Adapting New Categories for Food Recognition with Deep Representation

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*—Deep Convolutional Neural Network (CNN) has recently achieved great successes in large scale object recognition competitions and achieved the performance as good as human annotation. However, training deep CNN requires an ample amount of labeled examples and those large scale database could not include all the categories people desire. Many domain adaptation techniques have been proposed to learn categories across domain, but limited to either identical/similar categories or insufficient data. We find that when labeled examples are sparse, some previous methods suffer with deep representation. In this paper, we first train our efficient feature extractor, fine-tuning GoogLeNet on Food-101 dataset and achieve state-of-the-art performance. Then we propose a method that can learn new categories from target domain with deep representations and warm start technique. Our method divides the multi new category learning problem into several steps and learns a single category in each step with warm start. Compared to cold start, in each learning step, classifiers with warm start can achieve a better accuracy. Benefitting from the parameters warm started from previous step, our method can achieve better overall result with a few training iteration.**

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 large deep CNN on real recognition problem is always a complicated task. The model contains hundreds of millions of parameters and there are lots of hyper-parameters that can affect its performance. However, continuingly expending web-based datasets such as ImageNet promise 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. 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 as well as learning identical or similar categories across domains[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]. Moreover, when the labeled examples are sparse, it is very difficult to train a efficient deep neural network. To our best knowledge, little the previous work studies the problem while the target categories are different from the source.

In this paper, we propose an incremental adaptive approach with warm start technique and empirical experiments show that our method can learn new categories with deep representations and achieve better result. To generate deep representation from images, we first train the feature extractor with fine-tuned GoogLeNet on Food-101 dataset and achieve the state-of-the-art performance. We find that previous domain adaptation methods suffer as the difference between target and source domain increases. Benefitting from warm start parameters in each step, out method can achieve better result compared to cold start. Moreover, warm start parameters converges better with a few training iterations compared to cold start.

The rest of this paper is organized as follow: In Section II we introduce the two datasets and discuss some properties about the food recognition task. In order to obtain discriminative representations from images, we fine-tuned GoogLeNet with pre-trained parameters from ImageNet and achieve state-of-the-art performance on our source domain, Food-101 dataset in Section III. We discuss the limitations of some previous adaptation methods and propose our warm start adaptation

method for learning new categories in Section IV.

II. Background

In this section, we introduce the two food dataset used in this paper and discuss the challenges of food recognition task.

A. Food Datasets

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

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. Since this is a big dataset with even distribution for each category, it is used as the source dataset as well as to train the feature extractor.

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. Since this dataset contains more categories and less images, it is used as our target dataset.

B. Challenges of food recognition task

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 category is a small sub-category among all the general 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][4][20]. Compared to other shallow methods, features extracted from deep CNN can be distinguished by linear model. Therefore, we train our a deep CNN as our feature extractor in this paper and use the deep feature representations as the input of our method. In Section III, we show the-state-of-art performance on Food-101 datasets with deep CNN.

[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

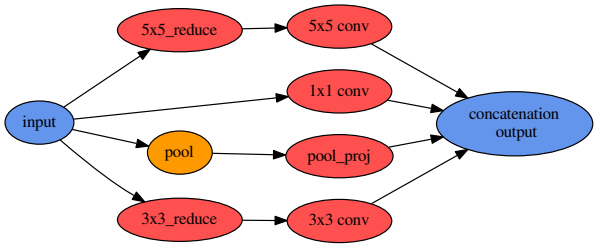

Fig. 1. Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n\times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input layers.

III. Deep Feature Extractor

In this section, we illustrate the architecture used for deep CNN and by utilizing the pre-trained model. We obtain an efficient feature extractor by fine-tuning the GoogLeNet incorporating the data from ImageNet and achieve the-state-of-the-art performance on Food-101 dataset. 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 can greatly 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 and we believe that its architecture can help us extract efficient deep feature representations for our adaptation experiment.

The architecture of GoogLeNet is unique to other deep CNN. Inspired by [21], 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 Food-101 dataset contains large amount of images, it is still not sufficient to train the 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 [22], 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 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 FOR DIFFERENT DEEP CNN ARCHITECTURES

	Fine-tune	Ft-last	Scratch
GoogLeNet	**93.51**	82.84	90.45
AlexNet	**88.12**	78.18	76.49

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), GoogLeNet outperforms the other methods with large margins and we provide the state-of-the-art performance on this datasets. In Section IV, the feature representations extracted from fine-tuned model are used as the input for our online domain adaptation method.

C. Discussion of the unique architecture of GoogLeNet

In the last part, we show the state-of-the-art performance for GoogLeNet on Food-101 dataset. Since few work has been found to discuss the architecture of GoogLeNet, we would like to show some interesting findings from our empirical study.

Instead of training a classifier for image recognition with deep CNN, we utilize deep CNN as the feature extractor for our adaptation task. Indeed, the classifier can take great advantage of discriminative feature representations for different categories. From our empirical study, we find that benefitting from the the unique architecture of Inception module in GoogLeNet, GoogLeNet can learn the feature representations in a efficient way while other architectures may suffer from vanished gradient a lot.

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 even though Food-101 is a relative large dataset with 1000 examples per category, the model can still takes advantage of the initialization from ImageNet dataset.

From Table I we can see that GoogLeNet outperforms AlexNet which implies that the higher level features of GoogLeNet are more discriminative compared to AlexNet, and we believe that this is due to the special architecture of its basic unit, Inception module. Table III and IV 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. From Table IV, we can observe that the weights of the pre-trained and fine-tuned models are extremely similar in AlexNet. This indicates that even though ReLUs are used in this architectures, AlexNet still suffers from vanished gradient. However, from Table III we can see that, fined-tuned GoogLeNet which is significantly deeper than AlexNet suffers less. So we believe there Inception module in GoogLeNet can prevent vanished gradient.

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 V 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 within the 1×1 convolutional layers and field to generate sparser representation in the next layer. The squeeze action promises the back propagation error can be transferred more efficiently and makes the whole network more flexible to fit different recognition tasks.

TABLE II. 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

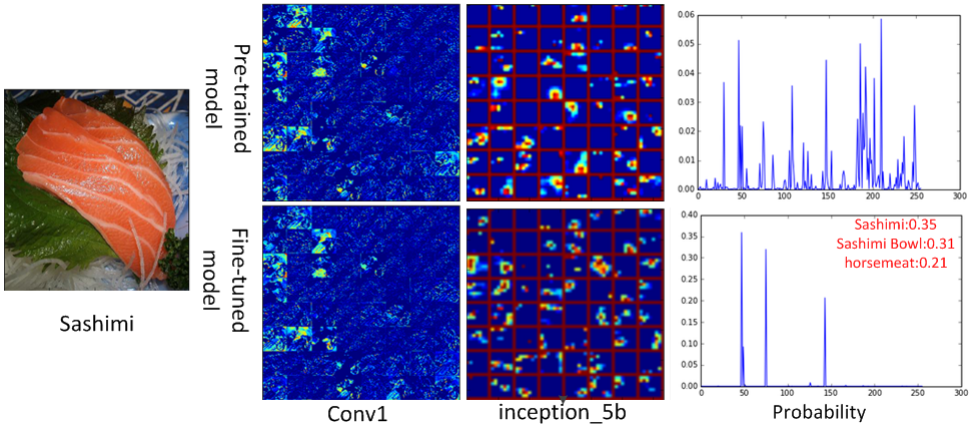

Fig. 2. Visualization of some feature maps of different GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

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

	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 IV. COSINE SIMILARITY OF THE LAYERS BETWEEN FINE-TUNED MODELS AND PRE-TRAINED MODEL FOR ALEXNET

	conv1	conv2	conv3	conv4	conv5	fc6	fc7
food101	0.996	0.984	0.963	0.960	0.963	0.925	0.933

IV. WARM START DOMAIN ADAPTATION FOR LEARNING NEW CATEGORIES

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 an adaptive method for learning new categories with a few target examples using the feature representations from GoogLeNet. By warm start the parameters of the classifiers, our method can effectively learning new categories. The experiment results show that our method is able to adapt new categories with just a few examples and outperforms the method that learning directly with a large margin. 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 effectively 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 mod-

TABLE V. 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

els, such as Adaptive-SVM (A-SVM) and PMT-SVM, try to utilize the knowledge from source domain and apply to target domain and hypothesize that there should be a relatively strong relationship between the categories among these two domains[12][13]. These methods are limited in our task for the reasons that all these methods try to find the similarity between the source and target domain. Indeed, they work well when these two domains have many overlapped or similar categories. However, sparse representations from deep CNN from which the linear classifiers would eventually benefit indicates that the distance of the representations between different categories could be very large.

From empirical experiments, we find that adaptive methods suffer as the difference of source and target categories increases. Table VI 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 worse than training without any of these adaptive methods.

As far as we know, few study has shown that there is an approach that can solve our task. Therefore, we propose an incremental adaptive learning algorithm to solve this problem.

B. Incremental adaptive learning for new category

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 and their 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 incremental learning mode. Instead of learning the whole target categories directly, our method adapts one category each time and iteratively updates all the parameters. Our method uses logistic regression to classify the representations obtained from deep CNN since compared to deep models, linear model can be more easily trained with just a few examples in each category. For learning a new category, rather than utilizing the parameters from source domain, we employ another negative binary classifier pre-trained with all the examples from other categories as the negative class and warm start the classifier with the parameters of the negative classifier for the new category. This approach can be extended into an incremental domain adaptation scenario when the algorithm just adapts one category in each step for multi-class situation. For each step, considering M categories in the source domain S and a new category t from target domain T, there are M binary classifiers $F = \{f(w_i, b_i)\}_{i=1}^{M}$ for each category in S and the negative classifier $\hat{f} = f(\hat{w}, \hat{b})$ using all the examples in $\mathcal{P}^- = S$ as negative examples. The classifier 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 classifier $\hat{f}$ with negative examples $\mathcal{P}^- = S \bigcup t$. The complete strategy for our method is given in Algorithm 1. In our task, compared to those methods using the previous parameters directly, the warm start can start with a better initialization for the new category and thus can be more adaptive for new categories. We use Stochastic Gradient Descent (SGD) to update the parameters for all the binary classifiers as well as the negative classifier.

Algorithm 1 Complete algorithm of warm start adaptation

Input: Source domain $S = \{s_i | i = 1, ..M\}$, Target domain $T = \{t_j | j = 1, ..N\}$, Classifier $F = \{\emptyset\}$
Output: F

for all $i \in M$ **do**
  $\mathcal{P}^+ \leftarrow s_i, \mathcal{P}^- \leftarrow S - s_i$
  Train $f_i(w_i, b_i)$ with $\mathcal{P}^+ \bigcup \mathcal{P}^-$, $F \leftarrow F \bigcup f_i$
end for
Training negative $\hat{f}\left(\hat{w_i}, \hat{b_i}\right)$ with $\mathcal{P}^- = S$
while $t_j \notin S$ **do**
  for all $i \in M$ **do**
    $\mathcal{P}^+ \leftarrow s_i, \mathcal{P}^- \leftarrow S - s_i + t_j$
    Update $f_i(w_i, b_i)$ with $\mathcal{P}^+ \bigcup \mathcal{P}^-$
  end for
  Initialize f_j with $(\hat{w}, \hat{b})$ and train with .
  $F \leftarrow F \bigcup f_j$
  $S \leftarrow S \bigcup t_j, M \leftarrow M + 1$
  Update $\hat{f}$ with $\mathcal{P}^- = S + t_j$
end while

C. Experiment setup & Evaluation

To illustrate our method, we design the following experiment, transferring from Food-101 dataset to Food-256 dataset 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

TABLE VI. AVERAGE PRECISION FOR A-SVM, PMT-SVM, SVM AND LOGISTIC REGRESSION. SOURCE CATEGORIES WITHOUT * ARE DETERMINED BY COSINE SIMILARITY

Category from food 256	Category from food 101	A-SVM	PMT-SVM	SVM	LR
Doughnut	Doughnut	0.4771	0.4735	0.4781	0.4554
Caesar salad	Caesar salad	0.9488	0.9496	0.9498	0.9486
White rice	Fried rice	0.8118	0.7932	0.7932	0.9004
Oatmeal	French onion soup*	0.0345	0.0381	0.0347	0.0454
Pork cutlet	French fries*	0.0446	0.0381	0.0450	0.0837

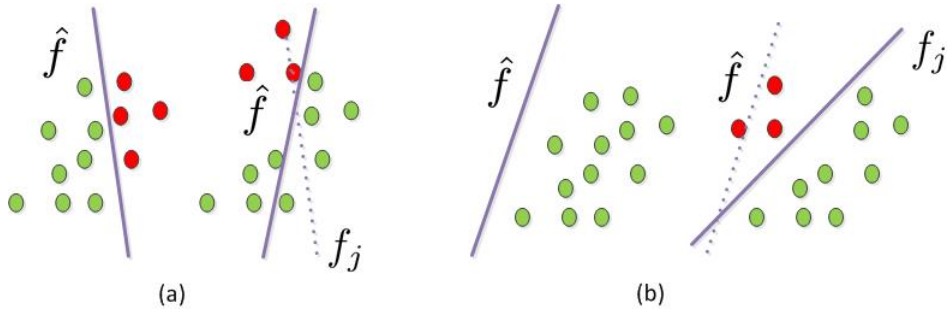

Fig. 3. (a) Conventional adaptation method fail to find good initialization for the new category from source domain. (b) The parameters from $\hat{f}$ are more adaptive for the new categories.

as the feature extractor to generate feature representations for both datasets and deep feature representations from *Pool5*, the layer before auxiliary linear classifier, are used as the inputs of our method.

The types of food in Food-101 are mainly western style while most types of food in Food-256 are typical Asian foods. They share about 36 categories even though the images in the same category may vary across these two datasets. These 36 duplicated categories are removed as noisy categories, so there are 220 new categories left in Food-256 dataset. To make this task more challenging, we also limited the number of examples in both source and target domain.

Baseline: Instead of using the supervised adaptation techniques, such as A-SVM, we use cold start which randomly initializes the parameters for all the classifiers, as our baseline, because from Table VI we can see when adapting new categories, SVM and LR without any adaptation technique show similar results to those adaptive methods. The hypothesis of these adaptive methods limits their ability to adapt new categories as we discuss above.

n shorts: In our experiments, the training set of n shots contains n randomly picked examples from each category in both Food-101 and Food-256 datasets and the test set contains all the rest examples in Food-256 only.

For our SGD, we use effective learn rate following polynomial decay, to be 0 at max iteration and set base learning rate to 0.01. The initialization of the classifiers for source domain are trained with all the examples from source domain and negative classifier are initialized the same way as well. Unlike the other studies which consider the performance within the target domain separately, since the categories in our two datasets are in the general food category we would rather considering to combine them together as a super-domain and evaluate the performance for target categories on this super-domain.

D. From 101 to 102 categories

When there are multiple categories in target domain, our method divides the learning procedure into several steps and adapts one category in each step, updating all the parameters of the binary logistic regression classifiers. So for the experiment adapting Food-256 from Food-101, the algorithm starts from a 101 to 102 categories situation. We conduct the following experiment: we try to add an arbitrary category from Food-256 to Food-101 and evaluate accuracy of the new category in the super-domain containing 102 categories. We go through all the 220 categories in Food-256 in each round and Table VII shows the average performance of 10 rounds experiments.

TABLE VII. ACCURACY FOR A SINGLE NEW TARGET CATEGORY IN M+1 EXPERIMENT. AVERAGE TOP-5 ACCURACIES FOR SOME CATEGORIES ARE SHOWN. LAST ROW SHOWS THE AVERAGE RESULTS FOR ALL CATEGORIES.

	5 shots		1 shot	
target category	Warm	Cold	Warm	Cold
crape	**84.16**	62.29	**56.20**	28.00
chip butty	**80.97**	65.82	**55.03**	37.40
meat loaf	**67.78**	53.15	**68.21**	56.07
dried fish	**92.00**	79.81	**83.85**	71.19
scrambled egg	**75.21**	63.54	**59.00**	43.20
pork belly	**81.76**	70.59	**53.21**	32.45
Overall average	**91.91**	88.82	**80.77**	71.78

From Table VII we can see that by taking advantage of the warm start, the warm start does a slightly better than cold start in M+1 experiment since the initialization difference between these two methods are too small. We believe the margin of these two methods would increase as more new categories are learned from warm start.

E. From 101 to 101+220 categories

In this part, we show the performance of warm start in multiple new categories situation. From the experiments above, warm start shows some improvement on cold start and

experiment shows that the improvement can be accumulated as more categories are learned. Because the parameters of the negative classifier can affect the whole procedure greatly and randomly picking only one example from each category can be significantly bias, we fail to get a convergent result for both warm and cold starts in one short experiment. From empirical experiments we find that in order to get a convergent result, we have to pick at least 5 examples in each category. So we only show the performance of five shots for this experiment. In each step, the new category is determined by its original class index in Food-256. We use run SGD 50 iterations for each step and run this experiment 10 times as well. Figure 4 shows the average results in 5 short for both warm and cold start in each step.

From Figure 4 we can see that benefitting from warm start, classifiers can accumulate the information from previous steps and thus the margin between warm and cold starts increases as more new categories are learned. Indeed, cold start may not converge well for just 50 iterations. We believe that given enough training iteration, warm start can still converge to a better result (see Figure 5). We compare the results for different iterations and the average performance of 10 experiments are shown in Table VIII. The results of cold start in 10 and 20 iterations experiments, which are not even significantly better than random guess, are ignored. We observe that increasing training iteration won't help improve the performance of warm start very much as it converges very fast by taking advantage of better initialization.

V. Conclusion

In this paper, we propose a method that can adapt new categories using warm start parameters and deep feature representations. In order to obtain efficient feature representations, we first train a deep neural network as our feature extractor. Compared to previous shallow methods, our deep CNN model achieves the state-of-the-art performance on Food-101 dataset. Unlike many previous methods that can just adapt identical/similar categories between domains, our method can iteratively learn new categories. By learning one new category and warm start with the parameters from negative classifier in each step, our method shows improved result compared to training with the examples from new categories directly.

Our method can be a very efficient method when the computational resource is limited, such as mobile device etc. The idea of warm start can achieve a better result with just a few training iterations. Our future work can be a natural extension of this work: learning efficient shallow model from deep representations when the target examples are limited, such as one shot learning etc. In this scenario, randomly pick few examples from each categories could not be the best solution. Sophisticated pooling strategy could be a possible way to best eliminate the bias of data size.

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] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI*, 2011, pp. 1237–1242.

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

[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.

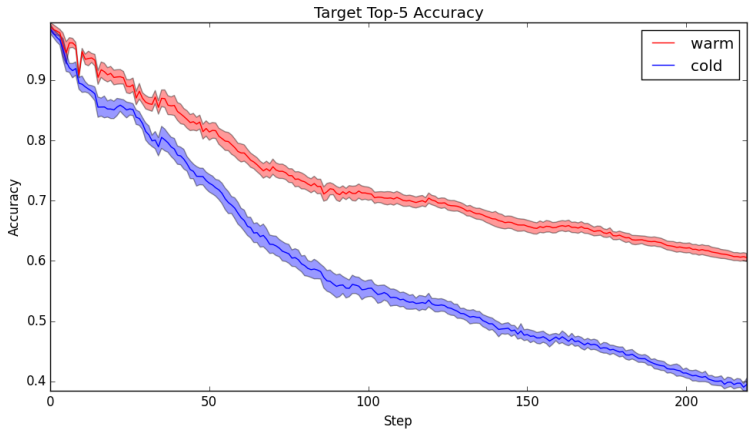

Fig. 4. Top-5 accuracy curve for categories in Food-256 in super-domain with 5 shots. Mean and standard deviation are shown.

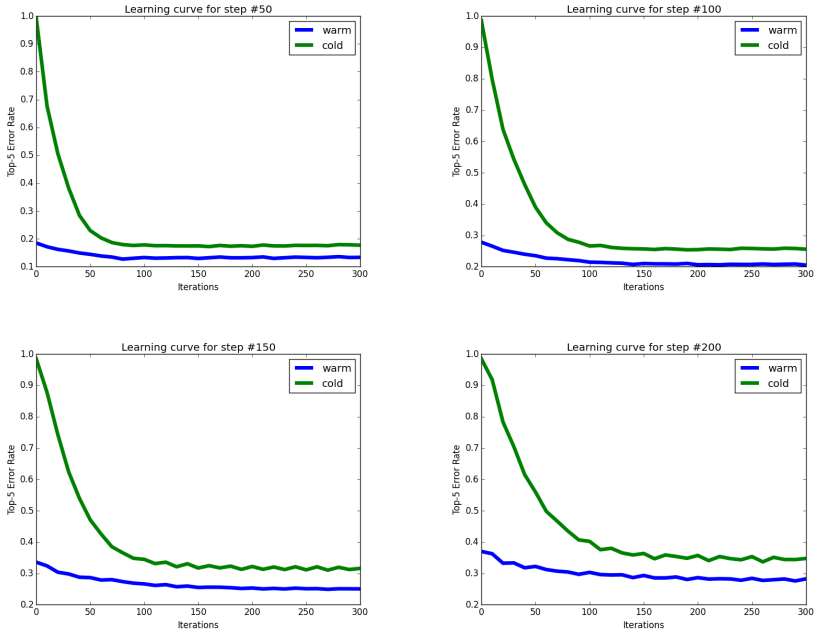

Fig. 5. Overall learning curve in some steps for 300 iterations. **Observation:** even training with enough iteration, warm start can still outperform cold start in a single step.

TABLE VIII. TOP-5 ACCURACY FOR 220 NEW CATEGORIES IN SUPER-DOMAIN WITH DIFFERENT TRAINING ITERATION IN EACH STEP.

	10 iterations	20 iterations	50 Iterations		300 iterations	
	warm start	warm start	warm start	cold start	warm start	cold start
Top-5	60.37± 0.58	60.54±0.32	60.33±0.62	39.10±0.66	**61.48±0.59**	55.35±0.48

[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."