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. Background

A. Food Datasets

In this paper, we use two image datasets Food-256 [17][1] and Food-101 [18][2] for our domain adaptation task. 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.

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

[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

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.

In our domain adaptation experiment, we use the categories from Food-101 dataset as our source domain and the categories from Food-256 dataset is the target domain.

B. *Deep Convolutional Neural Network*

There are some inherent properties of the food that makes our task challenging.

- 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. Learning useful representation merely for food recognition itself is a complex task.
- 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. Domain adaptation techniques could suffer from this and degrade the performance.

These properties make food recognition catastrophic for some recognition algorithms. Recognizing a image consists of two major parts: extracting features and predicting the label with some classifiers. In general, the first part is more important and training an efficient feature extractor could reduce the complexity of our task greatly. To achieve high accuracy for similar foods and adaptive for new foods, we require an efficient feature extractor that can learn both general and discriminative representations for our task. Deep CNN has achieved great progress in recent years and it is proved to learn hierarchical features for image recognition task[19]. Compared to those shallow methods, features extracted from deep CNN are discriminative with linear models. Therefore, we train our a deep CNN for our task in this paper and use the feature representations from deep CNN as the input for our online adaptive model. In Section III, we show the-state-of-art performance on both food datasets with deep CNN.

III. Training Deep Feature Extractor

In this section, we illustrate the architecture used for deep CNN and by utilizing the pre-trained model, we obtain efficient feature extractors and achieve the-state-of-the-art performance on GoogLeNet for both food datasets. Since the architecture of GoogLeNet is a recently discovered, there is not much work about its performance on other image dataset. We would like to discuss some features we find for GoogLeNet in comparison with AlexNet[4] in this section as well.

A. *Architecture of GoogLeNet*

The architecture of deep CNN is deterministic for the performance of the feature extractor. Instead of exploring our own deep CNN architecture, we decide to use the existing one, GoogLeNet. GoogLeNet achieved 93.33% top-5 accuracy on a 1000-category image recognition task which is very close to human annotation.

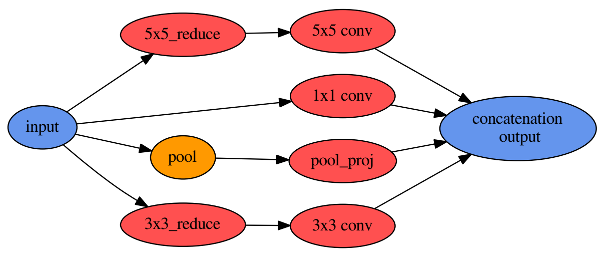

Fig. 1. Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input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

The architecture of GoogLeNet is unique. Inspired by [20], small 1×1 receptive field are intensively used throughout the network. There are 9 Inception modules in GoogLeNet and Figure 1 shows the architecture of a single inception module. 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. *Pre-training and Fine-tuning*

Even though we have two dataset containing large amount of images, it is still not sufficient to train a GoogLeNet well. Training a deep CNN with millions of parameters with insufficient data 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 [21], but this bias can be reduced as the pre-trained image dataset increases and the fine-tuning should be benefit from it.

We use the AlexNet, another architecture of deep CNN, as the baseline to illustrate the property of GoogLeNet. 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[22]. 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.

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.

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

	RFDC[18]	MLDS(≈[24])	GoogLeNet	AlexNet
Top1 accuracy	50.76	42.63	**78.11**	66.40

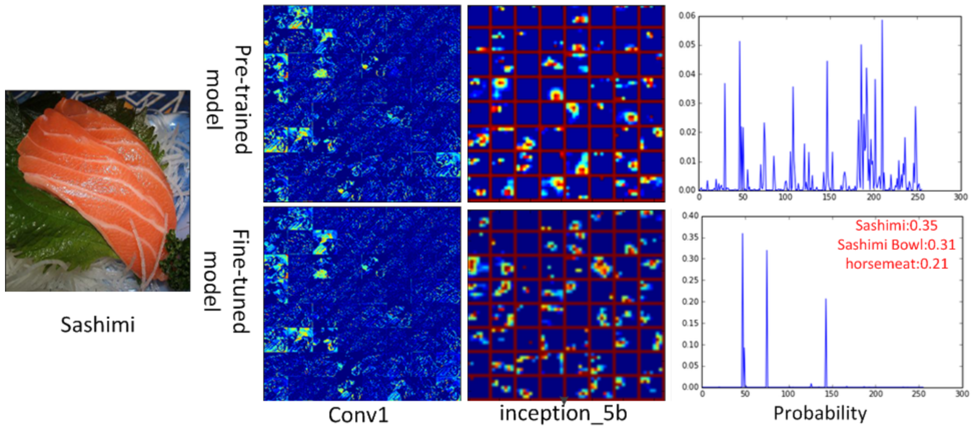

Fig. 2. Visualization of some feature maps of different GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

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 [23]	GoogLeNet	AlexNet
Top1	50.1	**70.13**	63.82
Top5	74.4	**90.66**	85.59

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:

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

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

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

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.

IV. DOMAIN ADAPTATION

For a real world recognition task, an algorithm that can continuously adaptive to new labels just like the human does, is always very attractive and practical. In this section, we propose a framework for online adaptive learning with a few labeled data in both domain and target examples using the features extracted from GoogLeNet. We design the following experiment, transferring from Food-101 dataset to Food-256 dataset to illustrate our method while just using a few examples. Since Food-101 has more images per category, we use it as the source domain and the fine-tuned GoogLeNet on Food-101 from Section III is used as the feature extractor to generate feature representations for both datasets. 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. To make this task more challenging, we also limited the number of examples in both source and target domain. Unlike those previous studies which only consider the performance within the target domain, since the two domains are within the general food domain, we also consider how our method performs on the target domain incorporating the source domain. The experiment results show that not only can our method adapt the new categories, but also it can keep a relatively high overall accuracy for both source and target domains. For the rest part of this section, we first discuss the limitation of some previous domain adaptation approaches for our task, then we introduce our method and show the improved performance on the same task.

A. *Limits of previous approaches*

From previous studies, there are two kinds of approach to solve our task. The first approach is to fine-tuning the deep CNN with the target examples incorporating with the sources ones. Fine-tuning the deep CNN model focuses on learning good feature representations from the images and using linear models for classification. From Section III, we successfully use this approach to transfer the knowledge from a general domain to our food domain with impressive results. Indeed, deep CNN can learn discriminative features and by taking advantage of this, this approach achieved some impressive results from previous studies[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]. There are many hyperparameters that affect the performance of deep CNN and fine-tuning it on a sparse label condition can lead to horrible overfitting. Apart from its sensitivity to the hyperparameters, fine-tuning the whole network requires intensive computational resources which makes it inappropriate for online learning.

Rather than learning efficient representations, another typical approach are more focused on dealing with the representation learned from conventional feature extraction methods for domain adaptation. Supervised domain adaptation models, such as Adaptive-SVM (A-SVM) and PMT-SVM, try to utilize the knowledge from source domain and apply to target domain[12][13]. These methods are limited in our task for the following reasons: all these methods try to find the similarity between the source and target domain. Indeed, they show some good performance when these two domains have many overlapped or similar categories. From empirical experiments, we find that they suffer when the target domain is different from the source one. Table VII shows some empirical experiment results for two typical domain adaptation methods in a binary classification scenario. We manually choose the similar/identical source categories for the target ones if possible. For the those categories which we fail to find even similar category in source domain, we just choose the category whose representations are most similar to the target one measured by cosine similarity. The parameters used in these methods are set as the default because we think that tuned the parameters for these methods won't improve the performance very much for this experiment. From the result of our simple experiment, we can see that A-SVM and PMT-SVM suffers from choosing the appropriate domain categories for new categories in target domain. And for our task which is a multi-class situation and more complicated than this, the performance of these methods could be even worse.

TABLE VII. AVERAGE PRECISION FOR A-SVM AND PMT-SVM. SOURCE CATEGORIES WITHOUT * ARE DETERMINED BY COSINE SIMILARITY

Target category from food 256	Source category from food 101	A-SVM	PMT-SVM
Doughnut	Doughnut	0.4771	0.4735
Caesar salad	Caesar salad	0.9488	0.9496
Rice	Fried rice	0.8118	0.7932
Oatmeal	French onion soup*	0.3257	0.4100
pork cutlet	French fries*	0.0446	0.0381

As far as we know, few study has shown that there is an approach that can solve our task.

B. our method

Chu et al. recently proposed a warm start approach for parameter selection in linear model and by iteratively updating the parameters optimized from previous knowledge, the algorithm can search the optimal value for a specific task[25]. 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 representations obtained from deep CNN. For the new category in target domain, rather than utilizing the parameters from source domain, we employ another negative binary predictor pre-trained using all the examples as the negative class and warm start the pre-trained predictor with the parameters of this negative predictor for a new category. This approach can be extended into an online domain adaptation scenario when the algorithm just adopts one new category each time for multi-class target domain situation. Considering M categories in the source domain S and a new category t from target domain T, we can train M binary predictors $F = \{f(w_i, b_i)\}_{i=1}^{M}$ for each category in S and the negative predictor $\hat{f} = f(\hat{w}, \hat{b})$ using all the examples in $\mathcal{P}^- = S$ as negative examples. The predictor f_t for the new category t is initialized with $\left\{\hat{w}, \hat{b}\right\}$ and trained with $\mathcal{P}^- \bigcup \mathcal{P}^+$ while $\mathcal{P}^+ = t$. Then we update the negative predictor $\hat{f}$ with negative examples $\mathcal{P}^- = S \bigcup t$. The complete strategy for our method is given in Algorithm 1.

Algorithm 1 Complete algorithm of warm start online adaptation

```
Input: Source domain $S = \{s_i | i = 1, ..M\}$, Target domain
    $T = \{t_j | j = 1, ..N\}$, Classifier $F = \{\emptyset\}$
Output: $F$
 1: for all $i \in M$ do
 2:     $\mathcal{P}^+ \leftarrow s_i, \mathcal{P}^- \leftarrow S - s_i$
        Train $f_i(w_i, b_i)$ with $\mathcal{P}^+ \bigcup \mathcal{P}^-$, $F \leftarrow F \bigcup f_i$
 3: end for
 4: Training negative $\hat{f}\left(\hat{w_i}, \hat{b_i}\right)$ with $\mathcal{P}^- = S$
 5: while $t_j \notin S$ do
 6:     for all $i \in M$ do
 7:         $\mathcal{P}^+ \leftarrow s_i, \mathcal{P}^- \leftarrow S - s_i + t_j$
            Update $f_i(w_i, b_i)$ with $\mathcal{P}^+ \bigcup \mathcal{P}^-$
 8:     end for
 9:     Initialize $f_j$ with $(\hat{w}, \hat{b})$ and train with .
10:     $F \leftarrow F \bigcup f_j$
11:     $S \leftarrow S \bigcup t_j, M \leftarrow M + 1$
12:     Update $\hat{f}$ with $\mathcal{P}^- = S + t_j$
13: end while
```

In our task, compared to those methods using the previous parameters directly, the warm start can choose a better initialization for the new category. Besides that, our method warm start strategy gets rid of the constraint of We use Stochastic Gradient Descent to update the parameters for all the binary predictors as well as the negative one.

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

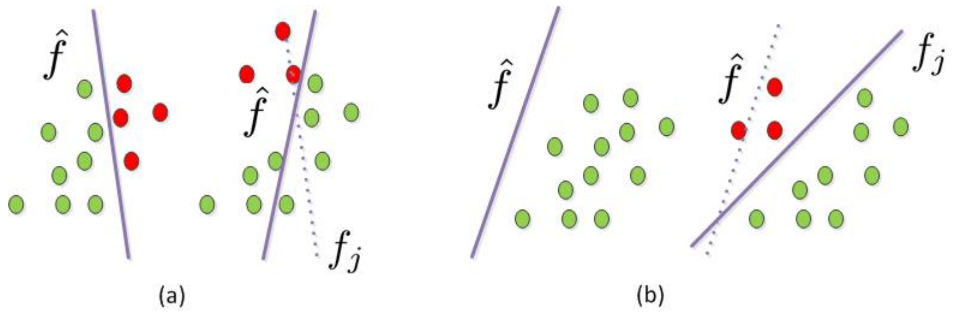

Fig. 3. (a) Conventional adaptation method fail to find good initialization for the new category from source domain. (b) The parameters from $\hat{f}$ are more adaptive for the new categories.

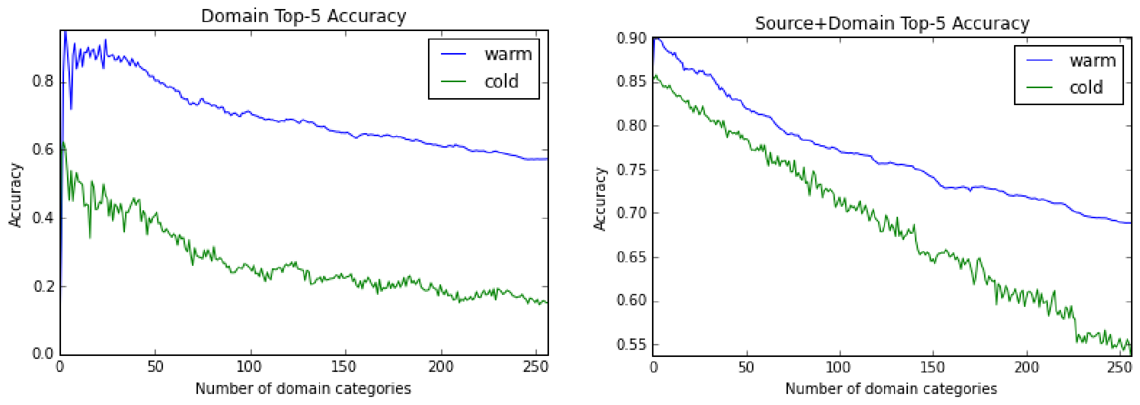

Fig. 4. Warm start Vs Cold Start on Top-5 accuracy for limited iterations

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. 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.

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

[19] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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