Domain Adaptation for Food Recognition with Deep Neural Netwrok

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*—**

I. Introduction

Recognizing the objects is the most fundamental function for human to understand the world, the whole procedure of which only takes a few tens of milliseconds for human brain. Recently, Convolutional Neural Network (CNN) shows its potential to replace the human engineered features, such as SIFT [1], SURF [2] and HOG [3] etc, in the large object recognition tasks[4][5][6]. In ILSVRC2014, the 1st prize winner, referred as GoogLeNet, achieved a 93.33% top-5 accuracy, almost as good as human annotation[7]. Unlike the local features such as SIFT or SURF, which present an shallow interpretation of spatial property, deep CNN can automatically learn top-down hierarchical feature representations. Therefore, deep CNN has been intensively used as the feature extractor for image recognition[8]. 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, continuingly expending web-based datasets such as ImageNet promises the researchers to utilize large amount of labeled images and train very deep CNNs. The truth that deep CNN outperforms other shallow models by a large margin in some real image recognition tasks encourages researchers to build deeper architecture with powerful high performance hardware and larger datasets.

Even though, these large scale web-based datasets contain almost any desired categories, it is still not possible for them to include images across all the domains that people may interest. Domain adaptation aims to build classifiers that are robust to mismatched data distributions. However, image recognition models trained from source domain are inherently biased due to the datasets. Previous empirical and theoretical studies have shown that the testing error is positively proportional to the difference between the test and training input distribution[9][10]. Many domain adaptation methods have been proposed to solve this bias, but most of them are limited to features extracted from shallow models[11][12][13]. Since deep CNN models are trained on these very large image datasets and can learn hierarchical features, they have strong generalization ability and can be applied in many other domains. Applying the pre-trained model from ImageNet dataset to other domains without taking advantage of domain adaptation shows some impressive results[14] [5]. Some work can be found for deep learning domain adaptation, but is limited to identical categories for the source and target domain[15][16]. To our best knowledge, almost none of the previous domain adaptation studies the problem while the target categories are different from the source.

In this paper, we first train the feature extractor with fine-tuned deep CNN on two food datasets. Our fine-tuned models achieve the state-of-the-art performance on both datasets and show that deep CNN can learn discriminative representations for food recognition task. We find that previous domain adaptation methods suffer when the categories in source domain and target domain are different.

II. Experimental Setup

In this section, we introduce some details about the two architectures and the food datasets used in our experiments.

A. Models

In this paper, AlexNet and GoogLeNet are their Caffe [17] implementations 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 [18] layer are followed.

GoogLeNet shows another trend of deep CNN architecture with lots of small receptive fields. There are 9 Inception modules in GoogLeNet and Figure 1 shows the architecture of a single inception module. Inspired by [19], 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].

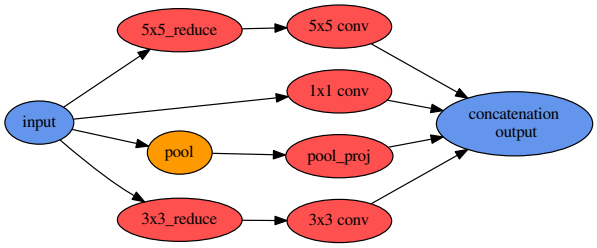

Fig. 1. Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input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

B. Food Datasets

Besides ImageNet dataset, there are many popular benchmark datasets for image recognition such as Caltech dataset and CIFAR dataset, both of which contain hundreds of classes. However, in this paper, we try to focus on a more specific area, food classification. Compared to other recognition tasks, there are some properties of the food (dishes) which make the tasks become a real challenge:

- Food doesn't have any distinctive spatial layout: for other tasks like scene recognition, we can always find some discriminative features such as buildings or trees, etc;
- 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 components.

These properties make food recognition catastrophic for some recognition algorithms. Therefore, the training these two architectures on the food recognition task can reveal some important aspects of themselves and help people better understand them. In this paper, we use two image datasets Food-256 [20][1] and Food-101 [21][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 applicable for real recognition task.

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 around 80% (25361 images) and 20% (6303 images) 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.

[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

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 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 architectures.

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 the 5 cropped images to obtain 10 crops. For the testing set, the prediction of an image is the average prediction of the 10 crops.

III. Experimental Discuss

Training a CNN with millions of parameters on a small dataset could easily lead to horrible overfitting. But the idea of supervised pre-training on some huge image datasets could preventing this problem in certain degree. Compared to other randomly initialized strategies with certain distribution, supervised pre-training is to initialize the weights according to the model trained from a specific task. Indeed, initialization with pre-trained model has certain bias as there is no single dataset including all the invariance for natural images [22], but this bias can be reduced as the pre-trained image dataset increases and the fine-tuning should be benefit from it.

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 models are initialized with Gaussian distribution for AlexNet and Xavier algorithm for GoogLeNet[23]. These two initializations are used for training the original models for the ImageNet task. The ft-last and fine-tuned models are initialized with the weights pre-trained from ImageNet dataset. For the ft-last model, we just re-train the fully connected layers while the whole network is fine-tuned for the fine-tune model.

TABLE I. Top-5 Accuracy in percent on fine-tuned, ft-last 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**
Ft-last	76.49	79.26	82.84	83.77
Scratch	78.18	75.35	90.45	81.20

TABLE II. Accuracy compared to other method on Food-256 dataset in percent

	fv+linear [24]	GoogLeNet	AlexNet
Top1	50.1	**70.13**	63.82
Top5	74.4	**90.66**	85.59

TABLE III. TOP-1 ACCURACY COMPARED TO OTHER METHODS ON FOOD-101 DATASET IN PERCENT

	RFDC[21]	MLDS(≈[25])	GoogLeNet	AlexNet
Top1 accuracy	50.76	42.63	**78.11**	66.40

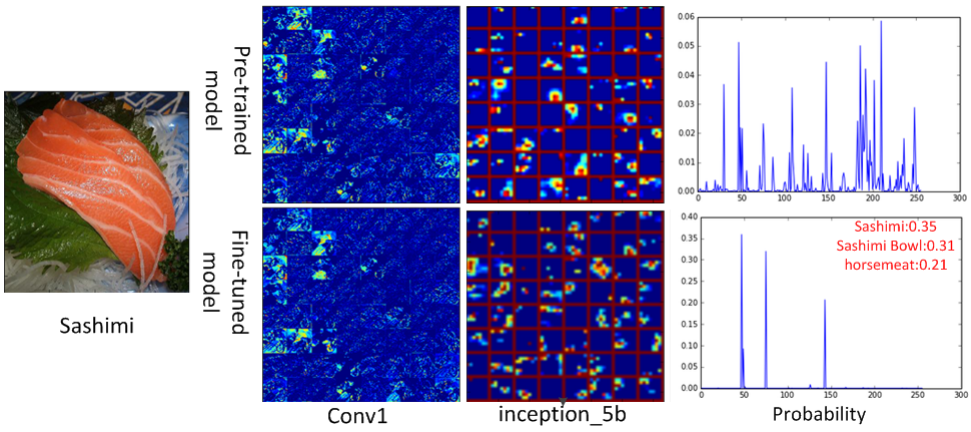

Fig. 2. Visualization of some feature maps of different GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

From Table I we can see that fine-tuning the whole network can improve the performance of the CNN for our task. Compared to other traditional computer vision methods (see Table II and III), GoogLeNet outperforms the other methods with large margins and we provide the state-of-the-art performance of these two food image datasets.

In Figure 2 we visualize the feature maps of the pre-trained GoogLeNet model and fined-tuned GoogLeNet model with the same input image for some layers. We can see that the feature maps of the lower layer are similar as the lower level features are similar for most recognition tasks. Then we can see that the feature maps in the high-level are different which leads to totally different recognition results. Since only the last layer (auxiliary classifier) of the ft-last model is optimized, we can infer that the higher level features are more important which is consistent with our intuition. Also from Table I, it is interesting to see that for the Food-101 task, the accuracy of the scratch models outperforms the pre-trained models. Since Food-101 is a relatively large dataset with 750 images per class while Food-256 dataset is an imbalanced small one, this indicates that it is difficult to obtain a good deep CNN model while the data is insufficient.

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 discriminative compared to AlexNet and this is due to the special architecture of its basic unit, Inception module. Table IV and V 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, the difference of the objects is determined by high-level ones which are the combination of these low level features. Also from Table V, we can observe that, the weights of the pre-trained and fine-tuned models are extremely similar in AlexNet . This can be caused by the size of 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 as well. Sparsity can improve the performance of the linear classifier on top, but on the other hand, sparse representations make the network more difficult to train as well as fine-tune. The derivative of the filter 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 sparse input could lead to sparse filter derivative for back propagation which would eventually prevent the errors passing down effectively. Therefore, the filters of the fine-tuned AlexNet is extremely similar. Compared to large receptive field used in AlexNet, the inception module in GoogLeNet employs 2 additional $n \times n\_reduced$ convolutional layers before the 3×3 and 5×5 convolutional layers (see Figure 1). Even though the original purpose of these two 1×1 convolutional layer is for computational efficiency, these 2 convolutional layers tend to squeeze their sparse inputs and generate a dense outputs for the following layer. We can see from Table VI that the sparsity of the $n\times n\_reduce$ layers are denser than other layers within the inception module. This makes the filters in the following layer more easily to be trained for transfer learning and generate efficient sparse representations.

The unique structure of the Inception module guarantees that the sparse outputs from previous layer can be squeezed with the 1×1 convolutional layers and feed to convolutional layers with bigger receptive field to generate sparser representation. The squeeze action promises the back propagation error can be transferred more efficiently and makes the whole network more flexible to fit different recogntion tasks.

TABLE IV. COSINE SIMILARITY OF THE LAYERS IN INCEPTION MODULES BETWEEN FINE-TUNED MODELS AND PRE-TRAINED MODEL FOR GOOGLENET

food256						
	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
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						
	1x1	3x3_reduce	3x3	5x5_reduce	5x5	pool_proj
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 V. COSINE SIMILARITY OF THE LAYERS BETWEEN FINE-TUNED MODELS AND PRE-TRAINED 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

TABLE VI. SPARSITY OF THE OUTPUT FOR EACH UNIT IN GOOGLENET INCEPTION MODULE 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

B. Learning across the datasets

From the previous experiments we can see that pre-training on the ImageNet dataset can improve the performance of the deep convolutional neural network on our specific area. In this part, we will discuss the generalization ability within the food recognition problem. Zhou et al. trained AlexNet for Scene Recognition across two datasets with identical categories [16]. But for more complex situation, such as two similar datasets with a little overlapped categories, we are very interested in exploring whether deep CNN can still successfully handle. Therefore, we conduct the following experiment to stimulate a more challenging real world problem: transferring the knowledge from the fine-tuned Food-101 model to a target set, Food-256 dataset. To make the experiment more practical, we limit the number of samples per category from Food-256 for training, because if we want to build a our own model using deep CNN for a specific task, the resource is always 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 compare the top-5 accuracy trained from different size of subset for Food-256 on different pre-trained model and the results are shown in Table VII. The ImageNet columns denote using the model pre-trained from ImageNet dataset as the pre-trained model and Food101_ft columns denote using the fine-tuned Food-101 model (the same one in Table I) as the pre-trained model.

From the result of Table VII we can see that, with this further transfer learning, both CNNs can achieve around 95% of the accuracy trained on full dataset while just utilizing about half of them (50 per class, 12800 of 25361 images). This indicates that when there is not enough labeled data, with its strong generalization ability, deep CNN trained from general task can still achieve satisfying result and perform even better when an additional relevant dataset is involved. This encouraging result may attract more people to use deep CNN for their specific task and continue to explore the potential of the existing architecture as well as designing new ones.

IV. DOMAIN ADAPTATION

In this section, we propose a framework for online adaptive learning for a few labeled data for both domain and target examples with deep CNN while the categories in source and

TABLE VII. TOP5 ACCURACY FOR TRANSFERRING FROM FOOD101 TO SUBSET OF FOOD256 IN PERCENT

	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	87.21	90.66	90.65

target domains are different. A typical approach for domain adaption is to fine-tuning the deep CNN model with these labeled examples with low learning rate. By taking advantage of the discriminative features learned by deep CNN, this approach achieved some impressive results[14] [5]. However, fine-tuning on deep CNN requires an ample amount of labeled target data and sometimes could degrade the performance when the labeled examples are scarce[15]. Other domain adaption methods such as Adaptive-SVM and PMT-SVM, are originally designed for learning across the similar categories and suffer from the mismatched distribution of the source and target domains, as the knowledge from source domain could be useless for target domain. In practice, people are more interested in learning the new categories in the target domain efficiently. As far as we know, few study has shown that there is an approach that can learn new categories with a few examples.

Chu et al. recently proposed a warm start approach for parameter selection in linear model. By iteratively updating the parameters from previous knowledge, the algorithm can search the optimal value for a specific task. Inspired by this, we propose a warm start adaptive learning algorithm that can adapt new categories from the target domain in an online learning scenario. Our method uses logistic regression to classify the representation obtained from deep CNN. For the new categories in target domain, rather than utilizing the parameters from source domain, we introduce another affiliated binary linear predictor pre-trained using all the examples as the negative class and warm start the pre-trained predictor while a new category comes.

V. CONCLUSION

In this paper, we compare two different deep convolutional neural network architectures and their transferring ability on food datasets. Both architectures show their potential on generalization ability and we provide the state-of-the-art on both Food-101 dataset and Food-256 dataset using fine-tuned GoogLeNet. GoogLeNet shows its strong ability on transferring the knowledge between different tasks with the help of the specially designed unit, Inception. We find that not only does the intensively used 1×1 convolutional layer in Inception reduce the computational cost, but it also helps to train the filters in the following layer. The filters after the 1×1 convolutional layer can be trained more efficiently while the 1×1 convolutional layer provides squeezed input and this helps GoogLeNet learn more complex high-level features. Moreover, in a more practical situation such as transfer learning within a specific area with a few labeled instances for the target set, both of these two architectures show some encouraging results in transferring knowledge across the dataset, reaching around 95% of the accuracy trained on full dataset with just half data.

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," *CoRR*, vol. abs/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] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1915–1929, 2013.

[9] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira *et al.*, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, p. 137, 2007.

[10] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, "Learning bounds for domain adaptation," in *Advances in neural information processing systems*, 2008, pp. 129–136.

[11] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[12] J. Yang, R. Yan, and A. G. Hauptmann, "Adapting svm classifiers to data with shifted distributions," in *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, 2007, pp. 69–76.

[13] Y. Aytar and A. Zisserman, "Tabula rasa: Model transfer for object category detection," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2252–2259.

[14] 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.

[15] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell, "One-shot adaptation of supervised deep convolutional models," *arXiv preprint arXiv:1312.6204*, 2013.

[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.

[17] 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.

[18] 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.

[19] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[20] 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.

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

[22] 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.

[23] 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.

[24] Y. Kawano and K. Yanai, "Foodcam-256: A large-scale real-time mobile food recognitionsystem employing high-dimensional features and compression of classifier weights," in *Proceedings of the ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 761–762.

[25] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 73–86.

[26] B.-Y. Chu, C.-H. Ho, C.-H. Tsai, C.-Y. Lin, and C.-J. Lin, "Warm start for parameter selection of linear classifiers."