ResNet-based Food Recognition
University of Pennsylvania CIS519 Course Project

Wudao Ling WUDAO@SEAS.UPENN.EDU
Siyu Zheng SIYUZ@SEAS.UPENN.EDU
Zeshen Liu ZESHEN@SEAS.UPENN.EDU

Abstract

In our project, two state-of-art convolutional neural network model, VGG16 and ResNet50, are implemented to recognize 101 kinds of food. Transfer learning is applied for feature extraction and learning acceleration. Our best ResNet50 model achieves 77.25% Top1 accuracy and 92.90% Top5 accuracy after just 9 training epoch which takes only 5 hour.

1. Introduction

Computer Vision develops rapidly in recent years and become increasingly significant in applications in many fields, like image recognition and self-driving cars. Convolutional neural network is one of the key driving forces of those developments. Image recognition can bring an added level of convenience to humans' daily life, such as food labeling, since delicious food plays an important role in people's lifestyle. Consequently, our group decides to apply CNN with transfer learning ideas on image classification. Our food recognition can be utilized in commercial food recommendation systems or food science research.

2. Methodology

2.1. Image Data

ETHZ-Food-101[1] contains 101,000 images crawled from foodspotting.com and covers 101 most popular food categories. This dataset comes with wrong labels and noisy images, thus the testing result could be a good estimate of real world application on food image classification.

Kaggle Subset of Food-101[2] provides 10,099 training and 1,000 testing images while the image size is $64\times64\times3$. It includes massively downscaled versions of the images to enable quick training and test.

At the beginning, we tried VGG16 model on Kaggle subset of Food-101. After 30 epochs of training from scratch or 10 epochs from ImageNet pre-trained weight, the training accuracy is close to 1 but the testing Top1 accuracy stays around 0.2 (Top5 accuracy 0.45). The overfitting problem results from relatively small image size (these complex CNN models are designed for $224\times224\times3$ or larger image) and small dataset. This trial indicates 2 tasks for us. On one hand, we need to preprocess ETHZ-Food-101 to obtain more data with appropriate image size. On the other hand, transfer learning should be fully utilized, which means we can make use of ImageNet pre-trained weights to extract features and accelerate training.

Since images from ETHZ-Food-101 dataset have different sizes, we need to resize images. Using OpenCV, We center cropped and resized all the images to $224 \times 224 \times 3$ pixels, as large as most of images in ImageNet. As food images are with different background environment, it would be better to normalize color as well. After resizing, we can implement Grey World method to adjust all images to have similar grey value, then use Histogram Equalization algorithm to adjust the contrast and luminance for all images. For comparison, the dataset only with resizing is called raw data while the dataset also with color normalization is called normalized data.

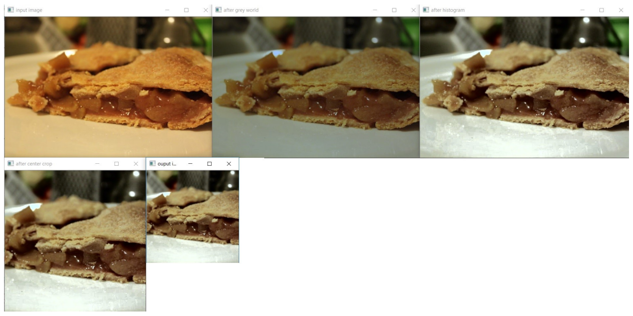


Figure 1. Image Normalization and Resizing

The last step of preprocessing is training and testing split. There are 1000 images for each food category, 20% of which are randomly chosen for testing purpose.

2.2. Hardware

The prototyping of this project is on Dell XPS15 with Windows TensorFlow in python 3.5. This computer has a 16GB RAM and a NVIDIA GTX 1050 GPU with 4GB memory. The memory sizes of RAM and GPU are limitation. The RAM can't load the preprossessed image data as a whole numpy array (17GB), therefore we manage to use python generator and load data as stream. The GPU can't train VGG16 with batch size 1 due to model parameter size (3 fully connected layers at last), and this is why we turn to ResNet50. The GPU handles last layer training for ResNet50, but limit the batch size during full model training.

In search for better performance and less limitation, our deep learning framework was also built on Amazon AWS. We used p2.xlarge instance for full network training. It uses NVIDIA Tesla K80 GPU with 12GB memory and 61 GB RAM.

2.3. Models

2.3.1. VGG16

VGG16 is a 16 layer VGG Net that used 3x3 convolutional filters with stride and pad of 1, along with 2x2 max pooling layers with stride 2 and fully connected layers[3].

2.3.2. ResNet50

ResNet50 is a 50 layer Residual Neural Networks and its basic idea is adding the output f(x) derived from some residual block of conv-relu-conv series to the original input x. It is easier to optimize the residual mapping probably because during the process of back propagation, the gradient, with the help of the addition, will flow easily through the graph[2].

2.4. Transfer Learning

As known, few people train an entire CNN from scratch due to the limit of dataset size. It is common to use the pre-trained CNN on a very large dataset as an initialization or a fixed feature extractor[5].

The size of our ETHZ-Food-101 dataset is limited, while ImageNet contains 1.2 million images with 1001 categories. Naturally, we decided to apply transfer learning and replace pre-trained model's last layer with 101 output softmax dense layer.

There are two scenarios for transfer learning. One is to take ConvNet as fixed feature extractor, remove the last fully-connected layer(s) and keep the rest fixed during training. Another is to not only replace the last layer(s) but also tune the weights of the pre-trained network to obtain fresh model[5]. After trials, we decided to combine these two ways, train last layer for 5 epochs (batch size 101, initial learning rate 0.001) and then fine-tune full network for 4 epochs (batch size 50, initial learning rate 0.00001).

3. Results

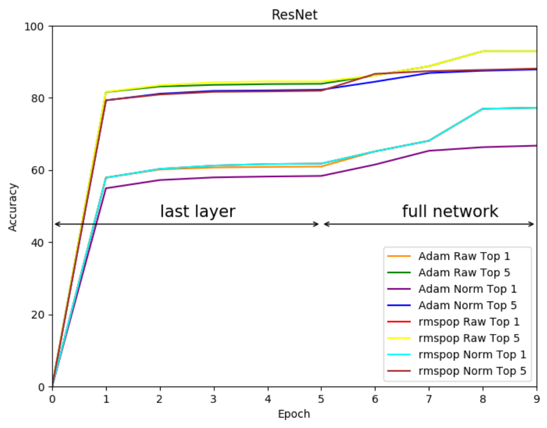


Figure 2. Model Performance (Test Accuracy)

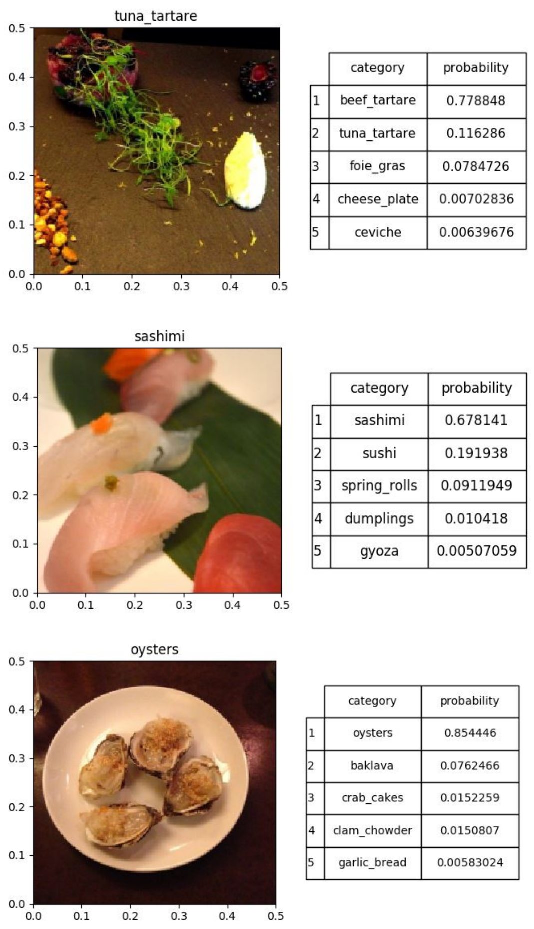
Table 1. ResNet50 last layer training on raw data with RMSProp optimizer

Epoch	1	2	3	4	5
Loss	1.6967	1.6102	1.5882	1.5905	1.6053
Top1	0.5786	0.6031	0.6123	0.6165	0.6178
Top5	0.8170	0.8349	0.8431	0.8458	0.8455

Table 2. ResNet50 full network training on raw data with RMSProp optimizer

Epoch	6	7	8	9
Loss	1.4646	1.3023	0.9357	0.9553
Top1	0.6516	0.6811	0.7697	0.7725
Top5	0.8621	0.8876	0.9291	0.9290

We used Adam and RMSProp optimizers to train the model with raw and normalized data respectively. The best model we acquired used RMSProp optimizer and raw data, which can achieve 77.25% Top1 test accuracy and 92.90% Top5 test accuracy. The total training time is about 5 hours.



	category	probability
1	beef_tartare	0.778848
2	tuna_tartare	0.116286
3	foie_gras	0.0784726
4	cheese_plate	0.00702836
5	ceviche	0.00639676

	category	probability
1	sashimi	0.678141
2	sushi	0.191938
3	spring_rolls	0.0911949
4	dumplings	0.010418
5	gyoza	0.00507059

	category	probability
1	oysters	0.854446
2	baklava	0.0762466
3	crab_cakes	0.0152259
4	clam_chowder	0.0150807
5	garlic_bread	0.00583024

Figure 3. Examples of Top5 Predictions

4. Analysis

4.1. Image Processing

Table 3. Training Raw/Normalized data using RMSProp optimizers after 9 epochs

Dataset	Top1	Top5
Raw	0.7725	0.9290
Nomalized	0.6721	0.8811

In the learning process, we found out the model using raw images has higher testing accuracy than that using normalized images. This might indicate 2 reasons. When ResNet model was trained on ImageNet, images were not preprocessed. Some layers of ResNet model may have learned how to handle different color background or contrast. Also, uniform environment and tone might not be good for food classification, since the environment and tone can somehow indicate food category. For instance, baby back ribs will look dark and yellow due to the light of steakhouse while hamburger might look bright due to fast food restaurant background.

4.2. Optimizer Choice

Table 4. Training Raw data using two optimizers after 9 epochs

Optimizer	Top1	Top5
Adam	0.7645	0.9264
RMSProp	0.7725	0.9290

For two optimizers used in our project. RMSProp divides the learning rate by an exponentially decaying average of squared gradients to resolve Adagrad's radically diminishing learning rates[6]. Adam is essentially an RMSProp with momentum terms that dynamically adjust the learning rate of each parameter using the first and second order moment estimates of the gradient. The main advantage of Adam is that after the offset correction, the learning rate of each iteration has a certain range, making the parameters relatively stable[6].

The choice of optimizers is a trade-off between speed and stability. In this project, performance of RMSProp are slightly better than performance of Adam. Because training batch size is reasonably large(50 or 101), the update at each step tends to be stable enough. In this case, RMSProp will converge faster. Adam might be a better optimizer for a smaller training batch size.

4.3. Transfer Learning Scenarios

We tried learning full model from scratch. The training takes around 1 hour per epoch due to larger amount of weight involved and smaller batch size, but it can only achieve 23.9% Top1 accuracy and 51.2% Top5 accuracy after 5 epochs, which is unsatisfactory.

In contrast, using ResNet as fixed feature extractor yields better result after 5 epochs. Top1 accuracy higher than 58% and top5 accuracy at 82%. The training took only around 15 minute per epoch.

After these attempts, our team decided to train only the last layer for first 5 epochs, and pre-trained ResNet parameters are fixed. In this way, we spend much less time and obtain a model with baseline performance. Then, for later epochs, full network training is employed to fine-tune parameters of previous layers.

4.4. Model Comparison

Table 5. Accuracy comparison

MODEL	TOP1	TOP5
ALEXNET[7]	0.564	N/A
GOOGLENET[8]	0.774	0.937
RESNET50(LAST LAYER)	0.6178	0.8455
RESNET50(FULL NETWORK)	0.7725	0.9290

4.5. Visualization

The idea is to visualize the input image that maximizes the last layer activation of trained model, in other word, maximizes specific categorical probability. To do that, we randomly initialize an input image and back-propagate from the last layer's output, which gives the gradient of output with respect to the input image pixels.Then we perform gradient ascent to get image pixels that maximize the output [9].

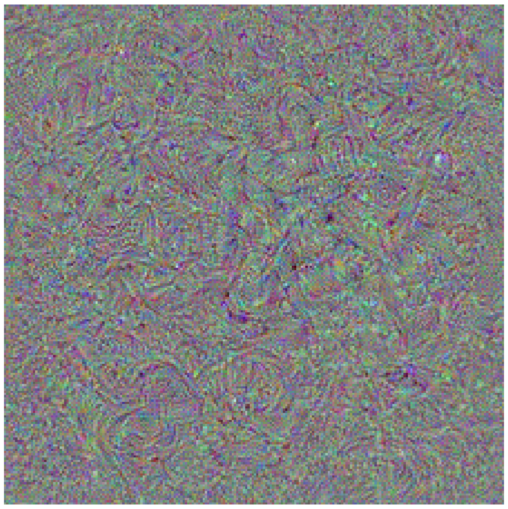


Figure 4. Generated Input Images for french_fries

Most generated input images are with over 0.999 loss, which means our model is over 99.9% confident that these generated input belongs to a specific food category. However, these generated images don't look like their class for human and the only similarity is at level of local texture. The reason is that CNN model understand image as a hierarchical probabilistic mapping between certain combinations of layers and a set of labels, while human doesn't see in this way[9].

5. Conclusions

In conclusion, transfer learning from ImageNet is promising for food image classification. The most efficient method is to fix pre-trained model and learn a stable last layer, then fine-tune the whole network. The choice of optimizer is a trade-off between speed and stability, researchers should consider about dataset and training approach. Raw image data without color normalization tends to fit better with most ConvNets.

There are aspects which can be explored after this project. For data, we can use multiple crops rather than center crop for a single image in our dataset. Such that we can acquire a larger dataset to train a better model which can recognize food from a smaller part. At testing stage, the final prediction could combine predictions from different crops. In addition, we can add in similar food dataset like UPMC-FOOD-101 or crawl images by ourselves.

Besides ResNet, we can implement and tune other cutting-edge models such as GoogLeNet and DenseNet, seeking for more accurate recognition. Furthermore, it would be beneficial to perform CNN food localization before recognition. Because in lots of pictures the food itself only stays at a small area, and localization remove non-related objects.

What's more, we can export food recognition model to tensorflow mobile to create a recommender app that can help user label the food or give them some delicious advice. User can also upload food image to find out the nearest restaurant serves it.

References

[1] ETHZ-FOOD-101 data set, https://www.vision.ee.ethz.ch/datasets_extra/food-101/

[2] Kaggle Subset of Food-101, https://www.kaggle.com/kmader/food41/data

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.

[5] Transfer Learning, https://cs231n.github.io/transfer-learning/

[6] An overview of gradient descent optimization algorithms, http://sebastianruder.com/optimizing-gradient-descent/

[7] Bossard, L., Guillaumin, M., & Van Gool, L. (2014, September). Food-101mining discriminative components with random forests. In European Conference on Computer Vision (pp. 446-461). Springer International Publishing

[8] Liu, C., Cao, Y., Luo, Y., Chen, G., Vokkarane, V., & Ma, Y. (2016, May). DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment. In International Conference on Smart Homes and Health Telematics (pp. 37-48). Springer International Publishing.

[9] How convolutional neural networks see the world, https://blog.keras.io/category/demo.html

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9).

[12] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167

[13] Yu, Q., Mao, D., Wang, J. (2016). Deep Learning Based Food Recognition http://cs229.stanford.edu/proj2016/report/YuMaoWang-Deep%20Learning%20Based%20Food%20Recognition-report.pdf