &+ \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \end{aligned} \tag{2}$$

where i is the i^{th} anchor in a mini-batch and p_i is the probability that there is an object in the i^{th} anchor. p_i is the ground truth whether there is an object in the *ith* anchor. $p_i = 1$ indicates the i^{th} anchor is positive and $p_i = 0$ indicates the anchor is negative. t_i is a vector representing the

[1] https://github.com/sgp715/simple$_i mage_a nnotator$

[2] https://www.myfitnesspal.com/

Name	Num. Samples	Num. Categories	Samples per Category
Food Dataset	186,343	221	221
Calorie Dataset	66,300	221	221
AFDD	8966	10	896

Table 2: Summary statistics about our food and calorie datasets.

4 parameterized coordinates of the the predicted bounding box and t_i^* is that of the ground truth box associated with a positive anchor.

Calorie Retriever: The Calorie Receiver takes a list of predicted food labels as input and returns a list of mean calorie counts for each food item. The Receiver consults a lookup table that includes mean calorie counts, standard deviations, and number of samples for each of the 10 foods in the dataset. The overall calorie count is then computed as a weighted sum of the food's calorie counts. In future models, we plan to incorporate the standard deviations to calculate confidence intervals around the calorie predictions.

4.3. Proposed Model

Architecture: Our proposed model (Stage 3) shares the same convolutional layer architecture with faster-rcnn. However, instead of two outputs, our last linear layer outputs three: food classifications, bounding boxes, and calorie predictions which are outputs of a bounding box regression layer, a food classification layer and a calorie regression layer respectively.

Input Image Size: Due to fully connected convolutional layer, we are able to input images of any size.

Loss Function: The loss function we proposed in this model is a linear combination of classification loss, box regression loss and calorie regression loss. It is defined by:

$$\begin{aligned} L(p_i, t_i) = & \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \\ & + \lambda \frac{1}{N_{breg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \\ & + \beta \frac{1}{N_{creg}} \sum_i p_i^* L_{reg}(c_i, c_i^*), \end{aligned} \tag{3}$$

where *i* is the i^{th} anchor in a mini-batch and p_i is the probability that there is an object in the i^{th} anchor. p_i is the ground truth whether there is an object in the i^{th} anchor. $p_i = 1$ indicates the i^{th} anchor is positive and $p_i = 0$ indicates the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the the predicted bounding box and t_i^* is that of the ground truth box associated with a positive anchor. c_i is a scalar indicate the calorie of the object in a predicted bounding box and t_i^* is that of the ground truth calorie associated with the object in a positive anchor.

5. Experiments and Results

5.1. Food Classification

In order to perform object detection, we needed demonstrate we could succeed at food classification. To do this, we trained a pre-trained VGG16 network on our aggregated food dataset consisting of over 220 classes. For training, we used a batch size of 16, cross entropy loss, and stochastic gradient descent with a learning rate of 0.01. The model was trained for 10 epochs.

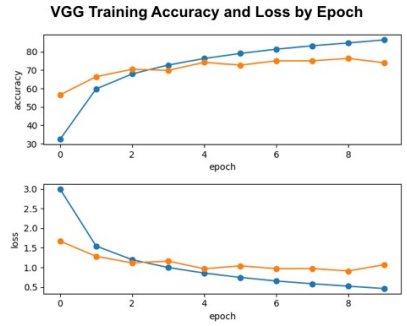


Figure 2: Accuracy and loss over epoch.

Figure 3a are top 9 results for the category carp and tofu soup selected by our classifier in the validation dataset.And they are predicted with confidence 1. Figure 3b is the result of a random sample from validation dataset. As shown in the figure, our classifier predict it as ramen with a high confidence. In a nutshell, statistics and sample results demonstrated that our classifier is legitimate.

5.2. Food Detection

We used the VGG16 model trained for classification as the feature extraction engine for the food detection model. During training, we froze the first three convolutional layers in the network and trained the whole model for 70000 iterations. The object detector is built by pytorch-faster-rcnn [3]from github.

Figures 4a, 4b, 4d show the test results when we set confidence threshold for detection to 0.8, which is denoted as $P(lable|box) >= 0.8$. As can be seen, it is not able to capture fries in the prime rib dish. However, it is able to capture

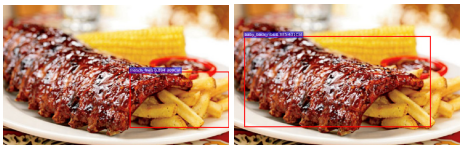
[3]https://github.com/ruotianluo/pytorch-faster-rcnn

(a) Top 9 results for carp and tofu soup

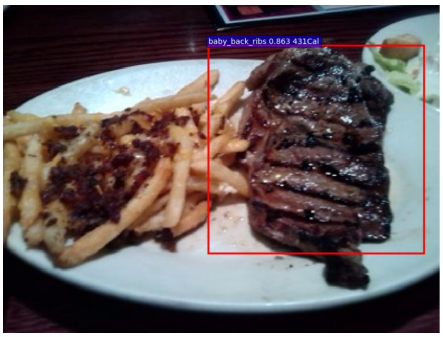
```
Image predicted as ramen with confidence 0.99
y_hat[carp and tofu_soup] = 0.01
y_hat[stirfry_pork] = 0.00
y_hat[fried_peppers] = 0.00
y_hat[pork_ribs and lotus_soup] = 0.00
```

(b) Statistics for a random sample

Figure 3: Visualization of classification results



(a) French Fries (b) Baby back ribs



(c) Prime rib

(d) French Fries (e) Prime rib

Figure 4: Visualization of detection results

(a) Corn 1 (b) Corn 2

Figure 5: Samples of corn pictures

fries when we set confidence threshold to 0.4. We can also see that the model missed the corn in the background, but this is probably due to blurring.

Table 3 contains detailed information about our validation results. Our food detector struggled to detect corn because we have multiple types of corn: corn in a bowl alone, corn salad and corn off the cob. The corn off the cob images were scraped from the top 100 results from Google images. Thus, there was significant variability in the corn types we were predicting, so this class was particularly challenging. We believe in the future we'll need to add more pictures of these corn types to improve our food detector in that category. Similarly, the model struggled with mashed potatoes and occasionally missed french fries. Again, we need to annotate more images of these class types. Furthermore, some of the error can be attributed to human error, as some categories were particularly challenging to label. For example, it was difficult to distinghish between vanilla ice cream and mashed potatoes when the lighting was dim.

In future work, we plan to 1) examine our annotations, 2) expand our dataset to many other foods and 3) extend the prediction to include other food data, such as glucose levels, carbs, fat, etc. Additionally, we plan to construct a dataset that has images, bounding boxes, and calorie labels so we can train the calorie regressor.

6. Acknowledgents

This research is conducted on Jingyun Ning's computer.

Category	Mean AP
Hamburger	0.746
Pizza	0.924
Pulled Pork Sandwich	0.73
French Fries	0.763
Prime Rib	0.799
Apple Pie	0.606
Corn	0.628
Baby-Back Ribs	0.741
Chocolate Cake	0.667
Mashed Potatoes	0.673
Overall	0.728

Table 3: Summary statistics about validation results.

References

[1] O. Beijbom, N. Joshi, D. Morris, S. Saponas, and S. Khullar. Menu-match: restaurant-specific food logging from images. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 844–851. IEEE, 2015.

[2] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[4] C.-w. N. Jing-jing Chen. Deep-based ingredient recognition for cooking recipe retrival. *ACM Multimedia*, 2016.

[5] H. Kagaya, K. Aizawa, and M. Ogawa. Food detection and recognition using convolutional neural network. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1085–1088. ACM, 2014.

[6] Y. Kawano and K. Yanai. Food image recognition with deep convolutional features. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 589–593. ACM, 2014.

[7] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pages 25–30. IEEE, 2012.

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[9] G. Shroff, A. Smailagic, and D. P. Siewiorek. Wearable context-aware food recognition for calorie monitoring. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pages 119–120. IEEE, 2008.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] L. Zepeda and D. Deal. Think before you eat: photographic food diaries as intervention tools to change dietary decision making and attitudes. *International Journal of Consumer Studies*, 32(6):692–698, 2008.