

Cost-Effective Training of Deep CNNs with Active Model Adaptation

Sheng-Jun Huang
College of Computer Science and
Technology,
Nanjing University of Aeronautics
and Astronautics
Nanjing, China
huangsj@nuaa.edu.cn

Jia-Wei Zhao
College of Computer Science and
Technology,
Nanjing University of Aeronautics
and Astronautics
Nanjing, China
jiaweizhao@nuaa.edu.cn

Zhao-Yang Liu
College of Computer Science and
Technology,
Nanjing University of Aeronautics
and Astronautics
Nanjing, China
zhaoyangliu@nuaa.edu.cn

ABSTRACT

Deep convolutional neural networks have achieved great success in various applications. However, training an effective DNN model for a specific task is rather challenging because it requires a prior knowledge or experience to design the network architecture, repeated trial-and-error process to tune the parameters, and a large set of labeled data to train the model. In this paper, we propose to overcome these challenges by actively adapting a pre-trained model to a new task with less labeled examples. Specifically, the pre-trained model is iteratively fine tuned based on the most useful examples. The examples are actively selected based on a novel criterion, which jointly estimates the potential contribution of an instance on optimizing the feature representation as well as improving the classification model for the target task. On one hand, the pre-trained model brings plentiful information from its original task, avoiding redesign of the network architecture or training from scratch; and on the other hand, the labeling cost can be significantly reduced by active label querying. Experiments on multiple datasets and different pre-trained models demonstrate that the proposed approach can achieve cost-effective training of DNNs.

1 INTRODUCTION

Deep neural networks have shown to be very effective in many fields. In addition to the impressive performance, it is also well known that training an effective deep model for a new task from scratch could be rather challenging [12, 22, 34]. Firstly, a prior knowledge or rich experience is needed to design a suitable network architecture for a new task. Secondly, one should repeat the trial-and-error process to tune the hyper-parameters, which may significantly affect the learning performance. Lastly, a huge amount of labeled data is required for model training, leading to high annotation cost. These problems commonly occur, and cannot be overcome in most real applications, strongly limiting the application of deep learning to more tasks. It is thus important to have some effective strategies to train deep neural networks with lower cost on model designing, parameter optimization and data labeling.

To reduce the cost of designing network architecture and optimizing the model parameters, one straightforward idea is to exploit some pre-trained models instead of building the model from scratch. Transfer learning is an important method to ease the training of the target model by exploiting information from a different but related source domain [25]. Under the deep neuron network framework, knowledge transfer from source domain to target domain has been

implemented via distribution matching [32], feature sharing [21] or components transformation [24]. However, these methods typically revise the architecture of the pre-trained networks, making the retraining still challenging with too many parameters. A better choice is to perform model adaptation with fixed architecture of pre-trained models. For example, in computer vision field, we are lucky to have a large dataset ImageNet [6] with all images manually labeled. Some well-known models such as AlexNet [19], VGG [30] and ResNet [15] have been pre-trained on ImageNet with descent performance. These models are good at learning effective representations for visual objects, and are public available for us to utilize. The simplest way to utilize the pre-trained model is to directly employ the entire network except for the output layer as a feature extractor. However, such a simple strategy is less practical because the feature extracted will be less effective when the target task is not very similar to the original one [39]. One alternative approach is to retain the architecture of the pre-trained model, and then retrain the model via fine-tuning its weights. Such methods can partially reduce the training cost, but still require a relatively large dataset to optimize the network weights [3].

Active learning is a primary approach for reducing the labeling cost [28]. It iteratively queries the labels for the most useful instances, and tries to train an effective model with less queries. Various criteria have been proposed for active selection in traditional shallow models [4, 10, 20]. For example, informativeness and representativeness have been demonstrated to be good choice for estimating the potential contribution of an instance on improving the classification [36]. However, the active learning strategies designed for traditional shallow models have been validated to be not effective for deep models, because of the conflict between the batch sampling redundancy and local optimization [27]. There are several studies focus on active learning for deep neural networks [27, 34, 38]. However, they usually do not consider the pre-trained model, and thus may lead to waste of annotation cost by querying information already contained in pre-trained models.

In this paper, we propose to perform active model adaptation for cost-effective training of deep convolutional neural networks. On one hand, the training cost is reduced by fine tuning the pre-trained models with frozen layers; and on the other hand, the labeling cost is reduced by actively querying the labels for the most useful instances. Specifically, a novel criterion for active selection is proposed to simultaneously consider and dynamically adjust the

distinctiveness (peculiarity endemism) and uncertainty of an instance. Thus the selected instances are expected to be most useful for both the classifier training and representation learning.

We perform experiments on multiple datasets and different pre-trained models. The results demonstrate that the proposed approach can effectively train a deep convolutional neural network model with significantly lower cost both on the training process and the label annotation. Specifically, our algorithm can achieve comparable performance by using only 5% of the training data compared to passive training from scratch.

The main contributions of this work are summarized as follows.

- A general framework of active model adaptation for deep convolutional neural networks. It can be applied to different pre-trained models and incorporated with various active selection strategies.
- A novel criterion *distinctiveness* is proposed to measure the potential contribution of an instance on improving the feature representation of the network.
- An algorithm can actively select instances based on dynamic trade-off between *distinctiveness* and *uncertainty*, and can also improve the deep model to achieve better feature representation and label prediction.
- Extensive experiments validate the effectiveness of the proposed approach.

The rest of this paper is organized as follows. In Section 2, we review the previous work. In Section 3, the propose approach is introduced. Section 4 presents the experiments, followed by the conclusion in Section 5.

2 RELATED WORK

Transfer learning tries to improve the performance in a target domain by transferring information from a related but different source domain [25, 37]. It has been extensively studied by transferring knowledge from source domain to target domain at instance, feature or model levels [7, 14, 25, 31, 40]. Among which, model level transferring is more related to our work [2, 24]. Generally speaking, these methods try to train a new model for the target domain by directly reusing or modifying the model that has been well trained in the source domain.

Recently, there are increasing interests on transfer learning for deep neural networks [13, 22]. For example, authors of [32] propose a new network structure which adds an adaptation layer to minimize both domain confusion loss and classification loss simultaneously. In [21], domain adaptation is considered in all task-specific layers for matching different domain distributions effectively. Inspired by WGAN[1], in [29], the domain discrepancy based on Wasserstein distance is minimized in adversarial learning process to optimize the representation.

These methods can partially ease the training of deep models, but they do not consider active learning during the transfer process, and thus still need a relatively large set of labeled examples to train an effective model for the target domain. Moreover, they usually need to revise the architecture of the network, and thus cannot overcome the challenge on expensive network design and optimization.

Active learning reduces the labeled training examples by selecting the most valuable instances to query their labels [28]. During

the past decades, many criteria have been proposed for active selection of instances [10, 28, 33, 41]. Recently, there are some studies applying active learning strategies into deep learning to reduce the training data [12, 27, 34, 38]. In [12], a tractable approximation method is proposed to estimate the predict variance of a Bayesian CNNs [11], based on which, the examples with high variance are selected for label querying. The authors of [27] transform active learning into a core-set selection problem under the deep learning setting, and try to select instances to make the model trained on the selected subset competitive for other examples. The method proposed in [34] tries to query labels for uncertain instances, while at the same time assigns pseudo labels for the examples with high prediction confidence. In [38], the similarity between examples at the last layer of a fully convolutional network combined is combined with uncertainty for active selection.

There are some studies incorporating active queries for transfer learning with traditional shallow models. For example, the method in [35] combines active learning and transfer learning into a Gaussian Process based approach, and sequentially selects query points from the target domain based on the predictive covariance. Kale and Liu [16] propose a principled framework to combine the agnostic active learning with transfer learning, and utilize labeled data from source domain to improve the performance in the target domain. Kale et al. [17] present a hierarchical framework to exploit cluster structure between different domains, and try to impute labels for unlabeled data and select active queries in the target domain based on the structure. These methods are generally designed for traditional shallow models, and neglect the special challenges of training deep neural networks.

There is one study trying to actively fine tune a pre-trained deep neural network for biomedical image analysis [42]. The authors propose a new criterion to estimate the diversity among different patches extracted from the same image, and expect the image with more diverse patches is more useful for updating the model. However, this method is specially designed for biomedical image analysis, and can only handle binary classification problems, which strongly restricts its application.

3 THE PROPOSED APPROACH

We denote by $U = \{\mathbf{x}_i\}_{i=1}^{n_u}$ the unlabeled dataset with n_u instances, where \mathbf{x}_i is the i -th instance. In this paper, we perform batch-mode active learning. At each iteration, a small batch of instances $Q = \{\mathbf{x}_q\}_{q=1}^b$ with size b will be selected from U to query their labels. We also denote by \mathcal{M}^0 the pre-trained model, and \mathcal{M}^t the model at the t -th iteration. In the following subsections, we will first introduce the batch-mode active learning framework for deep CNN model adaptation, then propose the criterion for selecting instances, and at last summarize the main steps of the algorithm.

3.1 The framework

We focus on batch-mode active learning, i.e., query labels for a small batch of instances selected from the unlabeled set. Figure 1 presents the framework of active model adaptation for deep convolutional neural networks. Here we take AlexNet [19] as an example, which is a well-known model pre-trained on ImageNet ILSVRC dataset.

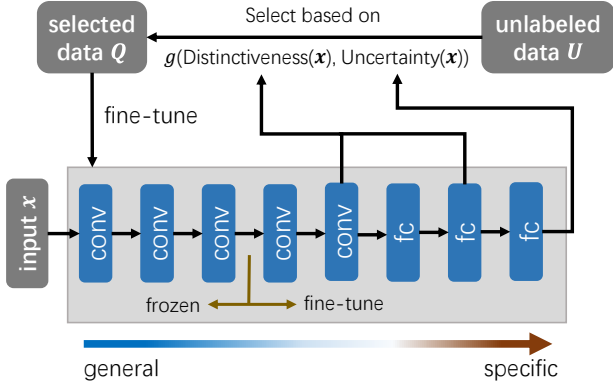


Figure 1: The framework of active model adaptation for deep neural networks.

Typically, the deep neural networks learn an effective representation from general to specific. The first few layers mainly capture universal features such as curves and edges, and then the later layers generate more task-specific features. The general features work for different tasks, while the specific features focus on describe the unique properties of a task. When adapting a pre-trained model from the source task to the target task, it is reasonable to retain the layers for general features, while update the specific layers to fit the requirement of the target task.

For each instance in the unlabeled set, the original features are input into the current neural network. Then based on the representations in different layers and the final prediction outputs, distinctiveness and uncertainty of the input instance are estimated respectively. The distinctiveness measures the ability of an instance on capturing the particular property of the target task that differs from the source task; while the uncertainty measures the ability of an instance on improving the classification model. In other words, distinctive instances are responsible to improve the layers of learning the representation; while uncertain instances are responsible to improve the last layer of classification. The two criteria are jointly considered to select a small batch of most informative instances. After querying their labels, the selected instances are utilized to fine tune the neural network, where the early layers are frozen to retain the information from source task.

3.2 The active selection criterion

In this subsection, we will introduce the two criteria *distinctiveness* and *uncertainty* respectively. In a deep convolutional neural network, the early layers produce general features, while later layers generate more specific features, and the last layer usually corresponds to a classifier. To adapt a deep network from a source task to a target task with fixed architecture, the key problem is to update the network weights. When actively selecting the instances to help the model adaptation, the model after adaptation is expected to on one hand, the adapted model is capable of learning an effective representation for the target task; and on the other hand, the classifier

can achieve high accuracy. We thus propose two criteria to estimate the usefulness of an instance on these two aspects, which are named *distinctiveness* and *uncertainty* respectively. The workflow of calculating these two criteria is summarized in Figure 2

3.2.1 Distinctiveness. We propose a novel criterion *distinctiveness* to measure the ability of an instance on improving the representation quality of the neural network for the target task. The pre-trained model is optimized for the source task. If we want to optimize it to fit the target task, then some instances which can capture the unique property of the target task should be used to fine tune the model. We call such capability as distinctiveness because it distinguishes the target task from the source target. To estimate the distinctiveness of an instance, the basic idea is to firstly exploit the pattern of the pre-trained model on feature transformation from early to later layers. If an instance in the target task has a transformation pattern that is significantly different from that of the source task, then this instance is expected to be more distinctive. Following we introduce the detailed steps of calculating the distinctiveness of an instance.

First of all, assume that there are in all K classes in the source task, then one representative instance is selected for each class, leading to the representative instance set $C = \{c_1, c_2, \dots, c_K\}$. We denote by z the final representation of an instance in the source task. Then for each class k , the mean of all instances belong this class can be calculated as

$$\bar{z}_k = \frac{1}{|\Omega_k|} \sum_{z \in \Omega_k} z, \quad (1)$$

where Ω_k is the set of instances belong to the k -th class, and $|\cdot|$ calculates the set size. Then c_k is selected as

$$c_k = \operatorname{argmin}_{z \in \Omega_k} \|z - \bar{z}_k\|^2. \quad (2)$$

The feature transformation pattern on these representative centers is used to describe how the network learns the features from early layers to later layers in the source task. For convenience of presentation, we specify one earlier layer as A , and a latter layer as B , and then denote by $S_c^{A \rightarrow B}$ the transformation pattern of the center c . These transform patterns can be considered to be optimal for the source task, but less optimal for the target task. Given an instance x in the target domain, a similar feature transform pattern $S_x^{A \rightarrow B}$ can be obtained. In addition, we can further get an approximated pattern $\hat{S}_x^{A \rightarrow B}$ by weighted combination of the representative source patterns. In fact, $\hat{S}_x^{A \rightarrow B}$ is estimating how the network will transform the features of x if it is an instance from source task, while $S_x^{A \rightarrow B}$ is the observed pattern of x . So the difference between these two patterns reflects how distinctive x is from the source task. Follows we discuss how to calculate feature transform pattern from layer A to layer B .

Given a source center c_k , its outputs at layer A and layer B are denoted by c_k^A and c_k^B , respectively. Similarly, we have x^A and x^B

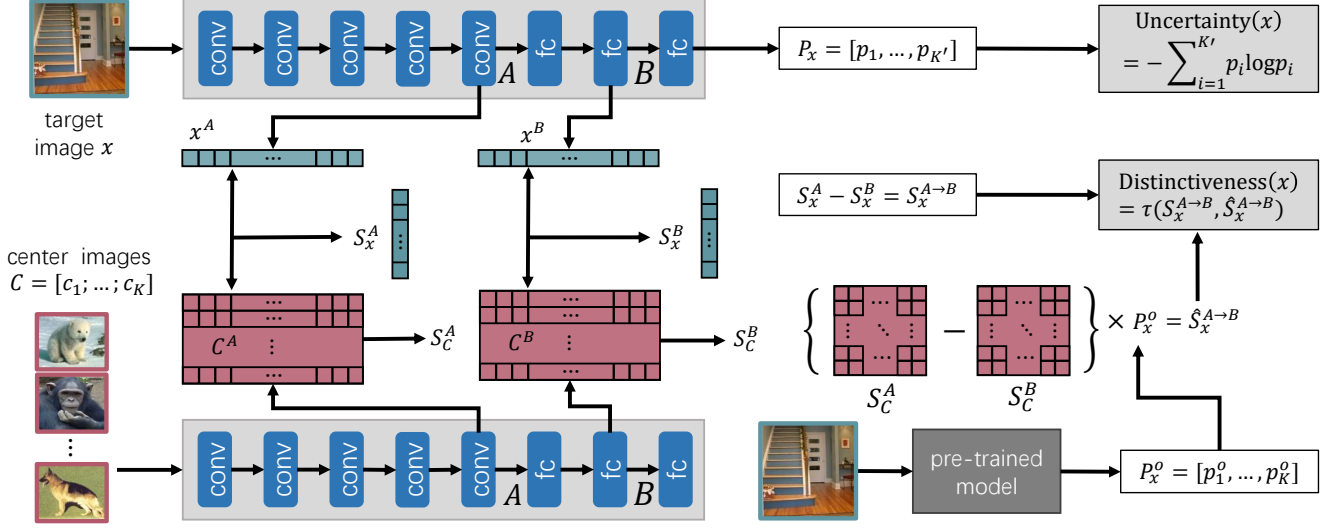


Figure 2: The criterion for active selection.

for an target instance \mathbf{x} . Then we define:

$$S_C^A = [S_{c_1}^A, S_{c_2}^A \dots, S_{c_K}^A]$$

$$= \begin{bmatrix} \|\mathbf{c}_1 - \mathbf{c}_1\|^2, & \|\mathbf{c}_2 - \mathbf{c}_1\|^2 & \dots & \|\mathbf{c}_K - \mathbf{c}_1\|^2 \\ \|\mathbf{c}_1 - \mathbf{c}_2\|^2, & \|\mathbf{c}_2 - \mathbf{c}_2\|^2 & \dots & \|\mathbf{c}_K - \mathbf{c}_2\|^2 \\ \dots & \dots & \dots & \dots \\ \|\mathbf{c}_1 - \mathbf{c}_K\|^2, & \|\mathbf{c}_2 - \mathbf{c}_K\|^2 & \dots & \|\mathbf{c}_K - \mathbf{c}_K\|^2 \end{bmatrix}, \quad (3)$$

and

$$S_{\mathbf{x}}^A = \begin{bmatrix} \|\mathbf{x} - \mathbf{c}_1\|^2 \\ \|\mathbf{x} - \mathbf{c}_2\|^2 \\ \dots \\ \|\mathbf{x} - \mathbf{c}_K\|^2 \end{bmatrix}. \quad (4)$$

Similarly, we have S_C^B and $S_{\mathbf{x}}^B$ corresponding to the layer B . In fact, the centers $\{\mathbf{c}_k\}_{k=1}^K$ are taken as landmarks, while $S_{\mathbf{x}}^A$ and $S_{\mathbf{x}}^B$ are capturing the relative representation at layers A and B based on the landmarks. Then, the feature transformation pattern from layer A to layer B can be simply obtained with the subtraction between the relative representation of two layers:

$$S_{\mathbf{x}}^{A \rightarrow B} = S_{\mathbf{x}}^A - S_{\mathbf{x}}^B. \quad (5)$$

Next, we try to explore how would be the transformation pattern from layer A to layer B if \mathbf{x} was a instance from the source task instead of target task. Firstly, we can have the transformation pattern of each representative center \mathbf{c}_k in the source task by taking all the other centers as landmarks, i.e.,

$$S_{c_k}^{A \rightarrow B} = S_{c_k}^A - S_{c_k}^B. \quad (6)$$

Then we try to approximate the transformation pattern of \mathbf{x} by a weighted linear combination of $\{S_{c_k}^{A \rightarrow B}\}_{k=1}^K$. Formally,

$$\hat{S}_{\mathbf{x}}^{A \rightarrow B} = \sum_{k=1}^K \alpha_k(\mathbf{x}) \cdot S_{c_k}^{A \rightarrow B}, \quad (7)$$

where $\alpha_k(\mathbf{x})$ is the weight corresponding to the k -th center. Here we define it as the probability of \mathbf{x} belongs to the k -th class based on the prediction of the original pre-trained model, i.e.,

$$\alpha_k(\mathbf{x}) = p(\mathcal{M}^0(\mathbf{x}) = k), \quad (8)$$

where \mathcal{M}^0 denotes the predicted class of \mathbf{x} by the pre-trained model \mathcal{M}^0 . We will discuss and compare other possible implementations of $\alpha_k(\mathbf{x})$ in the experiments.

By now we have got $S_{\mathbf{x}}^{A \rightarrow B}$ as the observed feature transformation pattern of \mathbf{x} , and $\hat{S}_{\mathbf{x}}^{A \rightarrow B}$ as the approximated transformation pattern by assuming it as an instance from the source task. The difference between these two patterns reflects the potential contribution to the model adaptation from source task to target task with regard to the representation learning, and is taken to estimate the distinctiveness of \mathbf{x} . Noticing that the two patterns actually are vectors, and there could be various ways to estimate the difference between them. In our case here, because the relative rank correlation is more important than the exact value comparison, we employ the Kendall's tau coefficient [18] to estimate the difference, and finally have the definition of the distinctiveness as

$$Distinctiveness(\mathbf{x}) = \frac{1 - \tau(S_{\mathbf{x}}^{A \rightarrow B}, \hat{S}_{\mathbf{x}}^{A \rightarrow B})}{2} \quad (9)$$

where $\tau(S_{\mathbf{x}}^{A \rightarrow B}, \hat{S}_{\mathbf{x}}^{A \rightarrow B})$ is the Kendall's tau coefficient between $S_{\mathbf{x}}^{A \rightarrow B}$ and $\hat{S}_{\mathbf{x}}^{A \rightarrow B}$.

3.2.2 Uncertainty. Uncertainty is a commonly used criterion in active learning to estimate how uncertain the prediction of the current model is for a given instance. Assume that there are K' classes in the target task. The uncertainty of \mathbf{x} is defined as:

$$Uncertainty(\mathbf{x}) = - \sum_{k'=1}^{K'} p(\mathcal{M}(\mathbf{x}) = k') \cdot (1 - p(\mathcal{M}(\mathbf{x}) = k')), \quad (10)$$

where \mathcal{M} is the current model, and $p(\mathcal{M}(\mathbf{x}) = k')$ is the probability of \mathbf{x} belongs to class k' based on the prediction of \mathcal{M} . In addition to the entropy in Eq. 10, there could be other definitions of the uncertainty criterion, such as the margin to decision boundary or prediction confidence.

3.2.3 Dynamic trade-off. As discussed before, the *distinctiveness* measures the contribution of an instance on improving the network for better representation learning, while the *uncertainty* measures the contribution on improving the classifier corresponding to the last layer of the neural network. To select the most useful instances for better adaptation of the pre-trained network to the target task, we should simultaneously consider the distinctiveness and uncertainty, such that the network can learn better representations as well as stronger classifier for the target task.

One key superiority of deep neural networks to traditional shallow models is that they can learn effective feature representations automatically. During the adaptation of the network, we believe the *distinctiveness* and *uncertainty* play different roles at different stages. At the beginning, the neural network is pre-trained for the source task, and the representation may be less effective for the target task. It is thus urgent to improve the representation at this stage. Moreover, based on the less effective features, it is meaningless to optimize the classifier of the last layer. So we should query more distinctive instances to improve the model towards better feature learning. After more and more queries, the representation is expected to be well adapted to the target task, and thus more uncertain instances are expected to improve the model towards better classification. Based on this motivation, we emphasize more on the *distinctiveness* during the early stages of model adaptation, and gradually increase the attention on *uncertainty*. Formally, we employ a trade-off parameter to dynamically balance the two criteria as the iterations progress:

$$\text{score}(\mathbf{x}) = (1 - \lambda \cdot t) \cdot \text{distinctiveness}(\mathbf{x}) + \lambda \cdot t \cdot \text{uncertainty}(\mathbf{x}), \quad (11)$$

where t is the iteration number.

Finally, the most useful instances with large values of $\text{score}(\cdot)$ will be selected to query their labels, and further used to fine tune the network.

3.3 The ADMA algorithm

The main steps for the proposed ADMA (Active Deep Model Adaptation) algorithm are summarized in Algorithm 1. Firstly, in the source task, one representative center instance is determined for each class. Then the feature transformation patterns from layer A to layer B are calculated for all the centers. After that, the algorithm performs batch-mode active learning to query labels and fine tunes the network iteratively. In the t -th iteration, for each unlabeled instance $\mathbf{x} \in U$, its feature transformation pattern as well as the approximated pattern are calculated, based on which, the *distinctiveness* is further obtained. Then, by combining the *distinctiveness* and *uncertainty*, the final *score* to evaluate the potential contribution of \mathbf{x} is calculated. The unlabeled instances are ranked with regard to this score in descending order, and the top batch Q is selected to query the labels. After that, the queried instances are used to fine tune the network from \mathcal{M}^{t-1} to \mathcal{M}^t . This process is

Algorithm 1 The ADMA algorithm

```

1: Input:
2:    $U$ : the unlabeled set of  $n_u$  instances
3:    $\mathcal{M}^0$ : the pre-trained model
4:    $A$ : the index of the initial layer for feature transformation
5:    $B$ : the index of the end layer for feature transformation
6:    $Z$ : the dataset in the source task used for training  $\mathcal{M}^0$ 
7: Initialization:
8:   Find the centers  $C = \{c_1, \dots, c_K\}$ , one for each class  $k$  in  $Z$ ;

9:   Obtain  $c_k^A$  and  $c_k^B$ : the outputs of  $c_k$  at layers  $A$  and  $B$ ;
10:  Calculate  $S_C^A$  and  $S_C^B$  according to Eq. 3;
11:  Calculate  $S_C^{A \rightarrow B} = S_C^A - S_C^B$ ;
12: Repeat:
13:   For each instance  $\mathbf{x} \in U$ 
14:     Obtain  $\mathbf{x}_k^A$  and  $\mathbf{x}_k^B$ : the outputs of  $\mathbf{x}$  at layers  $A$  and  $B$ ;
15:     Calculate  $S_{\mathbf{x}}^A$  and  $S_{\mathbf{x}}^B$  according to Eq. 4;
16:     Calculate  $S_{\mathbf{x}}^{A \rightarrow B} = S_{\mathbf{x}}^A - S_{\mathbf{x}}^B$ ;
17:     Calculate  $\hat{S}_{\mathbf{x}}^{A \rightarrow B}$  by weighted summarization of  $S_C^{A \rightarrow B}$ ;
18:     Calculate  $\text{Distinctiveness}(\mathbf{x}) = \tau(S_{\mathbf{x}}^{A \rightarrow B}, \hat{S}_{\mathbf{x}}^{A \rightarrow B})$ ;
19:     Calculate  $\text{Uncertainty}(\mathbf{x})$  according to Eq. 10;
20:     Calculate the criterion  $\text{score}(\mathbf{x})$  according to Eq. 11;
21:   End For
22:   Select a batch of instances  $Q$  from  $U$  with largest  $\text{score}$ ;
23:   Query the labels for  $Q$ , and remove  $Q$  from  $U$ ;
24:   Fine-tune the model  $\mathcal{M}^{t-1}$  with the queried data to get  $\mathcal{M}^t$ .
25: Until query budget or expected performance reached.

```

repeated until the the query budget is out or the network achieves a specified performance.

4 EXPERIMENTS

In the experiments, we study with the following three pre-trained models:

- AlexNet [19] is the winning solution of ImageNet Challenge 2012 which has 15.4% top-5 error on ILSVRC dataset. The network consists of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers.
- VGG-16 [30] is the model designed for ImageNet Challenge 2014, which has 7.3% top-5 error on ILSVRC dataset. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other with increasing depth. Max pooling is used to reduce the volume size.
- ResNet-18 [15] is the winning solution of ImageNet Challenge 2015 which has 3.6% top-5 error on ILSVRC dataset. Its architecture contains both plain network and residual network.

When applying the pre-trained models to new tasks, the number of nodes in the last layer is changed to fit the number of classes. For all the pre-trained models, the second last layer is specified as layer B , while the fourth layer from the end is specified as layer A in our approach.

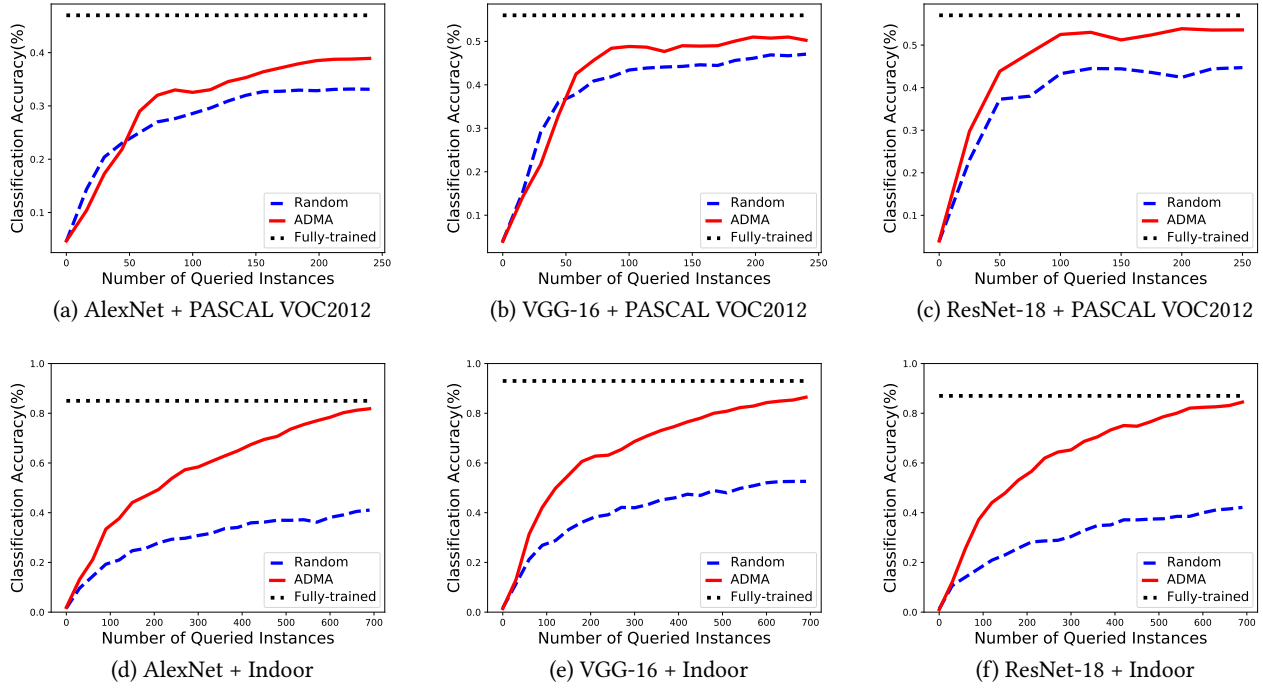


Figure 3: The comparison results on multi-class datasets with different pre-trained models.

We also introduce the datasets involved. All the above introduced models are pre-trained on the ImageNet ILSVRC2012 dataset, which consists of 14,197,122 images from 1000 categories. In Section 4.1, we will perform experiments on two multi-class datasets and two binary classification datasets. PASCAL VOC2012 [9] is an well known image dataset of visual objects in realistic scenes. It consists of 17,125 images from 20 classes. Indoor [26] is a dataset with a total of 15620 images from 67 indoor categories. DOGvsCAT is a dataset from kaggle for classifying 25,000 images between dog and cat [8]. INRIA Person Dataset is a dataset with a total of 1832 images to classify person from it[5]. All the images are resized to 224×224 in order to fit the input of pre-trained models.

The proposed approach is implemented with Pytorch, and the source code has been made public available ¹.

4.1 Performance comparison

We perform the comparison in both multi-class and binary classification tasks. Note that there is no general approach applicable to our setting, we compare the proposed approach with the Random method, which randomly select instances to query their labels. The method AIFT proposed in [42] can only handle binary tasks, and thus is not compared in the multi-class case. Also, the performance of the fully re-trained model is provided for reference. This model use exactly the same architecture of our method but is re-trained with all data from the target task as labeled.

We freeze the first two layers for AlexNet, the first seven layers for VGG-16, and the first eight layers for ResNet-18, and fine tune the other layers. For multi-class problems, the batch size is set to 2 and 10 for VOC and Indoor, respectively. Instead, we query one label at each time for binary tasks, because they are relatively simpler. For VOC and INRIA datasets, we follow the original partition to separate training and test set. For the other two datasets, we select 70% examples as the unlabeled pool for active selection, and the rest 30% as the test set to validate the classification performance.

For multi-class tasks, the accuracy curves are plotted in Figure 3. It can be observed that with different pre-trained models and different datasets, our algorithm consistently outperforms the random sampling method. The results validate the effectiveness of our approach. Especially, it is surprise to observe that by querying less than 5% examples of the dataset, our approach ADMA can achieve comparable performance with the fully trained model using all examples.

We also report the AUC results in Tables 1 and 2. AUC is commonly used to evaluate the performance of multi-class classification. The results are generally consistent with that in Figure 3.

For binary classification tasks, the accuracy curves are plotted in Figure 4. Because the task is relatively easier than that of multi-class learning, the performance curves increase more fast. The proposed ADMA approach can outperforms the other two methods in most cases. The performance of AIFT is mixed. It achieves decent performance with ResNet-18 on DOGvsCAT, and with VGG-16 on INRIA dataset, but loses its edge on the for the other cases.

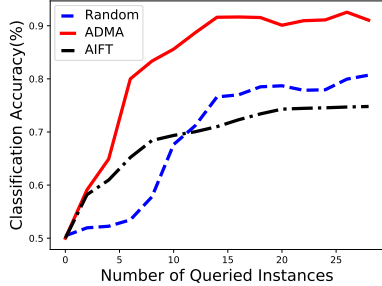
¹The source codes can be downloaded at <http://parnec.nuaa.edu.cn/huangsj/files/KDD18-ADMA.zip>.

Table 1: AUC results on Pascal VOC2012.

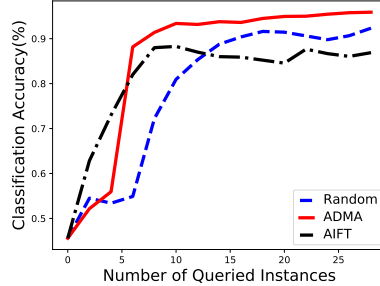
Models	Algorithms	Number of queried instances					
		20	40	60	80	100	120
AlexNet	ADMA	0.676(± 0.003)	0.756(± 0.025)	0.787 (± 0.025)	0.808 (± 0.023)	0.823 (± 0.024)	0.823 (± 0.024)
	RANDOM	0.725 (± 0.012)	0.767 (± 0.018)	0.778(± 0.010)	0.795(± 0.006)	0.806(± 0.004)	0.807(± 0.004)
VGG	ADMA	0.748(± 0.018)	0.863 (± 0.002)	0.886 (± 0.000)	0.891 (± 0.002)	0.896 (± 0.002)	0.897 (± 0.002)
	RANDOM	0.748 (± 0.013)	0.818(± 0.010)	0.855(± 0.019)	0.870(± 0.003)	0.879(± 0.003)	0.880(± 0.004)
ResNet	ADMA	0.820 (± 0.034)	0.876(± 0.005)	0.894(± 0.004)	0.907 (± 0.004)	0.907 (± 0.001)	0.909 (± 0.000)
	RANDOM	0.805(± 0.006)	0.886 (± 0.001)	0.895 (± 0.000)	0.900(± 0.001)	0.903(± 0.001)	0.907(± 0.002)

Table 2: AUC results on Indoor.

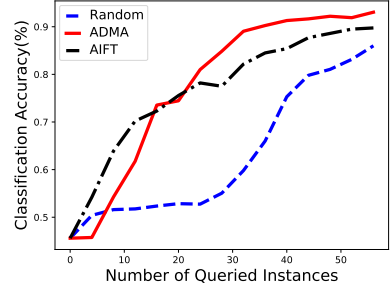
Models	Algorithms	Number of queried instances					
		100	200	300	400	500	600
AlexNet	ADMA	0.858 (± 0.014)	0.924 (± 0.013)	0.945 (± 0.026)	0.958 (± 0.005)	0.964 (± 0.003)	0.967 (± 0.005)
	RANDOM	0.716(± 0.013)	0.781(± 0.010)	0.820(± 0.020)	0.847(± 0.005)	0.863(± 0.003)	0.888(± 0.005)
VGG	ADMA	0.906 (± 0.001)	0.960 (± 0.006)	0.971 (± 0.003)	0.976 (± 0.003)	0.980 (± 0.001)	0.983 (± 0.009)
	RANDOM	0.766(± 0.007)	0.839(± 0.006)	0.871(± 0.009)	0.908(± 0.001)	0.922(± 0.003)	0.929(± 0.003)
ResNet	ADMA	0.834 (± 0.016)	0.938 (± 0.002)	0.961 (± 0.002)	0.9731 (± 0.001)	0.977 (± 0.001)	0.981 (± 0.002)
	RANDOM	0.685(± 0.013)	0.781(± 0.004)	0.826(± 0.002)	0.852(± 0.004)	0.866(± 0.007)	0.877(± 0.007)



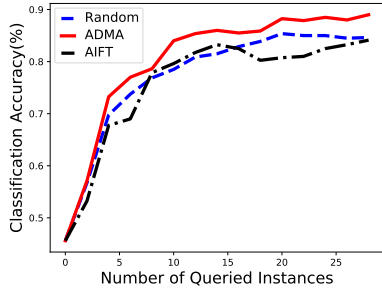
(a) AlexNet + DOG vs CAT



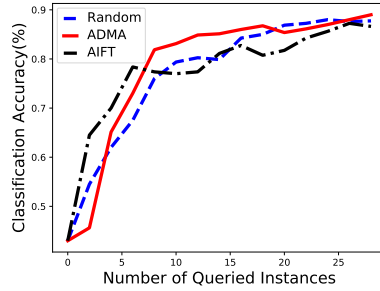
(b) VGG-16 + DOG vs CAT



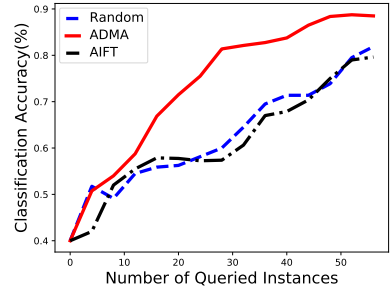
(c) ResNet-18 + DOG vs CAT



(d) AlexNet + INRIA Person Dataset



(e) VGG-16 + INRIA Person Dataset



(f) ResNet-18 + INRIA Person Dataset

Figure 4: The comparison results on binary-class datasets with different pre-trained models.

4.2 Study on the distinctiveness

The proposed criterion *distinctiveness* is calculated based on the feature transformation pattern from layer A to layer B . In this

subsection, we further examine whether the performance can be improved by exploiting multiple patterns. We take AlexNet on PASCAL VOC2012 dataset as an example to perform the experiment.

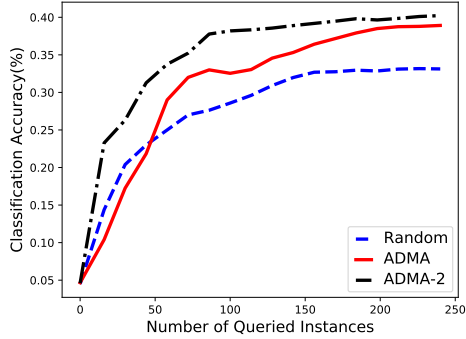


Figure 5: Comparison results of multiple transform patterns.

Specifically, we set two candidate starting layers, i.e. the fourth layer from the end as A_1 and the fifth layer from the end as A_2 . Then two transformation patterns $A_1 \rightarrow B$ and $A_2 \rightarrow B$ are calculated, and the variance between them is used to estimate the *distinctiveness*.

The comparison results are plotted in Figure 5. By exploiting multiple transform patterns between different layer pairs, ADMA-2 can further improve the performance. While we only examine the case with two patterns, it can be expected to achieve further improvements with the ensemble of more layer pairs.

In addition, we try to examine whether the query results well match our motivation that distinctive instances should be preferred. We visualize the data points via t-SNE [23] based on the representation of layer B . Figure 6 (a) and (b) show the queried examples by the proposed ADMA approach and random sampling, respectively. Obviously, the queried images of ADMA have a biased distribution

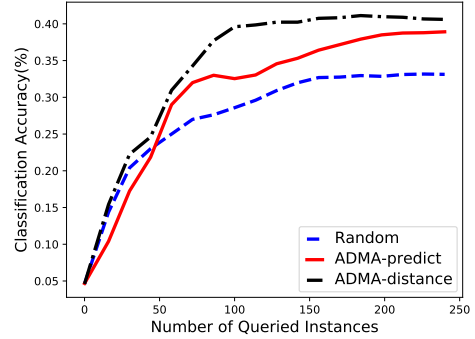


Figure 7: Comparison results of approximation weights.

while the queries of random sampling follow a uniform distribution. We further show some typical images among the queried and not-queried examples of our approach in Figure 6 (c) and (d), respectively. For each pair, the first row is the image queried/not-queried image in the target task, while the second row is the closest center image in the source task. For the queried group, the images in the source and target task have distinctive difference, while images in the not-queried group look similar. This observation is consistent with our motivation on *distinctiveness* that the images capture the unique property of the target task should be better queried.

4.3 Study on the approximation weights

Lastly, we examine the other possible solutions for the weights $\alpha_k(\mathbf{x})$ in Eq. 7 when approximating the transformation pattern of \mathbf{x} with weighted combination of source patterns. In fact, the weight $\alpha_k(\mathbf{x})$ is estimating how likely \mathbf{x} is from the k -th class if we assume it is an example in the source task. In all the above experiments, we calculate $\alpha_k(\mathbf{x})$ as Eq. 8, i.e., use the prediction probability on the k -th class as the weight corresponding to the k -th center. We denote the method with this implementation as ADMA-predict. Here we provide another solution, which calculates the weight $\alpha_k(\mathbf{x})$ by the reciprocal L_2 distance between \mathbf{x} and \mathbf{c}_k based on the representation of layer A . We denote by ADMA-distance. Again we perform the experiment with AlexNet on PASCAL VOC2012, and plot the results in Figure 7. It can be observed that ADMA-distance outperforms ADMA-predict consistently. This is probably because that the pre-trained model may be not reliable if it is directly employed to predict an example from a different task. This is also the reason we need to perform model adaptation across tasks.

5 CONCLUSION

In this paper, we propose an active model adaptation approach for cost-effective training of deep convolutional neural networks. Instead of training from scratch, a pre-trained model can be effectively adapted to a new target task by fine tuning with a few actively queried examples, significantly reducing the cost of designing the network architecture and labeling a large training set. To select the most useful instances for label querying, a novel active selection criterion is proposed, which dynamically balances between distinctiveness and uncertainty. The distinctiveness measures the

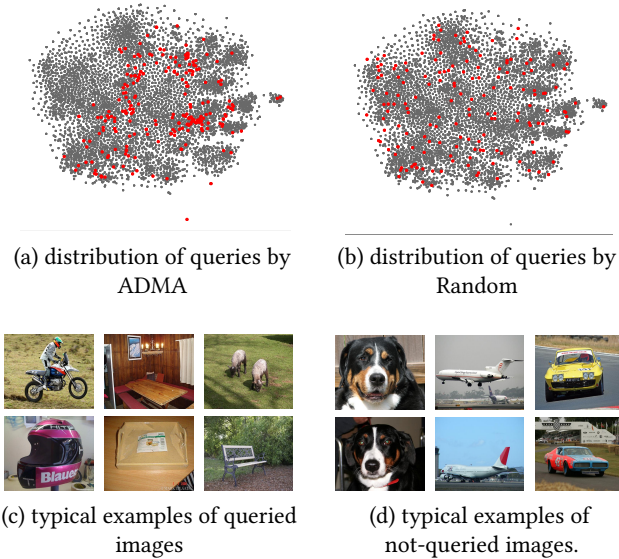


Figure 6: Visualization of the queried images.

potential contribution of an instance on improving the network for better feature representation, while uncertainty measures the capability on improving the classifier of the last layer. Experiments are performed on different datasets with different pre-trained models. The results show that the proposed approach can achieve effective deep network training with significantly lower cost. In the future, we plan to apply the approach on more datasets and more pre-trained models. Also, the distinctiveness criterion will be further studied by considering the feature transformation patterns among different layers.

REFERENCES

- [1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR* abs/1701.07875 (2017). arXiv:1701.07875 <http://arxiv.org/abs/1701.07875>
- [2] Bin Cao, Sinno Jialin Pan, Yu Zhang, Dit-Yan Yeung, and Qiang Yang. 2010. Adaptive Transfer Learning. In *AAAI Conference on Artificial Intelligence*.
- [3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. 2017. Partial Transfer Learning with Selective Adversarial Networks. *CoRR* abs/1707.07901 (2017). arXiv:1707.07901
- [4] Rita Chattopadhyay, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. 2013. Joint transfer and batch-mode active learning. In *International Conference on Machine Learning*. 253–261.
- [5] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1. 886–893.
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.
- [7] Lixin Duan, Ivor W Tsang, and Dong Xu. 2012. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 3 (2012), 465–479.
- [8] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*. ACM, 366–374.
- [9] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2015. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* 111, 1, 98–136.
- [10] Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and Information Systems* 35, 2 (2013), 249–283.
- [11] Yarin Gal and Zoubin Ghahramani. 2015. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. *CoRR* abs/1506.02158 (2015). arXiv:1506.02158
- [12] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian Active Learning with Image Data. In *International Conference on Machine Learning*. 1183–1192.
- [13] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*. 1180–1189.
- [14] Boqing Gong, Kristen Grauman, and Fei Sha. 2013. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International Conference on Machine Learning*. 222–230.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385. arXiv:1512.03385
- [16] David Kale and Yan Liu. 2013. Accelerating active learning with transfer learning. In *IEEE 13th International Conference on Data Mining*. 1085–1090.
- [17] David C. Kale, Marjan Ghazvininejad, Anil Ramakrishna, Jingrui He, and Yan Liu. 2015. Hierarchical active transfer learning. In *The SIAM International Conference on Data Mining*. 514–522.
- [18] William R Knight. 1966. A computer method for calculating Kendall’s tau with ungrouped data. *J. Amer. Statist. Assoc.* 61, 314 (1966), 436–439.
- [19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems* 25. 1097–1105.
- [20] Chun-Liang Li, Chun-Sung Ferng, and Hsuan-Tien Lin. 2015. Active Learning Using Hint Information. *Neural Computation* 27, 8 (Aug. 2015), 1738–1765.
- [21] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *International Conference on Machine Learning*. 97–105.
- [22] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep Transfer Learning with Joint Adaptation Networks. In *International Conference on Machine Learning*. 2208–2217.
- [23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov, 2579–2605.
- [24] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22, 2 (2011), 199–210.
- [25] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.
- [26] Ariadna Quattoni and Antonio Torralba. 2009. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 413–420.
- [27] Ozan Sener and Silvio Savarese. 2017. Active Learning for Convolutional Neural Networks: A Core-Set Approach. *stat* 1050 (2017), 27.
- [28] Burr Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.
- [29] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2017. Adversarial Representation Learning for Domain Adaptation. *CoRR* abs/1707.01217 (2017). arXiv:1707.01217
- [30] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556. arXiv:1409.1556
- [31] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. 2008. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*. 1433–1440.
- [32] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR* abs/1412.3474 (2014). arXiv:1412.3474
- [33] Hanmo Wang, Liang Du, Peng Zhou, Lei Shi, and Yi-Dong Shen. 2015. Convex Batch Mode Active Sampling via α -Relative Pearson Divergence. In *AAAI Conference on Artificial Intelligence*. 3045–3051.
- [34] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. 2017. Cost-Effective Active Learning for Deep Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 12 (2017), 2591–2600.
- [35] Xuezhi Wang, Tzu-Kuo Huang, and Jeff Schneider. 2014. Active transfer learning under model shift. In *International Conference on Machine Learning*. 1305–1313.
- [36] Zheng Wang and Jieping Ye. 2015. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 3 (2015), 17.
- [37] Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3 (2016), 9.
- [38] Lin Yang, Yizhe Zhang, Jianxu Chen, Siyuan Zhang, and Danny Z. Chen. 2017. Suggestive Annotation: A Deep Active Learning Framework for Biomedical Image Segmentation. In *Medical Image Computing and Computer Assisted Intervention*. 399–407.
- [39] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* 27. 3320–3328.
- [40] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*. 819–827.
- [41] Yexun Zhang, Yanfeng Wang, Wenbin Cai, Siyuan Zhou, and Ya Zhang. 2017. From Theory to Practice: Efficient Active Cost-sensitive Classification with Expected Error Reduction. In *The SIAM International Conference on Data Mining*. 153–161.
- [42] Zongwei Zhou, Jae Y. Shin, Lei Zhang, Suryakanth R. Gurudu, Michael B. Gotway, and Jianming Liang. 2017. Fine-Tuning Convolutional Neural Networks for Biomedical Image Analysis: Actively and Incrementally. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 4761–4772.