Project Report

Xianlong

Dec 2016

1 Introduction

In this report, I will use the attention model to classify and locate the foods in the image. First, I will use the CNN (vgg-19-layer model) to extract the feature of the images. Then I apply attention model to generate the labels.

2 Algorithm

I will use Encoder-Decoder Architecture.

2.1 Encoder

The encoder is the convolutional features of the images. I use a convolutional neural network: VGG-19 layers network, which is pretrained on "ILSVRC2012" dataset. The network produces L feature vectors (in this case, L = 196), each of which is a D-dimensional representation corresponding to a part of the image (in this case, D = 512).

$$A = \{\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_L}\} \tag{1}$$

2.2 Decoder

The decoder uses an attention model. Below is one way to construct the attention model, I also try many other ways, which will be present in the experiment part.

$\mathbf{a_l}$ is the feature vector extracted by CNN, $\mathbf{e_t}$ is the word embedding vector, α_t is the attention weight vector, $\mathbf{z_t}$ is the weighted context vector, t is the id for label/word, x is the input image, $p(t|x)$ is the probability of the output label/word given an image, W_a, U_a are the weight matrix, $\mathbf{v_a}, \mathbf{w_t}$ are the weight vector.

$$e_{tl} = \mathbf{v_a^T} \tanh(\mathbf{a_l} W_a + \mathbf{e_t} U_a) \tag{2}$$

$$\alpha_\mathbf{t} = softmax(e_{t1}, e_{t2}, ..., e_{tL}) = (\alpha_{t1}, \alpha_{t2}, ..., \alpha_{tL}) \tag{3}$$

$$\mathbf{z_t} = \sum_{l=0}^{L} \alpha_{tl} \mathbf{a_l} \tag{4}$$

$$p(t|\mathbf{x}) = \sigma(\mathbf{w_t^T z_t}) \tag{5}$$

Let $\mathbf{y}(\mathbf{x})$ be the vector of predictions from the model, $\mathbf{z}(\mathbf{x})$ be the vector of true labels, and $loss(\mathbf{x})$ be the log-likelihood loss incurred on example $\mathbf{x}$.

$$\mathbf{y}(\mathbf{x}) = [p(t_1|\mathbf{x}), p(t_2|\mathbf{x}), ..., p(t_T|\mathbf{x})] \tag{6}$$

$$loss(\mathbf{x}) = -\mathbf{z}^T(\mathbf{x}) \ln \mathbf{y}(\mathbf{x}) - (\mathbf{1} - \mathbf{z}(\mathbf{x}))^T \ln(\mathbf{1} - \mathbf{y}(\mathbf{x})) \tag{7}$$

I also try different decoder, which only change equation (2) to the following equation:

$$e_{tl} = \tanh(\mathbf{a_l}^T \mathbf{e_t}) \tag{8}$$

This change is due to the very small training dataset we have, and this will reduce the number of parameters.

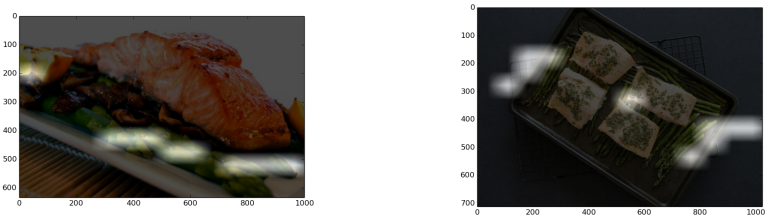
3 Dataset

To obtain the food image for training, I google the name of the food items, like "salmon and asparugus", "toast and eggs", "avocado and salmon", etc., and download the first 300-500 images. Overall, I download around 4500 images with 7 labels (salmon, asparagus, avocado, spinach, rice, toast, eggs) and 14 combination categories.
During the experiment, I randomly choose 90% of the images as the training dataset and 10% images as the testing dataset. I didn't use a validation dataset because the whole dataset is very small.

4 Experiment

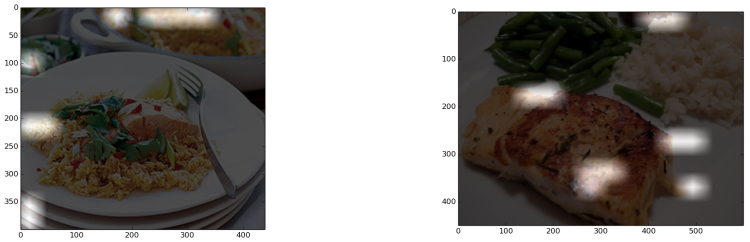
4.1 Quality of the features

I use the VGG-19 layers network, which is pretrained on "ILSVRC2012" dataset. However, the "ILSVRC2012" dataset contains more than 10000 object categories, like human, animals, plants, etc., it is not trained on the food images. So, I need to make sure the features I extracted from the model is good.
I run the following sanity check for the features.
Assume $\mathbf{a_i}$ is the feature vector for location i, $\mathbf{a_j}$ is the feature vector for location j, if the two locations contains the similar items, then their cosine similarity value should bigger than those locations contains dissimilar items. I located the feature vector that belong to a specific label, say asparagus, and then try to find the top-10 similar features by ranking their similarity. After that, I visualized the part of the result, which can be shown in Figure 1 and Figure 2. Roughly there are 60/100 are good result. Note that for rice features, the overall quality are worse than others.



(a) Good result (b) Bad result

Figure 1: Test quality for Asparagus features



(a) Good result (b) Bad result

Figure 2: Test quality for Rice features

4.2 Parameter Numbers

For the attention model mentioned in Section2, I use around 20K parameters.

$\mathbf{e_t} = [1, 256]$ (Word-Embedding parameters: 256)

$W_a = [512, 16]$ (Image-Encode parameters: 8K)

$U_a = [256, 16]$ (Word-Embedding-Encode parameters: 4K)

$\mathbf{v_a} = [1, 16]$ (attention weight parameters: 16)

$\mathbf{w_t} = [1, 512]$ (weighted context parameters: 512)

Note that I also try to increase the number of parameter by changing the size of W_a, U_a, $\mathbf{v_a}$:

For example: $W_a = [512, 256]$, $U_a = [256, 256]$, $\mathbf{v_a} = [1, 256]$ .This will dramatically increase the number of parameter to around 200K, the training loss for this parameter will decrease (as expected), but the performance of the model is not increasing.

For the model that use equation (8), it will decrease the number of parameters to around 8K.

4.3 Training

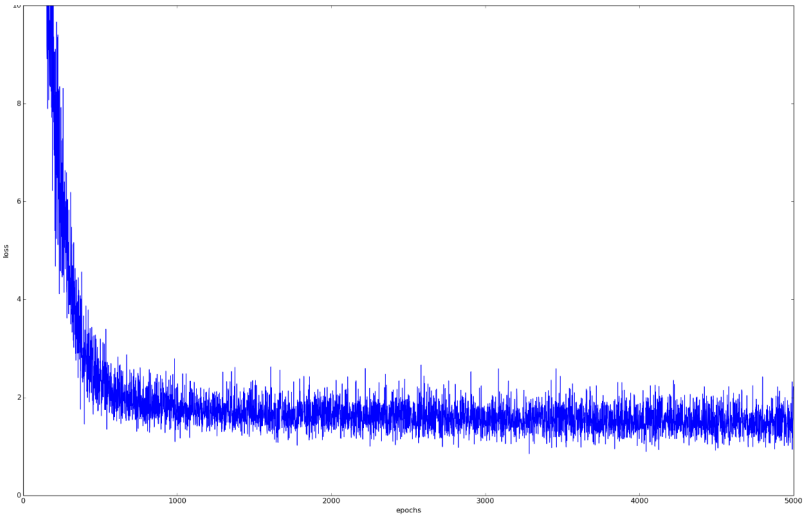
Using 90% of the dataset as the training data, Adam-Optimizer with learning-rate = 0.0001 and batch-size = 64, I have the training loss shown in Figure 3(b). From the image, we noticed that the loss has a lot of zig-zag pattern during training procedure. This might because the features of the image is not good enough to learning the pattern or the capacity of the model is not enough. In order to find out whether the capacity of the model is enough or not, I increase the number of parameters as mention in section 4.2. After increasing the number of parameters to 300K, I got the training loss in Figure 3(c), shown in Figure3. Note that I only ran for 2000 epochs under this setting due to the long running time. As a comparison, the loss converges to a lower value, but the zig-zag pattern still exist.
For the model that use equation (8), As show in Figure 3(a), it will converge to a higher loss value. Note that I ran 5000 epochs under this setting because I want to make sure the loss value is converge.

4.4 Testing

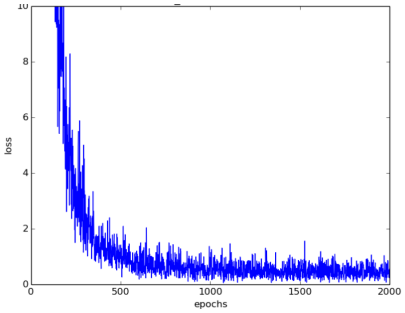
Using 10% of the dataset as the testing data. As mentioned in Training section, I used 3 models, one has 300K parameters, one has 20K parameters, one has 8K parameter, they have the overall accuracy: 136/450 (30.2%), 195/450 (43.3%), 154/450 (34.2%). I visualized the predicted images to see if the output image have the right attention, as shown in Figure4 and Figure5.
As we can see from the images, model with 8K parameters is doing better for capturing the attention for the corresponding label, the attention area is closer to humans' decision.

5 Related Work

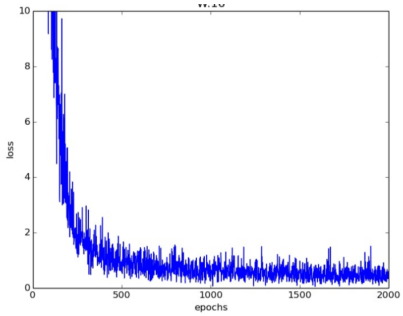
There are multiple works related to food image classification and the following works do the experiment on Food101 dataset[1], [1] introduced a Random Forests (RF) model and CNN model to do the classification. [7] use the fine-tuned DCNN which was pre-trained with 2000 categories in the ImageNet including 1000 food-related categories and achieved 70.41%. [2] use a CNN-NIN model to classify the food/non-food image and achieved accuracy: 99.1%
The following paper use food-101 dataset for some applications. [4] present a novel method for aligning a sequence of instructions to a video of someone carrying out a task. In paticular, the visual food detector (CNN-based, pretrained on ImageNet) that was used in the mothod reach the accuracy of 79%, which is the state-of-the-art classifier. [3] apply a new approach to scene and object classification tasks. [5] apply the CNN-based classifier (running on the phone) to predict which foods are present in the meal, and lookup the corresponding nutritional facts, it apply this method to a new dataset of images from 23 different restaurants. [6] deals with automatic systems for image recipe recognition (using UPMC Food-101 dataset, different from ETHZ Food-101 dataset, the



(a) loss with 8K parameters



(b) loss with 20K parameters



(c) loss with 300K parameters

Figure 3: Training Loss

one we are using)
None of the above works use multi-object food image dataset, hence my work is relatively new, I have to collect the data myself and this is also the reason I can't find enough labeled food images to train the model.

6 Future Work

The goal of this project is to build a model so that it can recognize the food image and get foods' location. The result I got so far is bad. I think it is due to two reasons:

1. The number of images in the dataset is small
2. The quality of the features is not good.

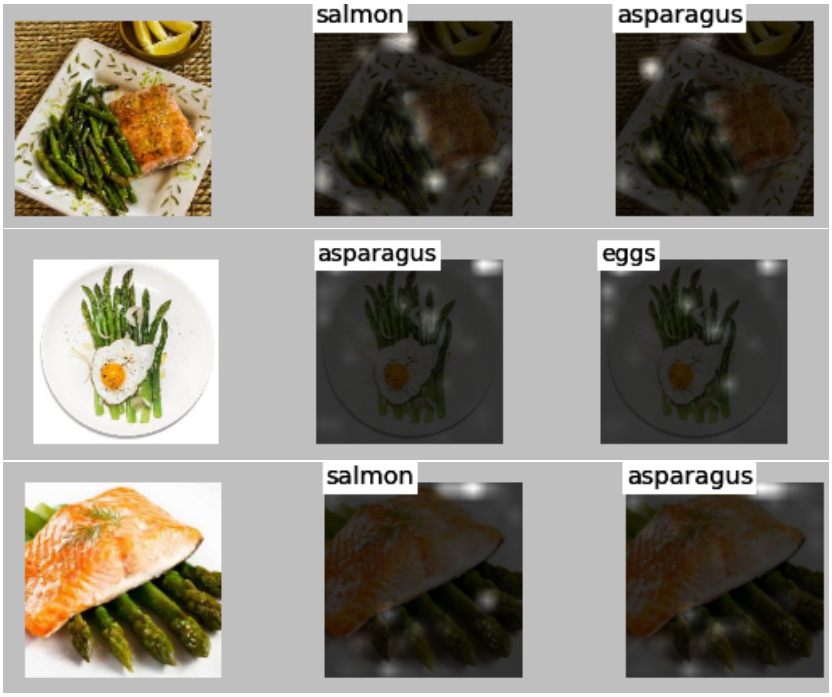


Figure 4: model with 20K and 300K parameters

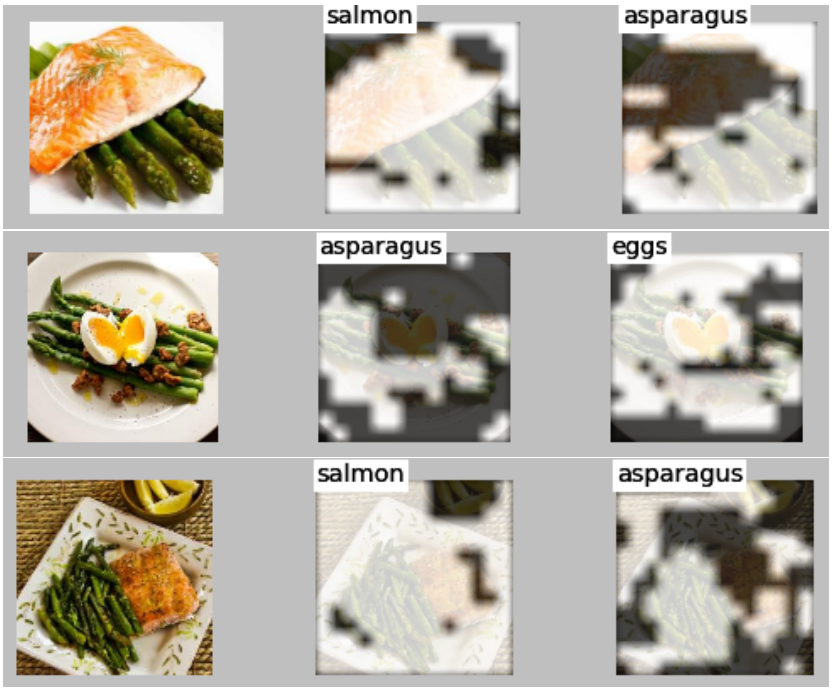


Figure 5: model with 8K parameters

So, to improve the result, I think the first way one can do is to enlarge the dataset. Actually I've done the similar work on single label dataset – Food101 dataset, which contains 101,000 images with 101 categories. I build the model and the result is much better than the one I've got on the smaller multi-object dataset, as can be seen in Figure6. This shows that with more training dataset, one should able to get a much better result on food image recognition. Also, with this small amount of dataset, we can't build a complicate model, we have to limit the number of parameters so that it won't overfit.
Another way to improve the result is extract a better feature vector for the food item. I used the pretrained CNN model that is not trained on food items. To deal with this problem, I used a weight matrix to encode the raw feature vectors. A better way to do this will be finetune the CNN model so that one can have a better feature vectors for the food items. This will be very important for goal, since in the multi-object recognition task, food items are always overlab with one another, for food items like rice, eggs, avocado, etc., it's very often to get a bad feature vectors for those small & overlap items.

7 Conclusion

In this project, I use the attention model to localize the location of the food items and classify them. I used around 4000 images to train the model and used 450 images for testing, I got the overall accuracy is 34.2% (I choose the model with best attention weight). The attention of the image is not very good, there are still many ways to improve the result.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[2] Hokuto Kagaya and Kiyoharu Aizawa. Highly accurate food/non-food image classification based on a deep convolutional neural network. In *International Conference on Image Analysis and Processing*, pages 350–357. Springer, 2015.

[3] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Mid-level deep pattern mining. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 971–980. IEEE, 2015.

[4] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558*, 2015.

[5] Austin Meyers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan

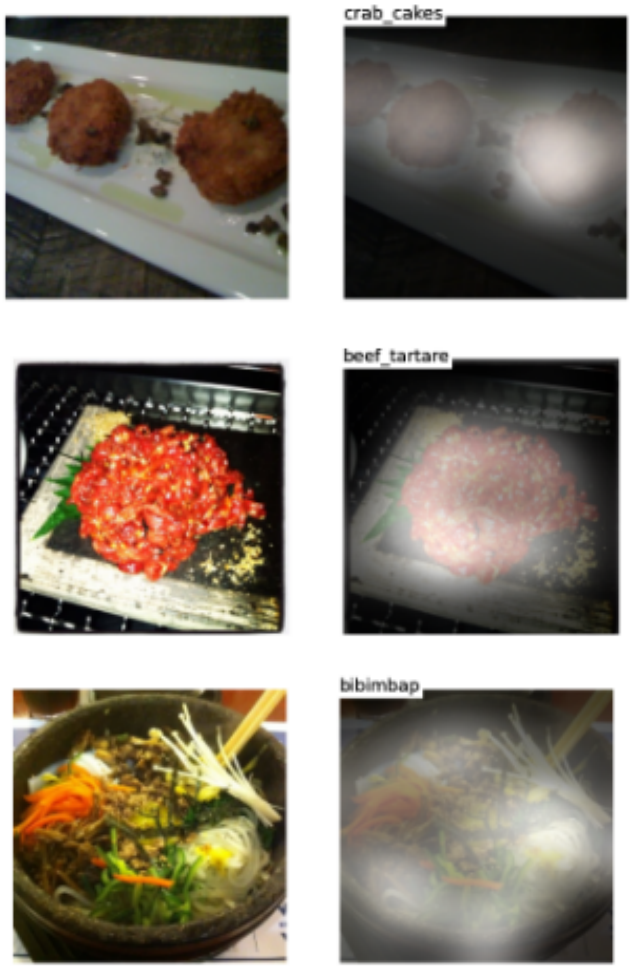


Figure 6: model with Food101 dataset

Huang, and Kevin P. Murphy. Im2calories: Towards an automated mobile vision food diary. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[6] Xin Wang, Devinder Kumar, Nicolas Thome, Matthieu Cord, and Frederic Precioso. Recipe recognition with large multimodal food dataset. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[7] Keiji Yanai and Yoshiyuki Kawano. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.