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 (target) categories from a different domain is always an interesting and challenging task in data mining. The classifier could suffer from dataset bias when predicting the new categories from target domain. Many adaptation methods have been proposed to adjust this bias but are limited to either similar categories or requiring a large number of labeled examples from the target domain. Automatically adapting and recognizing new food categories is a very practical task in daily life. In this paper, we propose a new method that can alleviate the dataset bias for food image recognition. To obtain less biased feature representation from the food images, we fine-tuned GoogLeNet as our deep feature extractor and achieve state-of-the-art performance on Food-101 dataset. Using the deep representation, our method can learn efficient classifiers with fewer labeled examples. More specifically, our method employs an external classifier for adaptation, called "negative classifier". Experiment results show that utilizing the parameters of the negative classifier, our method can achieve better performance and converges faster to adapt new categories.**

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 testing error is positively proportional to the difference between the test and training input distributions [1] [2]. Therefore, mismatched data distribution can be the major problem for predicting the data from a different domain. Domain adaptation is proposed to solve this problem, transferring the knowledge from source domain to target one. For image recognition tasks, model trained from the source domain is inherently biased as there is no database that includes all the transformations of the objects.

Automatically recognizing the food images is a practical technique in daily life, from which people would benefit in various ways, such as healthy eating and diet control. The properties of the food image makes it a very challenging task for recognition (see Section II). Recently, Deep 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 [7] [10] [11] [12]. However, fine-tuning deep CNNs 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 for fine-tuning deep CNNs on a small dataset, there is no limitation for training linear shallow models with deep representation from the 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, we find that fine-grained deep representation can also help the shallow classifier learn categories with fewer examples. So by applying appropriate adaptation strategy, the shallow models can still adapt new categories with fewer examples.

From previous studies, many shallow domain adaptation methods for the image recognition task have been proposed to compensate the dataset bias, assuming that the parameters of the linear classifier for similar categories have certain linear correlation [15] [16] [17]. However, these methods require the target category to be geometrically similar to the source one [17]. They use shallow feature representation such as HOG or SIFT etc, as the inputs of the linear model. So the inputs of the two categories 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. We find that even for two similar categories, the parameters of their classifiers is not necessarily to be similar for deep representation (see Section III).

Instead of finding linear correlated parameters for the new category from learned categories, we employ a negative classifier whose parameters are "far away" from any previous learned model parameters. Technically, this negative classifier is trained to classify all examples from learned categories as negative examples (see Figure 1). Experiment results show that initialize the new classifier with parameters of this negative classifier (warm start) 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. Previous studies suggest that warm started parameters can achieve better accuracy and fast convergency [18] [19] [20] [21].

In this paper, we aim the adaptation problem on a specific area, food recognition. We first obtained a model that can learn 101 kinds of food from deep representation and try to deal with the problem: adapting 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 food images, we first train the efficient feature extractor by fine-tuning GoogLeNet and achieve the state-of-the-art performance on our source domain, Food-101 dataset. With deep representation, compared to the results training on full dataset, we can achieve similar results while just using a few examples in each category.
2) Negative classifier for warm start. Warm start new classifier with negative classifier can take advantage of deep representation. Benefitting from warm start parameters in each step, experimental results show that our method can achieve better result. Also our warm started parameters can converge with fewer training iterations.

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-220 and Food-101[1] [22] 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-220 Dataset. This dataset is a subset of a large food dataset containing 256 foods[2] [23]. We removed 36 categories already contained in Food-101 dataset, from the original dataset as the target domain. This is a relatively small dataset containing 220 classes of foods and 26,278 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://www.vision.ee.ethz.ch/datasets_extra/food-101

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

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 method that can learn from local information of the images [22]. Generally, recognizing an image consists of two major parts: extracting feature representation from the image and predicting the label with the extracted representation. Previous studies show that the first part is more important and using efficient feature representaion could reduce the complexity of prediction greatly [7] [8]. To achieve high accuracy for similar foods and adapt new foods, we need an efficient feature extractor that can learn both general and fine-grained features from food images. 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 [6] [24] [25]. Compared to other representation from shallow model, representation extracted from deep CNN can be better distinguished by linear model. Therefore, we train a deep CNN as our feature extractor in this paper and use the deep feature representations as the input of our adaptive method. In Section III, we show the results of our fine-tuned deep CNN model on Food-101 datasets.

III. Training Deep Feature Extractor

In this section, we introduce the architecture used for deep CNN. 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 food category in Food-101, linear model can achieve competitive result compared to model trained on full dataset on Food-101. We also discuss some features we find for GoogLeNet in comparison with AlexNet [6] in this section to see the advantage of fine tuning GoogLeNet. Both architectures in this paper are their Caffe implementations [26].

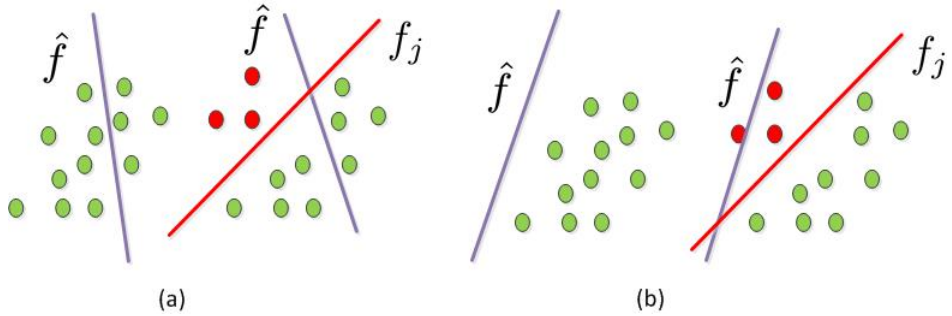

Fig. 1. The classifier for class j is initialized with $\hat{f}$ and converged at f_j. (a) Conventional adaptation method using the parameters of the learned category and need more training time find good solution for the target category. (b) The parameters from $\hat{f}$ reject all learned categories and can learn faster for the new categories.

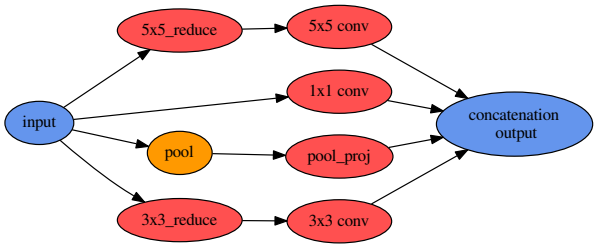

Fig. 2. Inception Module. $n \times n$ stands for size n receptive field, $n \times n\_reduce$ stands for the 1×1 convolutional layer before the $n \times n$ convolution layer and $pool\_proj$ is another 1×1 convolutional layer after the MAX pooling layer. The output layer concatenates all its input layers.

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 close to the performance of human annotation [27]. We finds that GoogLeNet can extract better feature representation compared to other method.

The architecture of GoogLeNet is unique to other deep CNN. Inspired by [28], small 1×1 receptive field are intensively used throughout the network. There are 9 "Inception" modules in GoogLeNet and Figure 2 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, even training a deep CNN directly with over a million labeled examples should be very careful to prevent overfitting [6]. Because the training set of Food-101 contains only 75,750 images, training GoogLeNet on it directly can be easily 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 , but the bias can be reduced as the size of the pre-trained image dataset increases [29]. So the fine-tuning the model on small dataset should benefit from it.

To compare the result of GoogLeNet with other deep CNN model, we use the AlexNet as the baseline of deep CNN. We conduct several experiments on both architectures and use different training initialization strategies for Food-101 datasets. In our experiment, the scratch models are initialized with Gaussian distribution for AlexNet and Xavier algorithm for GoogLeNet [30]. 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.

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 and we achieved the state-of-the-art performance on Food-101 dataset. Compared to other shallow models (see Table II), GoogLeNet outperforms the other methods with large margins.

In [9], Deep CNN is proved to learn hierarchical hierarchical feature representations automatically. In Figure 3 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 (such as features for lines and edges detection etc.) are similar for most recognition tasks. Then we can see that the difference of the feature maps in the high-level leads to total different recognition results. Since only the last layer (auxiliary classifier) of the ft-last

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

	RFDC[22]	MLDS(≈[31])	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

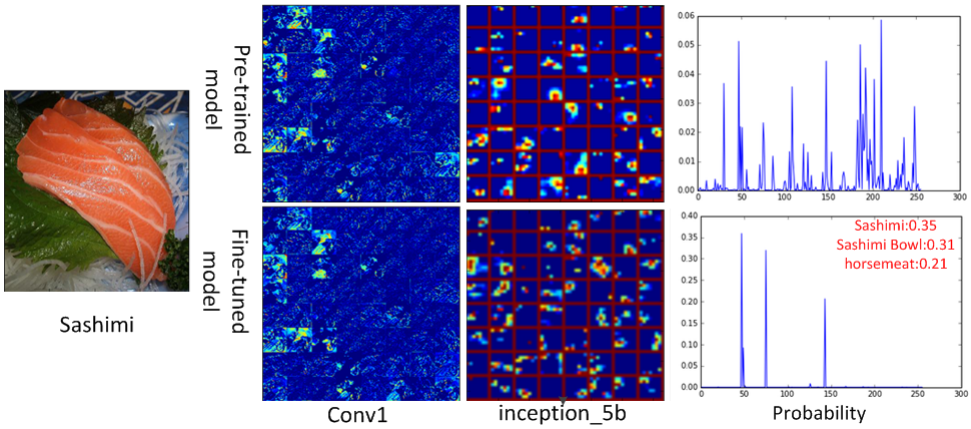

Fig. 3. Visualization of some feature maps of GoogLeNet models in different layers for the same input image. 64 feature maps of each layer are shown. Conv1 is the first convolutional layer and Inception_5b is the last convolutional layer.

model is optimized, we can infer that fine-tuning the network can learn better deep representation.

From previous work, deep feature representation from CNN has shown some impressive results for fine-grained recognition, which suggests that deep feature representation can learn distinctive features from similar categories [14] [32]. We shows the weight similarity of the linear auxiliary classifier and average representation similarity in GoogLeNet for some similar foods in Food-101 dataset in Table IV and V. We can observe that the different representation eventually leads to different weights of the linear classifier. Moreover, we find that with deep representation, we can achieve competitive accuracy while just using a few labeled examples from each category. We use deep representation from GoogLeNet as the input of logistic regression classifier and compare the results of different number of training examples on Food-101 dataset. Table III shows the average accuracy of the models on different number of examples in each category compared with model trained on full dataset. We can see that even though size of the training examples is significantly reduced, the performance of the classifier is not greatly degraded. This suggests that GoogLeNet can learn fine-grained feature representations on Food-101 which could be efficient way to alleviate the dataset bias for adaptation. In Section IV, we use the deep feature representation extracted from fine-tuned GoogLeNet as the input for our incremental domain adaptation method.

C. Advantage of GoogLeNet for fine-tuning

In the last part, we obtain an efficient feature extractor on Food-101 dataset with the architecture of GoogLeNet. In this part, we introduce some of our findings that makes GoogLeNet efficient.

TABLE IV. COSINE SIMILARITY OF THE WEIGHTS FOR SIMILAR FOODS

Caesar salad	Caprese salad 0.1406	Greek salad 0.3155
Cheesecake	Carrot cake 0.1711	Cup cakes 0.1276
Club sandwich	Grilled cheese sandwich 0.2866	Hamburger 0.2303

TABLE V. AVERAGE COSINE SIMILARITY OF THE REPRESENTATION FOR SIMILAR FOODS

Caesar salad	Caprese salad 0.4005	Greek salad 0.5612
Cheesecake	Carrot cake 0.4952	Cup cakes 0.4561
Club sandwich	Grilled cheese sandwich 0.6651	Hamburger 0.6038

Training a neural network is essentially adjusting the parameters of the network to minimize the empirical risk. Different from any previous architectures [6] [8], the GoogLeNet consists of several stacked Inception modules. We find that GoogLeNet model can learn the kernel weights of each layer more efficiently while the model with other architecture (we take AlexNet for contrast in this paper) would suffer from vanished gradient a lot.

From Table I we can see that GoogLeNet outperforms AlexNet which implies that the deep feature representations of GoogLeNet are more distinctive compared to AlexNet. To

illustrate the changes of the weights for different architectures after fine-tuning, we calculate the weights' cosine similarity of each layer between the fine-tuned models and their pre-trained models and show them in table VI and VII. Because the pre-trained models are used for general recognition task, for our specific food recognition task, we expect a model that can learn features only for food recognition. Therefore, the weights of the model should be greatly adjusted to better fit our new scenario.

From Table VII, we can observe that the weights of the pre-trained and fine-tuned models are extremely similar in AlexNet which we believe is caused by vanished gradient. Vanished gradient is a difficulty for deep neural network while errors propagate shrink greatly as it goes deeper [30]. This makes the network difficult to fit the data and degrades its performance. Even though ReLUs is proposed to alleviate this problem in [33] and used in both architectures, from Table VI and VII we can see that they still suffer from this problem. However, GoogLeNet suffers significantly less. This indicates that GoogLeNet could be more flexible to fit the new examples which is consistent with the results in Table I.

By comparing the changes of weights in fine-tuning, we can see that training GoogLeNet is more easy and achieve better result.

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 introduce our adaptive method for learning new food categories using the feature representations from GoogLeNet. We conduct several experiments on our adaptation task to illustrate the advantages of our method. For the rest part of this section, we first discuss the limitation of some previous domain adaptation approaches for our task in details. Then we introduce our method and show the experiment results on our task.

A. Limitation 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 and achieve an accuracy of 78.11%. 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 sampling noisy is a major problem that makes fine-tuning deep CNN on limited examples prone to overfitting [13]. Apart from these reasons, fine-tuning the whole network requires intensive computational resources which is not an efficient approach for learning new categories.

Rather than learning efficient representations, other approaches are more focused on dealing with the representations learned from conventional feature extraction methods for domain adaptation [16] [17]. Supervised domain adaptation models, such as Adaptive-SVM (A-SVM) and PMT-SVM[3], try to utilize the knowledge from source domain and apply to target domain, assuming that the classifier of the new category should have some linear relationship with the classifiers of the source categories. Indeed, they work well when the categories are geometrically similar. However, for deep representation from deep CNN, this assumption doesn't help much.

From empirical experiments, we find that adaptive methods suffer as the difference of source and target categories increases. Table VIII shows some experiment results for two typical domain adaptation methods (A-SVM & PMT-SVM) and two classifiers without any adaptation in a binary classification scenario. For adaptation methods, we manually choose the similar/identical source categories for the target ones. 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 [17]. The parameters used in these methods are set as their default. From the results of our simple experiment, we can see that using deep representation, even to adapt identical category, the improvement of the adaptation technique is not significant compared to raw models.

For our food recognition task, we expect to find a method that can learn new food with just a few labeled images. From Section III, we show that shallow linear model trained on small dataset can achieve competitive result with deep representation. This suggests that deep representation can adjust the dataset bias and help for adaptation. To adapt new category with deep representation, we propose a incremental adaptive learning method.

B. Incremental adaptive learning for new category

From previous section we can see that linear assumption doesn't help classifier to adapt new category with deep representation significantly. In Table IV we show the weight similarity of the classifiers for different categories and observe that even for similar categories, the parameters of the classifiers are quite different. So we think that if we warm start the classifier of the new category with some parameters that are different from any learned ones for optimization, the classifier can be trained more efficiently. Warm start is to initialize the model with some chosen parameters and has been used in many linear models. Chu et al. recently proposed a warm start approach for parameter search in linear model and by iteratively updating 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.

To choose the parameters that are different from previous learned categories, we introduce a negative classifier which is trained using all examples from the learned categories as

[3]We use the code in [17] downloaded from http://www.robots.ox.ac.uk/ vgg/software/tabularasa/ for these two adaptive methods

TABLE VI. 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 VII. 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 VIII. AVERAGE PRECISION FOR A-SVM, PMT-SVM, SVM AND LOGISTIC REGRESSION TRAINED WITH DEEP REPRESENTATION. SOURCE CATEGORIES WITHOUT * ARE DETERMINED BY COSINE SIMILARITY

Target Category	Source Category	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

negative examples. This negative classifier tends to reject all the learned categories and using its parameters, different from any previous learned parameters, to initialize the new classifier should be more effective for optimization. We uses logistic regression as the linear classifier because from Table III we can see that logistic regression can achieve competitive result with a few examples.

To learn multiple new categories, instead of learning the them once, our method adapts one category in each step and updates all the parameters for the learned classifiers as well as the negative one. The algorithm terminates while all the new categories are leaned. For each step, given M learned categories in set S and a new category t from target domain, there are M binary classifiers $F = \{f(w_i, b_i)\}_{i=1}^{M}$ for each learned category. We use $\mathcal{P}^-{}_i$ and $\mathcal{P}^+{}_i$ to denote the negative examples and positive examples for the ith class respectively. The negative classifier $\hat{f}(\hat{w}, \hat{b})$ is trained with $\mathcal{P}^-{}_{neg} = S$. The classifier f_t for the new category t is initialized with $\left\{\hat{w}, \hat{b}\right\}$ and trained with $\mathcal{P}^-{}_t \bigcup \mathcal{P}^+{}_t$ for $\mathcal{P}^+{}_t = t$ and $\mathcal{P}^-{}_t = S$. Then we update the negative classifier $\hat{f}$ with negative examples $\mathcal{P}^-{}_{neg} = S \bigcup t$. We use Stochastic Gradient Descent (SGD) to update the parameters for all the binary classifiers as well as the negative classifier. The complete strategy for our method is given in Algorithm 1.

Algorithm 1 Warm start adaptation with negative classifier

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

C. Experiment setup & Evaluation

We design the following experiment, transferring from Food-101 dataset to Food-220 dataset while just using a few examples to see the performance of our method. 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. Previous study suggests that deeper representations can achieve better performance [11], so deep feature representations from *Pool5*, the layer before auxiliary linear classifier on top, are used as the inputs of our method.

Baseline: To illustrate the advantage of our warm start strategy, we use the same model with random initialization (cold start) as the baseline. Also, compared supervised adaptation techniques, such as A-SVM, from Table VIII we can see that previous adaptation methods are not better than those model without adaptation.

5 shorts learning: In our experiments, to build a balanced class distribution, the training set contains 5 randomly picked

training examples from each category from both food datasets. To test the performance of our algorithm for new categories, the test set only contains all the rest examples in Food-220. For each step, we evaluate the accuracy of all adapted categories (learned categories in Food-220) among all learned categories.

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 [26]. We run SGD for 300 iterations for both warm start and cold start ensuring that they both converge.

D. Adapting single new 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. The experiment starts with 505 training examples from Food-101 and in each step, we add 5 randomly picked examples from one category in Food-220 and train the new classifier as well as updating the other ones. So to adapt all the categories in Food-220 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-220 to Food-101 and evaluate accuracy of the new category in the 102 learned categories. We run this experiment 220 times in each round, going through all the 220 categories. Table IX shows the average performance of 10 rounds.

TABLE IX. SOME RESULTS OF TOP-5 ACCURACY FOR ADAPTING SINGLE NEW CATEGORY.

new category	Warm	Cold
crape	**84.16**	62.29
chip butty	**80.97**	65.82
meat loaf	**67.78**	53.15
dried fish	**92.00**	79.81
scrambled egg	**75.21**	63.54
pork belly	**81.76**	70.59
Overall average	**91.91**	88.82

From Table IX we can see that by taking advantage of the warm started parameters from negative classifier, the warm start classifiers does a slightly better than cold start ones in this experiment. Even though the difference of these two method is small after adapting just one category, we believe the margin of these two methods would increase as more new categories are learned.

E. Adapting multiple new categories

In this part, we show the performance of warm start in adapting multiple new categories situation. From the experiments above, warm start shows some improvement on cold start and in this part, we show that the improvement can be accumulated as more categories are learned and our method can outperform cold start with larger margin. In order to get consistent results, we have to add the new categories according to certain sequence in each step. This sequence is determined by its original class index in Food-220. We run this experiment 10 times to eliminate the randomness caused by the training set and show the average results for both warm and cold start in Figure 4 .

From Figure 4 we can see that benefitting from warm start, classifiers can benefit from the "good" parameters of the previous step and thus the margin between warm and cold starts increases as more new categories are learned. Moreover, we find that in each step warm started classifier converges better than cold start. We select some steps and show their learning curve in Figure 5. From Figure 5 we can see that by taking advantage of the parameters from negative classifier, warm started classifier converges much faster than cold start. We also compare performance of our method using different training iteration and show the results after adapting all new categories in Table X. From Table X we can see that even training with 10 iterations, warm started classifier can still reach competitive results. All these evidences indicate that warm start the parameters from the negative classifier can help the model to achieve better results with fewer training iteration.

TABLE X. OVERALL TOP-5 ACCURACY FOR ADAPTED CATEGORIES IN LEARNED CATEGORIES WITH DIFFERENT TRAINING ITERATION IN EACH STEP.

	10 iterations	20 iterations	50 Iterations	300 iterations
Top-5	60.37± 0.58	60.54±0.32	60.33±0.62	**61.48±0.59**

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. we achieves the state-of-the-art performance on Food-101 dataset. We also show that deep representation can alleviate the dataset bias and can be used for adaptation efficiently. We propose a method that can adapt new food categories with deep representation. By learning one new category and warm start with the parameters from negative classifier in each step, our method shows improved result compared to method without warm start.

Since 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 data bias.

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.

[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.

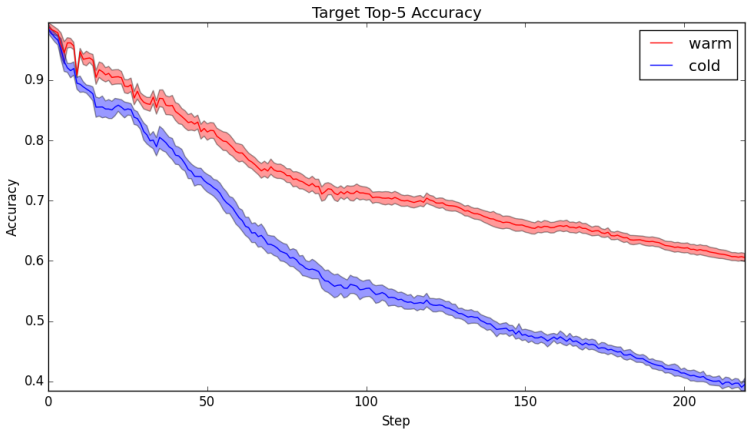

Fig. 4. Top-5 accuracy curve for categories in Food-220 in super-domain with 5 shots. Mean and standard deviation are shown.

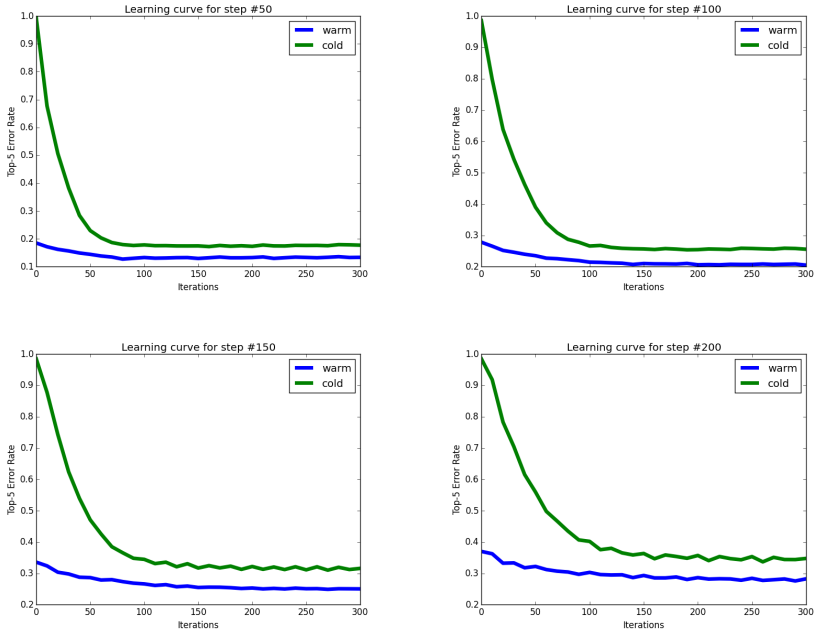

Fig. 5. Overall learning curve in some steps for 300 iterations. **Observation:** even training with enough iteration, warm start can still outperform cold start in a single step.

[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] 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.

[23] 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.

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

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

[29] 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.

[30] 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.

[31] 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.

[32] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 512–519.

[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010, pp. 807–814.