Calorie Estimation From Fast Food Images

Kaixi Ruan
kaixi@stanford.edu

Lin Shao
lins2@stanford.edu

December 12, 2015

Abstract

The project consists of two steps: identifying food from an image, and convert the food identification into a calorie estimation. We perform food image classification using SVM and deep learning algorithms. Different features such as LBP, HOG, and CNN) are explored and compared. For the calorie estimation step, we create a "calorie map" for the image classification labels.

1 Introduction

Calorie Tracker applications are popular for on-diet population. With such applications, people can enter their meals and estimate the daily consumption of calories. Somehow, the meal entry task can be inconvenient: sometimes it may not be possible to judge the size of the dish (while eating out for example). Thus a good idea to help making these applications more user-friendly is to ask users to take a photo of their meal, and from that photo to give a calorie estimation.

Hence, this project aims to achieve this task: predict the calorie content of a meal from an image showing one or more food objects. Technically speaking, the project involves several tasks: identifying the food contained in an image, estimating the quantity of food contained, and converting the food classification into an calorie estimation. For the scope of this project, we ignore the scale and quantity of food in an image and treat it as an image classification problem and we then convert the predicted labels to calorie estimations using online food database.

We use publicly available fast food image dataset for this project. We manually label our dataset, and test different classification algorithms: mainly SVM and CNN. Different features used in image processing are used and compared.

The outline of this report is as follows: in section 2 we describe the dataset we use and the pre-processing steps taken before any actual machine learning implementation. In section 3, we describe the different features we looked at for SVM, including the deep learning tools we used. In Section 4 we present some test results. Section 5 concludes with some discussion and guidelines for the next step.

2 Related Work

Tatsuya et al [5] tried to estimate Calorie by comparing input image with ground truth data based on image features such as color histograms, color correlograms and SURF features . Mei et al [2] used color histogram and bag of SIFT features to build a discriminative machine learning model to classify food images. However shape and texture features are rarely used in previous work, we adopt HOG and LBP descriptors to capture these features.

3 Dataset

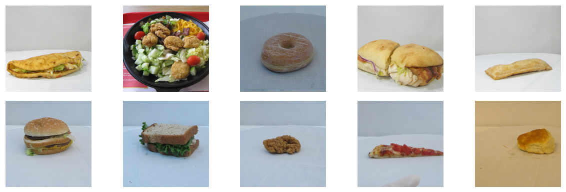


Figure 1: Ten categories we labeled for PFID data set (from top-left to bottom right): burrito, salad, donuts, breakfast sandwich, pie, burger, toast sandwich, chicken, pizza and bread

Pittsburgh Fast Food Image Dataset is a data set containing 4,545 images of fast food, 606 stereo pairs, 303 360 degree videos. We choose 1359 fast food images taken in laboratory or restaurants and divide them into ten categories. The ten categories are burrito, salad, donuts, breakfast sandwich, pie, burger, toast sandwich, chicken, pizza and bread.

4 Pre Processing & Features

Calorie Map We collected some data from the following web pages `http://nutrition.mcdonalds.com/getnutrition/nutritionfacts.pdf` `http://www.calorieking.com/foods/calories-in-fast-food-chains-restaurants_c-Y2lkPTIx.html?bid=747` and `http://fastfoodnutrition.org/pizza-hut`, which contain food calorie tables. Then we get a clean category-to-calorie dictionary so that once we get the classification result we can convert it easily into calorie.

Category	kCal
burrito	300
salad	250
donuts	300
breakfast sandwich	400
pie	230
burger	450
toast sandwich	300
chicken	350
pizza	250
bread	200

Table 1: category-to-calorie Table

Image Cropping Note that the images from PFID are taken in the laboratory with a huge white background. It is generally considered good to crop out the features in question for a better CNN result[1]. Thus, we use the function `crop` from Python Image Library to process these images.

Data Augmentation We rotate images every time by 45 degree from 45 degree to 315 degree. We gain 18323 images totally. We split the data set into training set (13425 images), validate set (2264 images) and test set (2632 images).

HOG features HOG (Histogram of Gradients) is a local feature descriptor applied in image processing. The image is divided into small regions called Cell and intensify gradients are calculated over the pixels in the cells. Histogram of gradients are computed in every cell. The histograms are then normalized over larger spatial regions called blocks. HOG is computed from images scaled to resolution of 120 × 120. The picture below shows hog feature of a given picture in our data set.

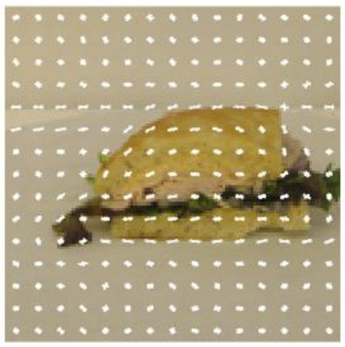


Figure 2: HOG feature

LBP features LBP is also a local feature descriptor used to capture texture information of an image. After changing the image from RGB to gray format, the image is divided into small cells. The value of every pixel is compared to its 8 neighbor pixels. If value in center is higher than neighbor's value, 1 is set. Otherwise, set it 0. It generate 8 digits after comparing all 8 neighbor pixels. A histogram counting the occurrence of 8 digit series in each cell is then created. Concatenating histogram in all cells will finally generate the feature vector. A picture below shows the pipeline of computing LBP.

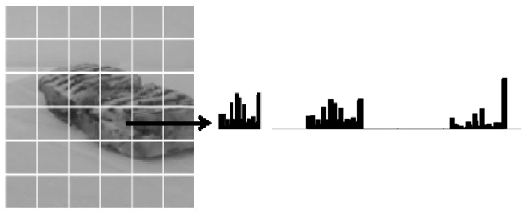


Figure 3: LBP feature

5 Methods

SVM SVM is a classical machine learning algorithm. It gives classification boundaries by optimizing the margin and some regularization terms. We use one vs one method when SVM is applied to multiclass classification problem.

A linear kernel is used when SVM classifies two classes at each iteration.

CNN We run CNN on Caffe[3]. In this stage, we adopt BVLC Reference CaffeNet from `http://caffe.berkeleyvision.org/model_zoo.html`. It is mainly based on the model that Alex trained on ImageNet Large Scale Vision Recognition Challenge in 2012 (ILSVRC 2012). Krizhevsky et al. describe it in ImageNet classification with deep convolutional neural networks in NIPS 2012[4]. Table 1 shows the architecture of CNN model we use.

Layers
fc7 relu7 drop7
fc6 relu6 drop6
conv5 relu5 pool5
conv4 relu4
conv3 relu3
conv2 relu2 pool2 norm2
conv1 relu1 pool1 norm1

Table 2: Architecture of CNN model

Fine Tuning Fine tuning uses a pretrained CNN model and starts to train on new data set after changing some higher lowers if necessary. The idea is based on observations that lower layers capture more generic image features while higher layers deal with more specific image features associated with user's data set. Because the data set we have is small, it is better to fine tune CNN model. We fixed the first five layers and update parameters in the last two fully connected layers. Therefore in the fine tuning process, back propagation only exits in the layer two layers.

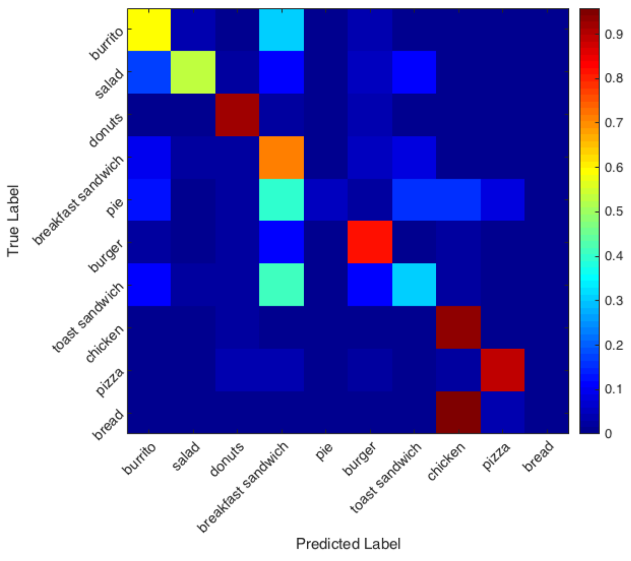
Back propagation is an optimization algorithm used in CNN models. The back propagation is based on the Chain Rule of taking derivatives.$\nabla f(g(x)) = \nabla f(g(x))\nabla g(x)$.

Back propagation algorithm could be divided into two stages. In first stage, a forward propagation from input data layer to output layer. Then a loss function is calculated based on the output and its true value. The derivative of the loss function is taken with respect to different layers' parameters. The direction of taking derivatives is from higher layers to lower layers based on the Chain Rule. In the second stage, the weight is updated by subtracting the gradient multiplied by a learning rate.

6 Results and Discussion

6.1 SVM on local features

We run SVM on local features HOG and LBP. The confusion matrices are shown below. Generally speaking, LBP is a little better than HOG features. It indicates that texture information is better than shape information in our classification problem. The LBP in classifying bread and pie outperforms the HOG feature.



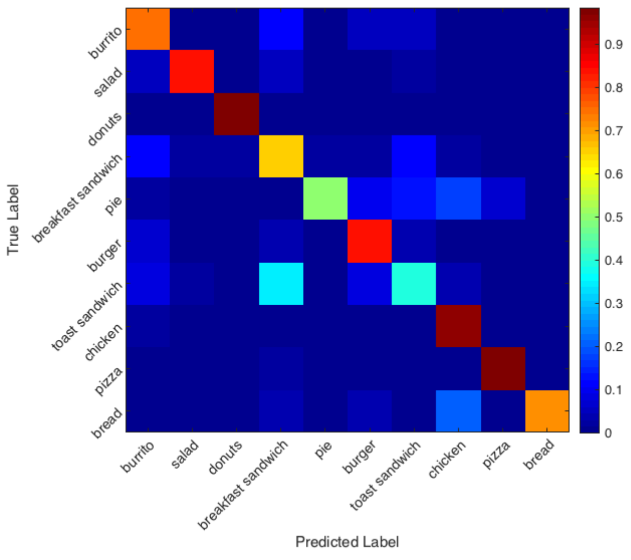


Figure 4: Confusion Matrix for HOG (top) and LBP(down)

6.2 SVM on CNN features

An accuracy curve is generated when we fine tune CNN models. The horizontal axis shows iterations. The vertical axis represent accuracy percentage on validate set.

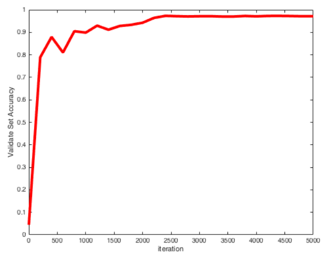


Figure 5: Accuracy Curve

We choose the model after 3000 iterations to avoid over fitting problem.

We then extract the features from the fc6 and fc7 layers. Each feature is a 4094 dimension vector. A linear classifier SVM is used to train these features. We also extract the features from fc6 and fc7 layers of the pre-trained CNN model. The features are then trained using the same linear classifier. The confusion Matrices are shown below.

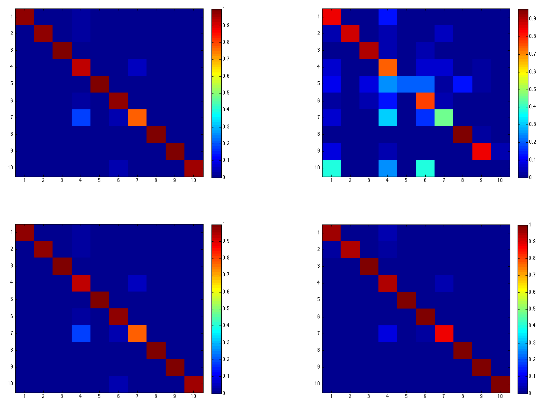


Figure 6: Confusion Matrices using CNN features: fc6 no fine-tuning(top left); fc7 no fine-tuning(top right); fc6 fine-tuning (down left); fc7 fine-tuning (down right).

We find out that the test results from fc6 features are the same for pre-trained model and fine-tuned model. The test error from fc7 in pre-trained model is bigger than fine-tuned model. It indicates that fine-tuning process mainly changes the fc7 layer and improves its output. It could be explained by the observations that higher layer is more specific with given dataset.

The accuracy percentage table below shows the accuracy percentage using different methods.

SVM + HOG	72.53%
SVM + LBP	80.48%
SVM + fc6(pre-trained model)	96.25%
SVM + fc7(pre-trained model)	79.20%
SVM + fc6(fine-tuned model)	96.25%
SVM + fc7(fine-tuned model)	97.17%
CNN(fine-tuned model)	96.84%

Table 3: Accuracy Percentage Table

7 Future work

We want to enlarge the dataset. Currently dataset is small and contains only 10 categories, we want to put more food types in the dataset. Ideally it is also better to add more images to the pre-existing categories: it might be interesting to combine different features to achieve better accuracy, also to use a nonlinear kernel in SVM (throughout the project we have been using linear kernels). Worried about over-fitting issues with our current data set, we choose not to do so.

The optimization becomes more and more important in improving CNN performance. We also want to partially focus on optimization method, i.e, we want to use different optimization methods and compare their performance in order to improve the optimization method if possible.

References

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014.

[2] Mei Chen, K. Dhingra, Wen Wu, Lei Yang, R. Sukthankar, and Jie Yang. Pfid: Pittsburgh fast-food image dataset. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 289–292, Nov 2009.

[3] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q.

Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Tatsuya Miyazaki, Gamhewage C. de Silva, and Kiyoharu Aizawa. Image-based calorie content estimation for dietary assessment. In *ISM*, pages 363–368. IEEE Computer Society, 2011.