Bare Demo of IEEEtran.cls for Conferences

Shuang Ao
Department of Computer Science
Western University
London, Ontario
Email: sao@uwo.ca

Charles Ling
Department of Computer Science
Western University
London, Ontario
Email: cling@csd.uwo.ca

***Abstract*—The abstract goes here.**

I. Introduction

Over the recent few years, Convolutional Neural Network (CNN) shows its potential to replace the human engineered features, such as SIFT[1], SURF[2] and HOG[3] etc, in real object recognition tasks. The success of CNN on large scale image set started from Krizhevsky et al[4] and their 8 layer model AlexNet in 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2012), reaching a on top-5 83% accuracy . Soon after, many attempts have been made to improve the model of Krizhevsky. By reducing the size of the receptive field and stride, Zeiler and Fergus improve AlexNet by 1.7% on top 5 accuracy[5]. With the help of high performances computing systems, such as GPU and large scale distributed clusters, it is possible for researchers to explore larger and more complex architecture. By both adding extra convolutional layers between two pooling layers and reduced the receptive window size, Simonyan and Zisserman built a 19 layer very deep CNN and achieved 92.5% top-5 accuracy[6]. While the AlexNet-like deep CNNs conquered ILSVRC, Szegedy et al built a 22 layers deep network, GoogLeNet [7] and won the 1st prize on ILSVRC2014, reaching a astonishing 93.33% top-5 accuracy.

Since these CNN models are trained on very large image data set, they have strong generalization ability and can be applied in many other scenarios. Applying the model pre-trained from ImageNet dataset on other object recognition dataset shows some impressive results. Zeiler et al. applied their pre-trained model on Caltech-256 with just 15 instances per class to fine-tune the model and improved the previous state of the art in which about 60 instances are used for training, by almost 10%[5]. Chatfield et al used their pre-trained model on VOC2007 dataset and outperformed the previous state of the art by 0.9%[8].

Unlike the local features such as SIFT or SURF, which presents an intuitive interpretation of spatial property that is invariant with some transformations such as scaling and rotation, we still don't have enough knowledge to understand the visual features of CNN learned in each layer. Training a large deep CNN on real recognition problem is always a complicated task. The model contains hundreds of millions of parameters to learn and lots of hyper-parameters that can affect its performance. However, the truth that deep CNN outperforms other shallow models by a large margin in some real image recognition tasks encourages researchers to build deep architecture with powerful high performance hardware and larger datasets. With the help of high performance GPU clusters and data argumentation, people are more enthusiastic to explore bigger network on complex recognition problems without much interpretation which makes other researchers difficult to follow. In this paper, we try to apply the two kinds of deep CNN model, AlexNet and GoogLeNet, on a specific real learning problem, food recognition, and try to reveal some tricks in fine-tuning the existing CNN architecture on this problem.

II. Experimental Setup

In this section, we will discuss some details about the datasets and models used for our experiments.

A. Models

In this paper, AlexNet and GoogLeNet are their Caffe[9] implementation and all the results for a specific CNN architecture are obtained from single model.

AlexNet contains 5 layers followed by the auxiliary classifier which contains 2 fully connected layers (FC) and 1 softmax layer. Each of the first two layers can be subdivided into 3 components: convolutional layer with rectified linear units (ReLUs), local response normalization layer (LRN) and max pooling layer. Layer 3 and layer 4 contain just convolutional layer with ReLUs while layer 5 is similar to the first two layers except for the LRN. For each of the fully connected layer, 1 ReLUs and 1 dropout[10] layer are followed.

GoogLeNet shows another trend of deep CNN architecture with lots of small receptive fields. Figure 1 shows the architecture a inception cell. Inspired by [11], lots of 1×1 convolutional layers are used for computational efficiency. Another interesting feature of GoogLeNet is that there are two extra auxiliary classifiers in intermediate layers. During the training procedure, the loss of these two classifiers are counted into the total loss with a discount weight 0.3, in addition with the loss of the classifier on top. More architecture details can be found from [7].

B. Food Datasets

Besides ImageNet dataset, there are many popular benchmark datasets for image classification tasks such as Caltech dataset and CIFAR dataset, which contain hundreds of classes. However, in this paper, we try to focus on a more specific area, food classification. Compare to other classification tasks, there are some properties of the food (dishes) which make the tasks become a real challenge: i)food doesn't have any distinctive

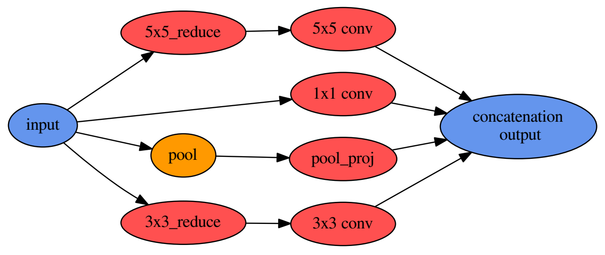

Fig. 1. Inception Cell. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 receptive field layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 receptive field after the MAX pooling layer. The output layer concatenate all its input layers.

spatial layout: for other tasks like scene recognition, we can always find some discriminative features such as buildings or trees, etc. ii) food class is a small sub-category among all the categories in daily life, so the inter-class variation is relatively small; on the other hand, the contour of a food varies depending on many aspects such as the point of the view or even its component. So these properties make food classification catastrophic for some recognition algorithms. Therefore, the training these two models on food classification task can reveal some important aspects of themselves and help us better understand their architecture. In this paper, we used two image datasets Food256[12][1] and Food101[13][2]. It is worthy to mention that PFID dataset is also a big public image database for classification, but their images are collected in a laboratory condition which is considerably not real enough.

Food-256 Dataset. This is a relatively small dataset containing 256 kinds of foods and 31644 images from various countries such as French, Italian, US, Chinese, Thai, Vietnamese, Japanese and Indonesia. The distribution among classes is not even and the biggest class (vegetable tempura) contains 731 images while the smallest one contains just 100 images. For this ”small” dataset, we randomly split the data into training and testing set, using 80% and 20% of the original data respectively and keep the class distribution in these two sets uniform. The collector of this dataset also provides boundary box for each image to separate different foods and our dataset is cropped according to these boundary boxes.

Food-101 Dataset. This dataset contains 101-class real-world food (dish) images which were taken and labeled manually. The total number of images is 101,000 and there are exactly 1000 images for each of the class. Also, each class has been divided into training and testing set containing 750 images and 250 images respectively by its collector. The testing set is well cleaned manually while the training set is not well cleaned on purpose. This noisy training set is more similar to our real recognition situation and it is also a good way to see the effect of the noise on these two models.

Data argumentation is an efficient way to enrich the data. There are also some techniques that can applied to enlarge the dataset such as *subsampling* and *mirroring*. The original images are firstly resized to 256×256 pixels. We crop the 4 corners and center for each image according to the input size of each model and flap them to obtain 10 crops. For the testing set, the prediction of the image is the average prediction of the 10 crops.

[1]Dataset can be found http://foodcam.mobi/dataset.html

[2]Dataset can be found http://www.vision.ee.ethz.ch/datasets_extra/food-101

III. Experimental Discuss

Training a CNN with millions of parameters on a small dataset could always lead to horrible overfitting. But the idea of supervised pre-training on some huge image datasets could preventing this problem in certain degree. Compare to other initialized strategies according to certain distributions, the pre-trained model is initialized according to the distribution of the specific task. Indeed, this initialization has certain bias as there is no single dataset including all the invariance for natural images[14], but this bias could be reduced as the pre-trained image dataset increases and the fine-tuning can be benefit from this initialization.

A. *Pre-training and Fine-tuning*

We conduct several experiments on both architectures and use different training initialization strategies for both Food-256 and Food-101 datasets. The scratch model is initialized with Gaussian distribution for AlexNet and Xavier algorithm[15], which automatically determines the scale of initialization based on the number of input and output neurons. These two initializations are used for training the model for the original ImageNet task. The pre-trained models and fine-tune models are initialized with the weights trained from ImageNet. For the pre-trained models, we just re-train the output layers while all the layers are re-trained for the fine-tune models.

TABLE I. Top-5 Accuracy in percent on fine-tuned, pre-trained and scratch model for two architectures

	AlexNet		GoogLeNet	
	Food-101	Food-256	Food-101	Food-256
Fine-tune	**88.12**	**85.59**	**93.51**	**90.66**
Pre-trained	76.49	79.26	82.84	83.77
Scratch	78.18	75.35	90.45	81.20

From Table I we can see that fine-tune from pre-trained model can boost the performance of the CNN for a specific task. It is interesting to see that for the Food-101 task, the accuracy of the scratch models outperforms the pre-trained models.

In Figure 2 we visualized the responses of the pre-trained GoogLeNet model and fined-tuned GoogLeNet model for the same input image for some layers. We can see that the responses of the lower layer are similar as the lower level features are similar. Then we can see that the decisions made by these two models is totally different. Since only the last layer (auxiliary classifier) of the pre-trained model is optimized, we can infer that the higher level features are more important which is consistent with our intuition. From Table I we can see that GoogLeNet always performances better than AlexNet on both datasets. This implies that the higher level features of GoogLeNet are more reasonable compared to AlexNet and this is due to the special architecture of its basic unit, Inception.

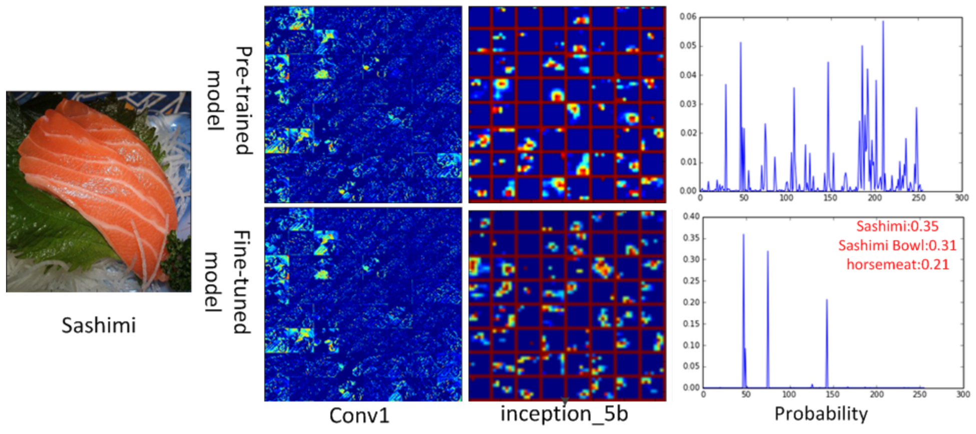

Fig. 2. Visualization of some responses of different GoogLeNet models in different layers for the same input image. Conv1 is the 1st convolutional layer and Inception_5a is the output of all convolutional layer.

Table III and II show the weights' cosine similarity of each layer between the fine-tuned models and their pre-trained models. From the results we can see that the weights in the low layer are more similar which implies that these two architectures can learn the hierarchical features as the low level features are similar for most of the tasks and the difference of the objects is determined by the combination of these low level features. From Table III, we can see that, in AlexNet the weights of the pre-trained and fine-tuned models are extremely similar. This can be caused by two reasons:

- Small receptive filed. Since ReLUs are used in Both architectures, vanishing gradients do not exist. Rectified activation function is mathematically given by

$$h = \max(w^T x, 0) = \begin{cases} w^T x & w^T x > 0 \\ 0 & else \end{cases} \quad (1)$$

  The ReLU is inactivated when its input is below 0 and its partial derivative is 0. Sparseness can boost the performance of the linear classifier on top, but on the other hand, sparseness will make the more difficult to train especially for fine-tuning. The derivative of the kernel is $\frac{\partial J}{\partial w} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w} = \frac{\partial J}{\partial y} * x$ where $\frac{\partial J}{\partial y}$ denotes the partial derivative of the activation function, $y = w^T x$ and x denotes the inputs of the layer. The sparseness of the input could lead to sparse kernel derivative. Therefore, the filters of the fine-tuned AlexNet is extremely similar. Compared to large receptive field used in AlexNet, GoogLeNet employs 2 additional $n \times n\_reduced$ convolutional layers before the 3×3 and 5×5 convolutional layers. Even though the most critical purpose of these two 1×1 convolutional layer is for computational efficiency, these 2 convolutional layers tend to squeeze the sparse input and generate a dense output as the input of the next layer.

- The pooling strategy. In AlexNet, max pooling is applied to all the pooling layers between several convolution layers and in back propagation, the max pooling layer will pass the error to the place where it came from. Since it only came from one place of the receptive field, the back propagation error is sparse and it keeps the kernels of the convolution layers stable. In GoogLeNet, even though, there is a max pooling layer in every inception, there are other 3 back propagation errors, from $5 \times 5\_reduce$ and $3 \times 3\_reduce$ that can affect the weights of the previous inception.

TABLE II. COSINE SIMILARITY OF THE INCEPTIONS BETWEEN FINE-TUNED MODELS AND SCRATCH MODEL FOR GOOGLENET

	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
food256						
inception_3a	0.72	0.72	0.64	0.67	0.73	0.69
inception_3b	0.59	0.64	0.53	0.70	0.60	0.56
inception_4a	0.46	0.53	0.54	0.50	0.67	0.38
inception_4b	0.55	0.58	0.63	0.52	0.69	0.41
inception_4c	0.63	0.64	0.63	0.57	0.68	0.52
inception_4d	0.60	0.62	0.60	0.58	0.68	0.50
inception_4e	0.60	0.61	0.67	0.61	0.68	0.50
inception_5a	0.51	0.53	0.58	0.48	0.60	0.39
inception_5b	0.40	0.44	0.50	0.41	0.59	0.40
food101						
inception_3a	0.71	0.72	0.63	0.67	0.73	0.68
inception_3b	0.56	0.63	0.50	0.71	0.60	0.53
inception_4a	0.43	0.50	0.50	0.47	0.62	0.36
inception_4b	0.48	0.52	0.57	0.50	0.67	0.35
inception_4c	0.57	0.61	0.59	0.53	0.63	0.47
inception_4d	0.54	0.58	0.53	0.54	0.64	0.44
inception_4e	0.53	0.54	0.61	0.55	0.62	0.42
inception_5a	0.43	0.47	0.53	0.45	0.57	0.34
inception_5b	0.36	0.39	0.46	0.38	0.52	0.37

TABLE III. COSINE SIMILARITY OF THE LAYERS BETWEEN FINE-TUNED MODELS AND SCRATCH MODEL FOR ALEXNET

	conv1	conv2	conv3	conv4	conv5	fc6	fc7
food256	0.997	0.987	0.976	0.976	0.978	0.936	0.923
food101	0.996	0.984	0.963	0.960	0.963	0.925	0.933

B. Training across the datasets

From the previous experiments we can see that pre-training on the ImageNet dataset can boost the performance of the deep convolutional neural network and the knowledge learned from the general recognition problem can be successfully transferred into our specific area. In this part, we will discuss the generalization ability within the our food recognition area. Zhou et al. trained AlexNet for Scene Recognition across two

TABLE IV. SPARSITY OF THE OUTPUT FOR EACH UNIT IN GOOGLENET INCEPTION FOR TRAINING DATA FROM FOOD101 IN PERCENT

	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
inception_3a	69.3 ± 1.3	69.6 ± 1.1	80.0 ± 1.0	64.1 ± 2.2	75.8 ± 1.6	76.2 ± 5.4
inception_3b	92.8 ± 0.9	76.5 ± 0.9	94.7 ± 0.9	71.6 ± 2.3	94.4 ± 0.5	94.7 ± 1.6
inception_4a	90.9 ± 0.9	70.0 ± 1.2	93.8 ± 1.1	63.3 ± 4.0	91.9 ± 1.8	95.1 ± 2.0
inception_4b	71.9 ± 1.6	67.5 ± 1.2	75.4 ± 1.0	58.5 ± 2.6	78.9 ± 1.6	85.6 ± 3.6
inception_4c	75.1 ± 2.4	72.6 ± 1.3	81.0 ± 2.0	66.3 ± 6.1	79.7 ± 3.6	88.1 ± 3.3
inception_4d	87.3 ± 2.7	78.0 ± 2.2	88.0 ± 1.6	67.9 ± 3.1	88.9 ± 2.8	93.0 ± 2.2
inception_4e	91.8 ± 1.1	62.3 ± 2.2	91.0 ± 2.5	49.5 ± 3.7	94.0 ± 1.0	92.3 ± 1.5
inception_5a	78.7 ± 1.6	66.5 ± 1.7	82.3 ± 2.6	59.9 ± 3.2	86.4 ± 2.3	87.1 ± 2.6
inception_5b	88.2 ± 2.3	86.8 ± 1.6	83.3 ± 4.4	84.0 ± 3.1	81.4 ± 5.3	94.7 ± 1.5

datasets with identical categories[16]. But for more complex situation, such as two similar datasets with different categories, we are very interested in exploring whether Deep CNN can still successfully handle. Therefore, we conduct the following experiment to stimulate a more complex real world problem: transferring the knowledge from the fine-tuned Food-101 model on Food-256 dataset and continue fine-tune it on Food-256. To make the experiment more practical, we limited the number of samples per category from Food-256 for training, because in practise, if we want to build a our own model from Deep CNN, the resource is limited and it is exhausted to collect hundreds of labeled images for each category.

The Food-101 and Food-256 datasets share about 46 categories of food even though the images in the same category may vary across these two datasets. The types of food in Food-101 are mainly western style while most types of food in Food-256 are typical Asian foods. We compared the top-5 accuracy of different size of subset of Food-256 on different pre-trained model and get Table V. The ImageNet columns denote the pre-trained model trained only on ImageNet images and the Food101_ft columns denote the pre-trained model trained on ImageNet images and then fine-tuned on Food-101. From the result of Table we can see that,

TABLE V. TOP5 ACCURACY FOR TRANSFERRING FROM FOOD101 TO SUBSET OF FOOD256

	AlexNet		GoogLeNet	
instances per class	ImageNet	Food101_ft	ImageNet	Food101_ft
20	68.80	**75.12**	74.54	**77.77**
30	73.15	**77.02**	79.21	**81.06**
40	76.04	**80.23**	81.76	**83.52**
50	78.90	**81.66**	84.22	**85.84**
all	85.59		90.66	

IV. CONCLUSION

The conclusion goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 818–833.

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

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference*, 2014.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[12] Y. Kawano and K. Yanai, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *Proc. of ECCV Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2014.

[13] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014.

[14] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 329–344.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[16] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 487–495. [Online]. Available: http://papers.nips.cc/paper/5349-learning-deep-features-for-scene-recognition-using-places-database.pdf