Project Report: Food Recognition with Convolutional Neural Networks

Andrew Hou, Sushanta K. Pani, Abhi Durgaraju

November 30, 2019

1 Data

For the training set, we have 14 food classes: Apple, Banana, Broccoli, Burger, Cookie, Egg, Frenchfry, Hotdog, Pasta, Pizza, Rice, Strawberry, Salad, and Tomato. Each training image is only a single food category and is resized to 128x128. Each food category has at least a few hundred (400-1000+) training samples and comes from the ImageNet dataset and google image search. Our testing set consists of 20 images per food category for a total of 280 images, where each image has a single food category. These are all in-the-wild images collected from google image search and include some challenging images for each category with varying pose, quantity, and form. For example, for the apple category we test not only whole, well centered apples, but also apple slices.

2 Model

We started by using a shallow convolutional neural network with 3 convolutional layers and 2 fully connected layers. The input was an RGB image of size 128x128x3, normalized between 0 and 1. The first convolutional layer had 7x7 convolutional filters and produced 32 feature maps. The second convolutional layer had 5x5 filters and produced 64 feature maps. The third convolutional layer had 3x3 filters and produced 128 feature maps. Every layer was followed by ReLU activation functions and used a stride of 2. Next we concatenated our 128 16x16 feature maps into a single 1x32768 feature vector. We passed this feature vector to 2 fully connected layers, the first of which had a hidden size of 128 and was followed by the ReLU activation function, and the second of which had hidden size equal to the number of food classes (14). We applied softmax to the output of this dense layer and used softmax cross entropy as our loss function. The groundtruth labels were represented as 14-dimension one-hot vectors, where each dimension is a separate food category. We used a batch size of 64 and a learning rate of 0.0001. We used the Adam Optimizer, and implemented this network in Tensorflow. We train for 1000 epochs.

As is, this model did not perform particularly well. The first problem that we noticed was overfitting. At some point in the training, the network would basically converge at 0 loss, which is a clear sign that overfitting was occurring. To prevent this, we employed two countermeasures: data augmentation and dropout. We used horizontal flipping and random rotations to make it more challenging for the network to "memorize" the training data, and also to make it more robust to varying poses and camera angles. We applied dropout with keep probability 0.5 to the first dense layer to further reduce overfitting.

Some other improvements that we made included adding a 4th convolutional layer to the network, which led to a performance boost, and replacing 2x2 stride with 2x2 max pooling (which led to a small increase in testing accuracy). The first 3 convolutional layers remained the same, and the 4th layer used 3x3 filters and produced 256 8x8 feature maps, which means the first dense layer was now passed a feature vector of size 1x8*8*256 or 1x16384.

3 Evaluation

We utilize the test set that we described in the data section for evaluation purposes. The recognition rates (average of true detection and true rejection rates) denoted as RR per class are shown below:

Class	TDR	TRR	RR
Apple	0.450	0.992	0.721
Banana	0.600	1.00	0.800
Brocolli	0.350	0.981	0.665
Burger	0.750	0.973	0.862
Cookie	0.600	0.985	0.792
Egg	0.850	0.958	0.904
Fries	0.350	0.981	0.665
Hotdog	0.750	0.950	0.850
Pasta	0.500	0.942	0.721
Pizza	0.600	0.962	0.781
Rice	0.350	0.996	0.673
Salad	0.900	0.931	0.915
Strawberry	0.900	0.969	0.935
Tomato	0.750	0.973	0.862
Avg (all classes)	0.621	0.971	0.796

Due to time constraints, we did not get to test on images with multiple food categories as we originally planned. However, we have an approach that we believe would have worked. We would use a sliding window approach to try to find every instance of food in the image. We would set a confidence threshold such that for each window, we will only accept the classification result if

the confidence (softmax probability) of the network's prediction surpasses the threshold. This will help filter out many windows that only contain background or that contain only part of the food item. We will then run non-max suppression to ensure that we don't have too many bounding boxes per food item. This approach would also allow us to count the number of food items in the image. It is important to note though that such an approach would lead to a significant increase in computation time per image, and thus the average speed per image would be much slower.