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*—Learning to classify new categories from a different domain is always an interesting and challenging task in data mining. The classifier could suffer from dataset bias when predictingp new categories from the target domain. To adjust the bias, previous domain adaptation approaches are limited to either similar categories or requiring a large number of labeled examples from target domain. In this paper, we propose a new method that can alleviate dataset bias for food image recognition and adapt new food categories with fewer examples. To obtain less biased feature representation from food images, we fine-tuned GoogLeNet as our deep feature extractor. Using deep representation, our method can learn efficient classifiers with fewer labeled examples. We employ a negative classifier rejecting all learned categories and assume that the classifier of the new category can achieve better result initialized with the parameters of this negative classifier, referred as ”warm-start”. To adapt multiple new categories, our algorithm adapts one new category each time and iteratively warm start a new classifier. Experiment results show that warm start can achieve better accuracy compared to cold start (random initialization) for the multi-new category food recognition task.**

I. Introduction

Learning to classify new categories from different domains is always an interesting and challenging topic in data mining. Previous empirical and theoretical studies have shown that the testing error is positively proportional to the difference between the test and training input distributions [1] [2]. Therefore, mismatched data distribution can be a hug problem for predicting the data from a different domain. Domain adaptation is proposed to solve this problem, transferring the knowledge from the source domain to target ones. For image recognition tasks, the model trained from the source domain is inherently biased as there is no database that includes all the transformations of the images.

Recently, Convolutional Neural Network (CNN) shows its potential to replace the shallow features, such as SIFT [3], SURF [4] and HOG [5] etc, in large object recognition tasks [6] [7] [8]. Unlike the local feature which presents 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 [9]. As deep feature representation has strong generalization ability, it can compensate for the data bias and be applied for domain adaptation. Fine-tuning the whole network with the pre-trained models on target domain directly has shown some impressive results from previous studies [10] [7] [11] [12]. However, fine-tuning large deep CNN with limited training data could lead to overfitting which is mostly due to the sampling noise [13].

Even though, it is difficult to obtain a satisfied result while fine-tuning the whole network for deep CNN on a small target domain dataset, there is no limitation for training linear shallow models with the deep features for target domain. Fine-grained feature representation can be obtained by fine-tuning deep CNN on a relatively large source domain and alleviate the distribution bias for target domain [14]. Moreover, our empirical experiments show that fine-grained deep representation can also help classifier learn categories with fewer examples. So by applying appropriate adaptation strategy on shallow model, we can learn new category with fewer examples. From previous s-tudies, many shallow domain adaptation methods for the image recognition task have been proposed to compensate the dataset bias, assuming that there should be a linear correlation of the parameters for similar categories from two domains [15] [16] [17]. However, working with shallow feature representation such as HOG or SIFT etc, these methods require the target category to be geometrically similar to source category [17]. So their shallow feature representations should be similar and the linear assumption of the parameters can be applied directly for adaptation. But according to our experiments (see SectionIV), this assumption could fail when the representations of two categories are not similar. Deep CNN can learn fine-grained feature representations and the feature representations of two similar categories are not necessarily to be close to each other. The linear correlation assumption may fail when using the deep representation. Therefore, the parameters of classifiers for different categories should be more dissimilar for deep representation compared to shallow ones.

Instead of finding linear correlated parameters for the new category from learned categories, we employ a negative classifier that rejects all learned categories and assume that warm start the new classifier with parameters of this negative classifier can achieve better result. The idea of ”warm start” has been widely used for linear optimization problem, where the algorithm iteratively initializes and updates the parameters from the result of previous steps [18] [19] [20] [21]. In our work, the warm start is used to initialize the classifier for the new category with the parameters from a pre-trained negative classifier that predicts all learned categories as negative examples.

In this paper, we aim the adaptation problem into a specific scenario, food recognition. We first obtained a model that can learn 101 kinds of food from deep representation and try to solve the problem that adapts new foods from other dataset. We propose an incremental adaptive approach that can learn new foods with a few labeled images. There are two main contributions of our work:

1) Training deep feature extractor. To generate fine-grained deep representation from images, we first train the efficient feature extractor with fine-tuned GoogLeNet and achieve the state-of-the-art performance on our source domain, Food-101 dataset. Compared to the results training on full dataset, we can achieve similar results while just using a few examples in each category with deep representation.
2) Negative classifier for warm start. We find that previous domain adaptation methods suffer as the difference between target and source domain increases. Warm start new classifier with negative classifier could take advantage of deep representation. Benefitting from warm start parameters in each step, empirical experiments show that out method can achieve better result compared to cold start and warm start parameters converges better with a few training iterations compared to cold start as well.

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 datasets 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] [22] and Food-101[2] [23] 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 1,000 images for each class. Each class is split into training and testing set containing 750 images and 250 images respectively by its collector. Since this is a big dataset with even class distribution, 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 classes of foods and 31,644 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 contains 731 images while the smallest one contains just 100 images. Since this dataset contains more categories and less images per class, it is used as our target dataset.

[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

B. Challenges of food recognition task

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

- Food is a small sub-category among all 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 inefficient feature representations and degrade the performance.
- Food doesn't have any distinctive spatial layout: for other tasks like scene recognition, we can always find some distinctive features such as buildings or trees, etc. Learning useful representation merely for food recognition itself is a complex task.

These properties requires a model that learn from local information of the images [23]. Generally, recognizing an image consists of two major parts: extracting features and predicting the label with some classifiers. Previous studies show that the first part is more important and training an efficient feature extractor could reduce the complexity of prediction greatly[7] [8]. To achieve high accuracy for similar foods and adapt new foods, we require an efficient feature extractor that can learn both general and distinctive representations for our task. Deep CNN has achieved great progress in recent years. Deep CNNs are proved to learn hierarchical features for image recognition task from some previous work[24][6][25]. 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 results of our fine-tuned deep CNN model on Food-101 datasets.

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. We achieve the-state-of-the-art performance on Food-101 dataset and show that with deep representation and just a few examples from each category, linear model can achieve similar result compared to model trained on full dataset on Food-101. 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[6] in this section as well.

A. Architecture of GoogLeNet

The architecture of deep CNN would greatly affects the performance of the feature extractor and optimized architecture leads to efficient feature representation. 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

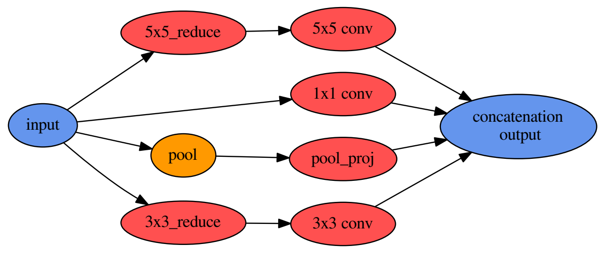

Fig. 1. Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input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

close to the performance of human annotation. We believe that its architecture can help us extract efficient deep feature representations for our adaptation task.

The architecture of GoogLeNet is unique to other deep C-NN. Inspired by [26], 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 details of its architecture can be found from [27].

B. Pre-training and Fine-tuning GoogLeNet

Indeed, training a deep CNN with millions of parameters with insufficient data could easily lead to horrible overfitting. Even though the training set of Food-101 contains 75,750 images, training GoogLeNet on it directly can still be prone to 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 another specific task. Initialization with pre-trained model has certain bias as there is no single dataset including all the invariance for natural images [28], but the bias can be reduced as the size of the pre-trained image dataset increases. So the fine-tuning the model on small dataset should 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 [29]. These two initialization strategies 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. But 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 shallow models (see Table II), GoogLeNet outperforms the other methods with large margins and we provide the state-of-the-art performance on this datasets. Indeed, the deep features representation from fine-tuned GoogLeNet is so efficient that we can achieve similar results by just using a few examples from each category. We evaluate the performance of logistic regression model trained from a few randomly selected examples using deep representation on Food-101 datasets. Table III shows the average performance of the models on different number of examples in each category compared with model trained on full dataset. We can see that with deep representation, model can achieve similar results while using just 5 examples from each category on Food-101 dataset. This indicates GoogLeNet can learn distinctive feature representations on Food-101. In Section IV, the deep feature representations extracted from fine-tuned model are used as the input for our incremental domain adaptation method.

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

C. Discussion of the unique architecture of GoogLeNet

In the last part, we obtain a efficient feature extractor on Food-101 dataset with the architecture of GoogLeNet. Since few work has been found to discuss this architecture, we would like to show some interesting findings from our empirical study.

In order to learn efficient deep representation, From our empirical study, we find that benefitting from the the unique architecture of Inception module, GoogLeNet model can learn the feature representations in a efficient way while models from 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 distinctive compared to AlexNet, and we believe that 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

TABLE II. Top-1 accuracy compared to other methods on Food-101 dataset in percent

	RFDC[?]	MLDS(≈[30])	GoogLeNet	AlexNet
Top1 accuracy	50.76	42.63	**78.11**	66.40

TABLE III. Accuracy for different number of examples in each category

	Number of examples in each category					
	Full (750)	5	4	3	2	1
Top-1	78.11	72.53 ± 0.36	71.07 ± 0.49	69.11 ± 0.84	64.94 ± 0.93	63.62 ± 1.79
Top-5	93.51	90.35 ± 0.26	89.56 ± 0.31	88.41 ± 0.48	86.14 ± 0.52	79.05 ± 1.37

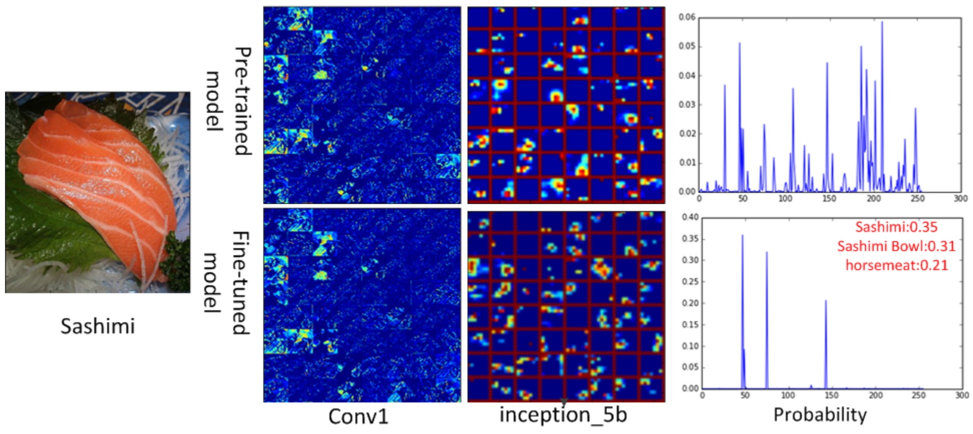

Fig. 2. Visualization of some feature maps of GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

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 V, 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 IV 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 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 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.

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

TABLE IV. 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 V. 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

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

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 distinctive features effectively and by taking advantage of this, this approach achieved some impressive results from previous studies[10] [7]. 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[11]. 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 is not an efficient approach for learning new categories.

Rather than learning efficient representations, another typical approach are more focused on dealing with the representations 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, following the hypothesis that there should be a relatively strong relationship between the categories between two domains[16][17]. Indeed, they work well when transferring the knowledge across domains for two similar or identical categories with shallow feature representation. Shallow representations used in these methods just focuses on local features and can be less distinctive for similar categories. However, for deep representation from CNN, even the distribution of two similar categories could be different and this leads to the failure of their basic hypothesis.

From empirical experiments, we find that adaptive methods suffer as the difference of source and target categories increases. 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 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 updat-

TABLE VII. 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

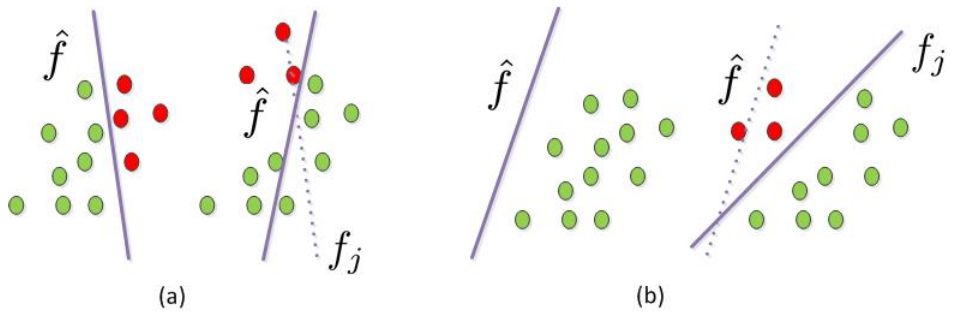

Fig. 3. (a) Conventional adaptation method fail to find good initialization for the new category from source domain. (b) The parameters from $\hat{f}$ are more adaptive for the new categories.

ing the parameters optimized from previous knowledge, their algorithm can search the optimal value for a specific task[21]. 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. The intuition of the warm start parameters is that since the deep representations from new categories are different from our source categories, we can use all examples from the source as negative examples to pre-train a negative classifier. It should be more effective to train the classifier for the new category with the parameters from negative classifier. 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. Instead of learning the whole target categories directly, our method adapts one category in each step and iteratively updates all the parameters. 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: $F$
 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
```

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. The fine-tuned GoogLeNet on Food-101 from Section III is used 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, because previous empirical studies suggest that deeper representations can achieve better performance[11].

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 VII 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. 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 accuracy of target categories on this super-domain (In each step, the super-domain contains all the categories that have learned by our model).

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 dataset, the algorithm starts from a 101 to 102 categories situation. We conduct the following experiment to test the first step of our method: we add an arbitrary new category from Food-256 to Food-101 and evaluate accuracy of the new category in the 102 categories super-domain. We run the first step experiment 220 times in each round, going through all the 220 categories and Table VIII shows the average performance of 10 rounds.

TABLE VIII. 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 VIII 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 in 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 IX. 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. 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] 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.

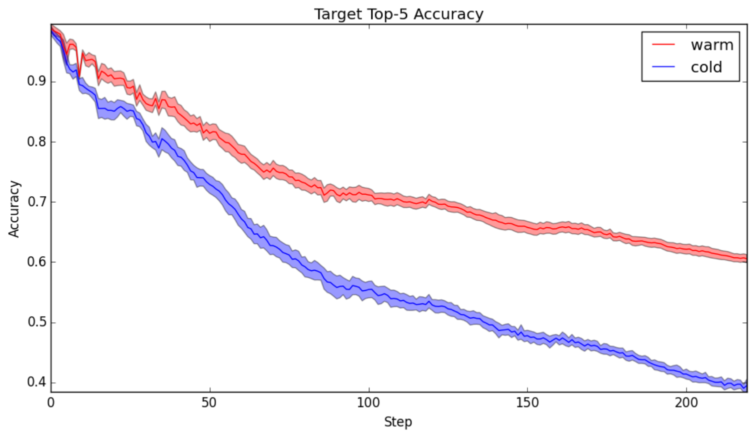

Fig. 4. Top-5 accuracy curve for categories in Food-256 in super-domain with 5 shots. Mean and standard deviation are shown.

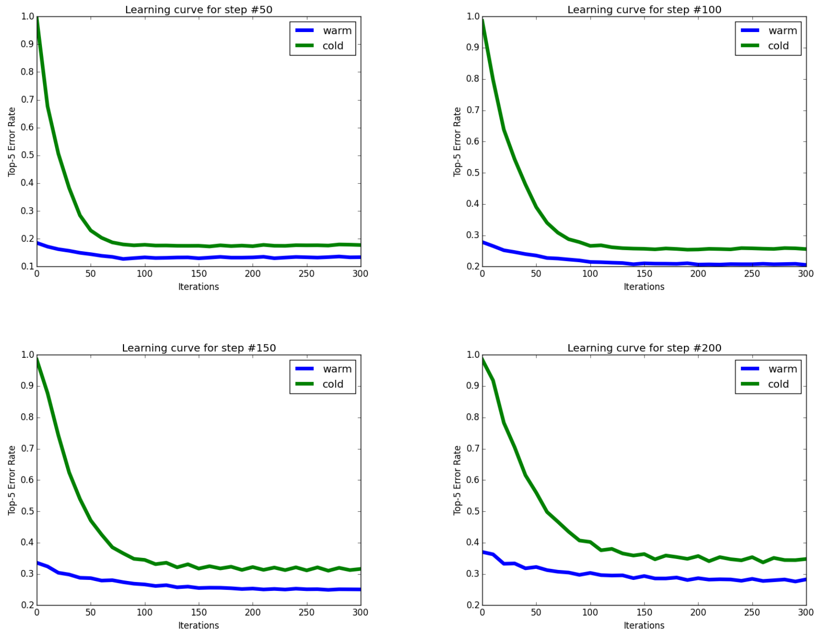

Fig. 5. Overall learning curve in some steps for 300 iterations. **Observation:** even training with enough iteration, warm start can still outperform cold start in a single step.

TABLE IX. OVERALL 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	cold start	warm start	cold 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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 834–849.

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

[16] 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.

[17] 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.

[18] E. A. Yildirim and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 782–810, 2002.

[19] E. John and E. A. Yıldırım, "Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension," *Computational Optimization and Applications*, vol. 41, no. 2, pp. 151–183, 2008.

[20] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," *Automatic Control, IEEE Transactions on*, vol. 56, no. 7, pp. 1524–1534, 2011.

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

[22] 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.

[23] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 446–461.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.