Learning by Ignoring, with Application to Domain Adaptation

Pengtao Xie [1], Xingchen Zhao [1], and Xuehai He [1]

[1]Affiliation not available

November 8, 2023

Abstract

Learning by ignoring, which identifies less important things and excludes them from the learning process, is broadly practiced in human learning and has shown ubiquitous effectiveness. There has been psychological studies showing that learning to ignore certain things is a powerful tool for helping people focus. In this paper, we explore whether this useful human learning methodology can be borrowed to improve machine learning. We propose a novel machine learning framework referred to as learning by ignoring (LBI). Our framework automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process. We formulate LBI as a three-level optimization framework where three learning stages are involved: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss. A gradient-based algorithm is developed to efficiently solve the three-level optimization problem in LBI. Experiments on various datasets demonstrate the effectiveness of our framework.

Learning by Ignoring, with Application to Domain Adaptation

Xingchen Zhao[†] XIZ168@PITT.EDU
Xuehai He[†] X5HE@ENG.UCSD.EDU
Pengtao Xie[*] P1XIE@ENG.UCSD.EDU
University of California San Diego

Abstract

Learning by ignoring, which identifies less important things and excludes them from the learning process, is broadly practiced in human learning and has shown ubiquitous effectiveness. There has been psychological studies showing that learning to ignore certain things is a powerful tool for helping people focus. In this paper, we explore whether this useful human learning methodology can be borrowed to improve machine learning. We propose a novel machine learning framework referred to as learning by ignoring (LBI). Our framework automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process. We formulate LBI as a three-level optimization framework where three learning stages are involved: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss. An gradient-based algorithm is developed to efficiently solve the three-level optimization problem in LBI. Experiments on various datasets demonstrate the effectiveness of our framework.

1. Introduction

In human learning, a widely-practiced effective learning methodology is learning by ignoring. For example, in course learning, given a large collection of practice problems provided in the textbook, the teacher selects a subset of problems as homework for the students to practice instead of using all problems in the textbook. Some practice problems are ignored because 1) they are too difficult which might confuse the students; 2) they are too simple which are not effective in helping the students to practice their knowledge learned during lectures; 3) they are repetitive. The study in (Cunningham and Egeth, 2016) shows that learning to ignore certain things is powerful for helping people focus.

Drawing inspirations from this effective human-learning method, we are intrigued in exploring whether this method is helpful for training better machine learning models as well. We propose a novel machine learning framework called learning by ignoring (LBI) (as illustrated in Figure 1). In this framework, a model is trained to perform a target task. The model consists of a data encoder and a task-specific head. The encoder is trained in two

. [†]Equal contribution.
. [*]Corresponding author.

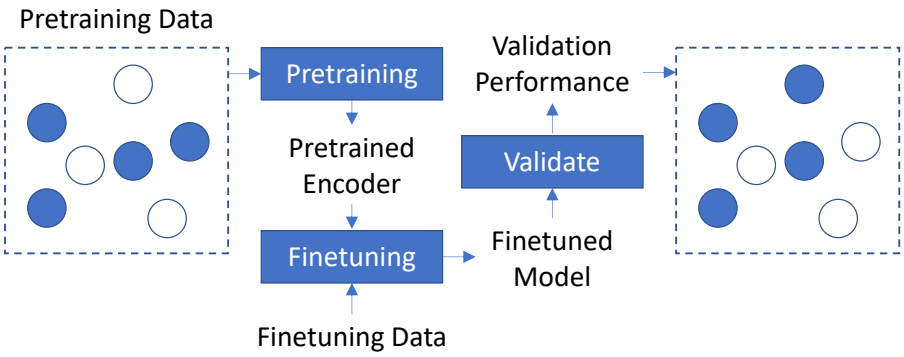

Figure 1: Illustration of learning by ignoring. Void circles denote ignored pretraining data examples. Given a set of intermediately selected pretraining examples, they are used to pretrain a model, which is then finetuned and validated. The validation performance provides guidance on what pretraining examples should be ignored in the next round of learning. This process iterates until convergence.

phrases: pretraining and finetuning, conducted on a pretraining dataset and a finetuning dataset. Pretraining is a commonly used technique in deep learning to learn more effective representations for alleviating overfitting. Given a target task where the amount of training data is limited, training deep neural networks on this small-sized dataset has high risk of overfitting. To address this problem, one can pretrain the feature extraction layers in the network on large-sized external data from some source domain, then finetune these layers on the target data. The abundance of source data enables the network to learn powerful representations that are robust to overfitting. And such representation power can be leveraged to assist in the learning of the target task with more resilience to overfitting.

Some pretraining data examples have a domain difference with the finetuning dataset, rendering them not suitable to pretrain the data encoder that will be used for performing the target task. We would like to identify such out-of-domain data examples and exclude them from the pretraining process. To achieve this goal, we associate each pretraining example with an ignoring variable $a \in [0, 1]$. If a is close to 0, it means that this example should be ignored. We develop a three-level framework (involving three learning stages) to automatically learn these ignoring variables. In the first learning stage, we pretrain a data encoder V by minimizing a weighted pretraining loss: the loss defined on each pretraining example is multiplied with the ignoring variable of this example. If an example x should be ignored, its a is close to 0; multiplying a to the loss of x makes this loss close to 0, which effectively excludes x from the pretraining process. In this stage, these ignoring variables $A = \{a\}$ are fixed. They will be updated at a later stage. Note that the optimally pretrained encoder $V^*(A)$ is a function of A since $V^*(A)$ is a function of the weighted loss and the weighted loss is a function of A. In the second stage, we train another data encoder W on the finetuning dataset. During the training of W, we encourage it to be close to the optimal encoder $V^*(A)$ trained in the first stage by minimizing the squared L2 distance between W and $V^*(A)$. $V^*(A)$ contains information distilled from non-ignored pretraining examples.

Encouraging W to be close to $V^*(A)$ effectively achieves the goal of pretraining W on these non-ignored examples. Note that the optimally trained encoder $W^*(V^*(A))$ is a function of $V^*(A)$. In the third stage, we apply $W^*(V^*(A))$ to make predictions on the validation set of the target task and update A by minimizing the validation loss. To summarize, we aim to learn an ignoring variable for each pretraining example so that the data encoder pretrained on non-ignored examples achieves the optimal performance on the validation set after finetuning. The three stages are conducted end-to-end in a joint manner. Experiments on various datasets demonstrate the effectiveness of our method.

The major contributions of this paper is as follows:

- Drawing inspirations from the ignoring-driven learning methodology of humans, we propose a novel machine learning framework called learning by ignoring (LBI). Our framework automatically identifies pretraining data examples that have large domain shift from the target distribution by learning an ignoring variable for each example and excludes them from the pretraining process.
- We formulate LBT as a multi-level optimization framework involving three learning stages: pretraining by minimizing the losses weighed by ignoring variables; finetuning; updating the ignoring variables by minimizing the validation loss.
- An efficient gradient-based algorithm is developed to solve the LBI problem.
- Experiments on various datasets demonstrate the effectiveness of our method.

2. Related Works

Data Re-weighting and Selection. Several approaches have been proposed for data selection. Matrix column subset selection (Deshpande and Rademacher, 2010; Boutsidis et al., 2009) aims to select a subset of data examples that can best reconstruct the entire dataset. Similarly, coreset selection (Bachem et al.) chooses representative training examples in a way that models trained on the selected examples have comparable performance with those trained on all training examples. These methods perform data selection and model training separately. As a result, the validation performance of the model cannot be used to guide data selection. Ren et al. (2018) proposes a meta learning method to learn the weights of training examples by performing a meta gradient descent step on the weights of the current mini-batch of examples. Shu et al. (2019) propose a method which can adaptively learn an explicit weighting function directly from data. These works focus on selecting training (finetuning) examples using a bi-level optimization framework while our work focuses on selecting pretraining examples using a three-level optimization framework.

Pretraining. Arguably, the most popular pretraining approach is supervised pretraining (SP) (Pan and Yang, 2009), which learns the weight parameters of a representation network by solving a supervised source task, i.e., correctly mapping input data examples to their labels (e.g., classes, segmentation masks, etc.). Recently, self-supervised learning (Oord et al., 2018; He et al., 2019; Chen et al., 2020; Misra and Maaten, 2020), as an unsupervised pretraining approach, has achieved promising success and outperforms supervised pretraining in a wide range of applications. Similar to SP, self-supervised pretraining (SSP) also

solves predictive tasks. But the output labels in SSP are constructed from the input data, rather than annotated by humans as in SP. The auxiliary predictive tasks in SSP could be predicting whether two augmented data examples originate from the same original data example (Hénaff et al., 2019; He et al., 2019; Chen et al., 2020; Misra and Maaten, 2020), inpainting masked regions in images (Pathak et al., 2016), etc.

Domain Adaptation. Domain adaptation (Pan et al., 2010; Ganin et al., 2016a; Bousmalis et al., 2017) considers the problem of transferring knowledge between two domains with distinctive data distributions. There are mainly two approaches to address the domain adaptation problem. One way is by using metric learning, where a distance metric is defined to measure the distribution discrepancy between domains and the target of training is to minimize the distance (Sun and Saenko, 2016; Long et al., 2017c; Ben-David et al., 2010). Another way is by doing adversarial domain adaptation (Hoffman et al., 2018a; Long et al., 2017a). Instead of resorting to metric learning which explicitly optimizes a similarity function, it learns a domain discriminator and a feature learning network adversarially. The domain discriminator is trained to tell whether an instance is from the source domain or the target domain, while the feature learning network learns domain-invariant features and is trained to fool the domain discriminator.

Apart from the classic domain adaptation settings, in recent years, there are also works focusing on unsupervised domain adaptation (UDA), which gives predictions for the unlabeled target data, where labels only exist in the source data. There are mainly three types of methods for unsupervised domain adaptation in computer vision. The first one uses a model trained on the labeled source data to estimate labels on the target data, then trains the model using pseudo target labels (Kang et al., 2019); the second one is to induce alignment between the source and the target domains in feature spaces, e.g. (Long et al., 2015b, 2017d) propose to minimize the Maximum Mean Discrepancy (MMD) between the source and target domain in the deep neural network; The third type uses generative models to transform the source images to resemble the target images (Hoffman et al., 2018b), which operates on image pixels rather than on an intermediate representation space like the first type of methods. Different from these approaches for UDA, in this paper, we use labels both from source dataset and target dataset. We are also the first to use learning by ignoring on domain adaptation problems.

3. Methods

Our framework aims to learn a machine learning model for accomplishing a target task T. The ML model is composed of a data encoder W and a task-specific head H. For instance, in a text classification task, the data encoder can be a BERT (Devlin et al., 2018) model which produces an embedding of the input text and the classification head can be a multilayer perceptron which predicts the class label of this text based on its embedding. The learning is performed in two phrases: pretraining and finetuning. In the pretraining phrase, we pretrain the data encoder W on a pretraining dataset $D^{(\text{pre})} = \{d_i^{(\text{pre})}\}_{i=1}^{M}$ by solving a pretraining task P. P could be the same as T. In this case, W and the task-specific head H are trained jointly on $D^{(\text{pre})}$. P could be different from T as well. Under such circumstances, P has its own task-specific head J while sharing the same encoder W with T. J and W are trained jointly on $D^{(\text{pre})}$. Oftentimes, some pretraining data examples have

Table 1: Notations in learning by ignoring

Notation	Meaning
M	Number of pretraining examples
N	Number of finetuning examples
O	Number of validation examples
$d_i^{(\text{pre})}$	The i-th pretraining data example
a_i	Ignoring variable of $d_i^{(\text{pre})}$ in pretraining
b_i	Ignoring variable of $d_i^{(\text{pre})}$ in finetuning
$d_i^{(\text{tr})}$	The i-th finetuning data example
$d_i^{(\text{val})}$	The i-th validation example
W	Encoder in finetuning
V	Encoder in pretraining
H	Head of the target task
J	Head of the pretraining task

a domain shift with the finetuning dataset. This domain shift renders these examples not suitable for pretraining the encoder. We aim to automatically identify such examples using ignoring variables and exclude them from the pretraining process. To achieve this goal, we multiply the loss of a pretraining example x with an ignoring variable $a \in [0, 1]$. If a is close to 0, it means that this example should be ignored; accordingly, the loss (after multiplied with a) is made close to 0, which effectively excludes x from the pretraining process. We aim to automatically learn the values of these ignoring variables, which will be detailed later. After pretraining, the encoder is finetuned on the training dataset $D^{(\text{tr})} = \{d_i^{(\text{tr})}\}_{i=1}^N$. For mathematical convenience, we formulate "pretraining" in the following way: in the pretraining phrase, we train another encoder V; in the finetuning phrase, we encourage the encoder W to be close to the optimally pretrained encoder V^* where the closeness is measured using squared L2 distance $\|W - V^*\|_2^2$.

Overall, the learning is performed in three stages. In the first stage, the model trains the encoder V and the head J specific to the pretraining task P on the pretraining dataset $D^{(\text{pre})} = \{d_i^{(\text{pre})}\}_{i=1}^M$, with the ignoring variables $A = \{a_i\}_{i=1}^M$ fixed:

$$V^*(A) = \min_{V,J} \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}). \tag{1}$$

After training, the optimal head is discarded. The optimal encoder $V^*(A)$ is retained for future use. The ignoring variables A are needed to make predictions and calculate training losses. But they should not be updated in this stage. Otherwise, the values of A will all be zero. Note that $V^*(A)$ is a function of A since it is a function of $\sum_{i=1}^M a_i L(V, J, d_i^{(\text{pre})})$ and $\sum_{i=1}^M a_i L(V, J, d_i^{(\text{pre})})$ is a function of A.

In the second stage, the model trains its data encoder W and task-specific head H by minimizing the training loss of the target task T. During training, W is encouraged to be close to $V^*(A)$ trained in the pretraining phrase by minimizing the squared L2 distance $\|W - V^*(A)\|_2^2$. This implicitly achieves the effect of pretraining W on the non-ignored

pretraining examples. The optimization problem in the second stage is:

$$W^*(V^*(A)), H^* = \min_{W,H} \sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2, \tag{2}$$

where λ is a tradeoff parameter. Note that $W^*(V^*(A))$ is a function of $V^*(A)$ since it is a function of $\|W - V^*(A)\|_2^2$, which is a function of $V^*(A)$.

In the third stage, we use the trained model consisting of $W^*(V^*(A))$ and H^* to make predictions on the validation dataset $D^{(\text{val})}$ of the target task T. We update A by minimizing the validation loss.

$$\min_A \sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})}). \tag{3}$$

The three stages mutually influence each other: $V^*(A)$ trained in the first stage is needed to calculate the loss function in the second stage; $W^*(V^*(A))$ trained in the second stage is needed to calculate the objective function in the third stage; the ignoring variables A updated in the third stage alter the loss function in the first stage, which subsequently changes $V^*(A)$ and $W^*(V^*(A))$ as well.

Putting the three learning stages together, we formulate LBI as the following three-level optimization problem:

$$\begin{array}{ll} \min_A & \sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})}) \\ s.t. & W^*(V^*(A)), H^* = \min_{W,H} \sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 \\ & V^*(A) = \min_{V,J} \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \end{array} \tag{4}$$

This formulation consists of three optimization problems. The two inner optimization problems (on the constraints) represent the first and second learning stage respectively. The outer optimization problem represents the third learning stage. The three stages are illustrated in Figure 2.

If the pretaining task and target task are the same, in the second stage we can use the pretraining data to train W as well. Due to domain difference, not all pretraining examples are suitable for training W. To exclude such examples, we associate each pretraining example $d_i^{(\text{pre})}$ with an ignoring variable $b_i \in [0, 1]$. Note that b_i is different from a_i. b_i is used to determine whether $d_i^{(\text{pre})}$ should be ignored during training W and a_i is used to determine whether $d_i^{(\text{pre})}$ should be ignored during training V. The corresponding formulation is:

$$\begin{array}{ll} \min\limits_{A,B} & \sum_{i=1}^{O} L(W^*(V^*(A), B), H^*(B), d_i^{(\text{val})}) \\ s.t. & W^*(V^*(A), B), H^*(B) = \min\limits_{W,H} \sum\limits_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum\limits_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}) \\ & V^*(A) = \min\limits_{V,J} \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}) \end{array} \tag{5}$$

where γ is a tradeoff parameter and $B = \{b_i\}_{i=1}^{M}$. Note that W^* and H^* are both functions of B.

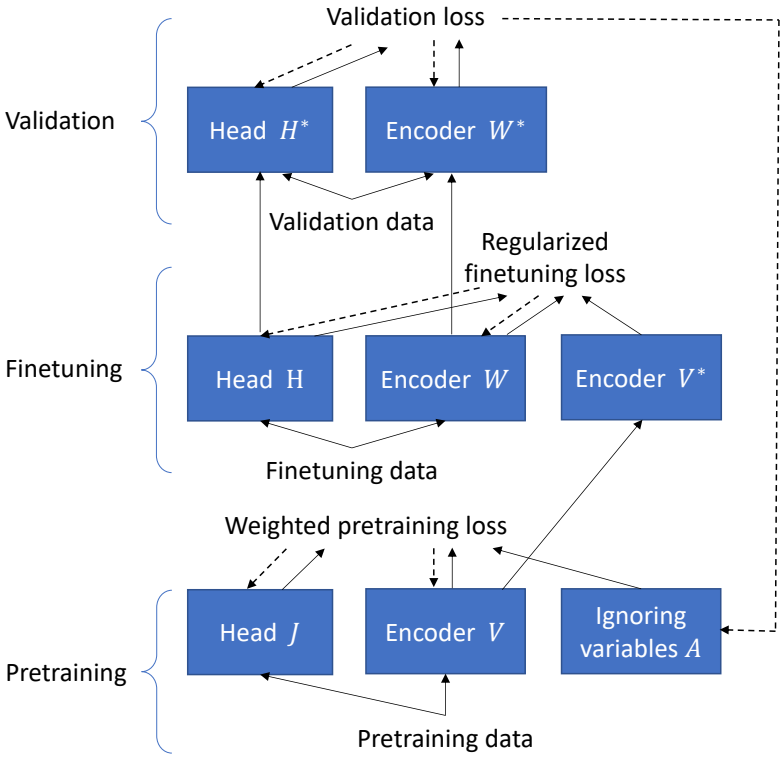

Figure 2: Learning by ignoring. Along the solid arrows, predictions are made and train/validation losses are calculated. Along the dotted arrows, gradients are calculated and parameters (including ignoring variables and network weights) are updated.

3.1. Optimization Algorithm

In this section, we develop a gradient-based optimization algorithm to solve the three-level optimization problem in Eq.(4). Drawing inspirations from (Liu et al., 2018), we approximate $V^*(A)$ using:

$$V' = V - \xi_V \nabla_V \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}), \tag{6}$$

where ξ_V is a learning rate. We update J as:

$$J \leftarrow J - \xi_J \sum_{i=1}^{M} a_i \nabla_J L(V, J, d_i^{(\text{pre})}). \tag{7}$$

Substituting V' into $\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2$, we obtain an approximated objective $\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2$. Then we calculate the gradient of O_W w.r.t

to W and approximate $W^*(V^*(A))$ using one-step gradient descent update of W:

$$W' = W - \xi_W \nabla_W (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2) \tag{8}$$

Similarly, we approximate H^* with:

$$H' = H - \xi_H \nabla_H (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V'\|_2^2) \tag{9}$$

Finally, we substitute W' and H' into $\sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})})$ and get $O_A = \sum_{i=1}^{O} L(W', H', d_i^{(\text{val})})$. We calculate the gradient of O_A w.r.t A and update A by descending the gradient:

$$A \leftarrow A - \xi_A \nabla_A \sum_{i=1}^{O} L(W', H', d_i^{(\text{val})}) \tag{10}$$

where

$$\begin{aligned} \frac{\partial V'}{\partial A} &= \frac{\partial (V - \xi_V \nabla_V \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}))}{\partial A} \\ &= -\xi_V \nabla_{A,V}^2 \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})}), \end{aligned} \tag{11}$$

and

$$\begin{aligned} \frac{\partial W'}{\partial V'} &= \frac{\partial (W - \xi_W (\sum_{i=1}^{N} \nabla_W L(W, H, d_i^{(\text{tr})}) + 2\lambda (W - V')))}{\partial V'} \\ &= 2\xi_W \lambda I, \end{aligned} \tag{12}$$

where I is an identity matrix.

For the formulation in Eq.(5), the optimization algorithm is similar. $V^*(A)$ is approximated using Eq.(6). We approximate $W^*(V^*(A), B)$ using

$$W' = W - \xi_W \nabla_W (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})})), \tag{13}$$

and approximate $H^*(B)$ using

$$H' = H - \xi_H \nabla_H (\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})})). \tag{14}$$

The update of A is the same as that in Eq.(10). The update of B is as follows:

$$\begin{aligned} B &\leftarrow B - \xi_B \nabla_B \sum_{i=1}^{O} L(W', H', d_i^{(\text{val})}) \\ &= B - \xi_B \sum_{i=1}^{O} (\frac{\partial W'}{\partial B} \nabla_{W'} L(W', H', d_i^{(\text{val})}) + \frac{\partial H'}{\partial B} \nabla_{H'} L(W', H', d_i^{(\text{val})})), \end{aligned} \tag{15}$$

where

$$\frac{\partial W'}{\partial B} = -\xi_W \gamma \nabla_{B,W}^2 \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}), \tag{16}$$

and

$$\frac{\partial H'}{\partial B} = -\xi_H \gamma \nabla_{B,H}^2 \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}). \tag{17}$$

Table 2: Accuracy (%) on the Office31 dataset. In the $A \rightarrow B$ notion, A denotes the source data and B denotes the target data. 1, 2, 3 in DAN, DANN, CDAN, MME denote unsupervised, semi-supervised, and supervised domain adaptation settings respectively.

			A→W	A→D	D→W	D→A	W→D	W→A	Average
DAN-1 (Long et al., 2015a)			78.11	67.27	91.12	50.66	98.18	53.85	73.20
DAN-2 (Long et al., 2015a)			95.86	98.18	96.45	86.12	97.27	87.05	93.49
DAN-3 (Long et al., 2015a)			97.04	96.36	97.04	**88.74**	**100.0**	87.24	94.40
DANN-1 (Ganin et al., 2016b)			78.70	68.18	87.57	46.72	94.55	50.28	71.00
DANN-2 (Ganin et al., 2016b)			94.08	93.64	94.67	85.55	95.45	84.99	91.40
DANN-3 (Ganin et al., 2016b)			96.45	94.55	**97.63**	86.68	**100.0**	85.37	93.45
CDAN-1 (Long et al., 2017b)			83.84	71.82	94.08	56.47	97.27	60.98	77.41
CDAN-2 (Long et al., 2017b)			94.67	95.45	96.45	86.68	97.27	84.80	92.55
CDAN-3 (Long et al., 2017b)			97.04	95.45	**97.63**	85.93	99.09	87.24	93.73
MME-1 (Saito et al., 2019)			60.95	50.91	88.17	40.71	95.45	41.28	62.91
MME-2 (Saito et al., 2019)			95.86	96.36	96.45	84.05	99.09	85.74	92.93
MME-3 (Saito et al., 2019)			94.67	90.90	**97.63**	87.24	**100.0**	84.43	92.48
	Pretrain	Finetune							
Ablation 1	No	No source	96.45	97.48	**97.63**	85.46	99.09	85.37	93.58
Ablation 2	No	Full source	96.45	93.64	**97.63**	87.43	99.09	86.12	93.39
Ablation 3	No	Weighted source	98.22	97.27	**97.63**	86.68	98.18	86.87	94.14
Ablation 4	Full source	No source	97.04	97.27	**97.63**	85.74	98.18	86.38	93.71
Ablation 5	Full source	Full source	95.86	95.45	97.04	87.05	**100.0**	86.30	93.62
Ablation 6	Full source	Weighted source	97.63	96.36	**97.63**	88.18	99.09	**87.99**	94.48
Ablation 7	Weighted source	No source	96.45	**99.09**	97.04	86.49	99.09	86.49	94.11
Ablation 8	Weighted source	Full source	95.86	91.82	**97.63**	88.37	**100.0**	87.05	93.46
Full LBI (ours)	Weighted source	Weighted source	**98.82**	**99.09**	97.04	87.80	99.09	86.87	**94.79**

Algorithm 1 Optimization algorithm for LBI

while *not converged* **do**

1. Update encoder V in pretraining phrase using Eq.(6)
2. Update head J in pretraining using Eq.(7)
3. Update encoder W in finetuning phrase using Eq.(8)
4. Update head H in finetuning using Eq.(9)
5. Update ignoring variables A using Eq.(10)

end

4. Experiments

In this section, we present experimental results. Please refer to the supplements for more hyperparameter settings and additional results.

4.1. Datasets

We perform experiments on three datasets: OfficeHome (Venkateswara et al., 2017) Office31 (Saenko et al., 2010), and ImageCLEF (Müller et al., 2010). OfficeHome consists of 15,500 images of daily objects from 65 categories and 4 domains, including Art (Ar), Clipart

Table 3: Accuracy (%) on the ImageCLEF dataset. In the $A \rightarrow B$ notion, A denotes the source data and B denotes the target data. 1, 2, 3 in DAN, DANN, CDAN, MME denote unsupervised, semi-supervised, and supervised domain adaptation settings respectively.

			C→P	C→I	P→C	P→I	I→C	I→P	Average
DAN-1 (Long et al., 2015a)			68.15	76.40	89.19	78.88	88.51	66.88	78.00
DAN-2 (Long et al., 2015a)			61.78	83.85	93.92	83.85	92.57	63.06	79.84
DAN-3 (Long et al., 2015a)			64.33	81.37	93.25	85.71	93.24	68.79	81.12
DANN-1 (Ganin et al., 2016b)			61.78	73.29	84.46	77.02	75.68	66.24	73.08
DANN-2 (Ganin et al., 2016b)			63.06	83.23	93.92	85.71	93.24	64.33	80.58
DANN-3 (Ganin et al., 2016b)			68.43	83.85	95.27	88.20	91.89	69.26	82.82
CDAN-1 (Long et al., 2017b)			64.97	70.81	70.27	78.26	90.54	64.33	73.20
CDAN-2 (Long et al., 2017b)			64.42	85.71	95.27	85.09	95.27	67.52	82.21
CDAN-3 (Long et al., 2017b)			64.97	87.58	93.24	**89.44**	94.59	67.52	82.89
MME-1 (Saito et al., 2019)			57.96	68.32	75.00	72.05	85.14	64.97	70.57
MME-2 (Saito et al., 2019)			61.15	80.12	93.92	81.99	93.92	59.97	78.51
MME-3 (Saito et al., 2019)			64.97	80.75	88.51	84.47	91.22	61.78	78.62
	Pretrain	Finetune							
Ablation 1	No	No source	59.87	86.34	91.89	84.47	91.22	65.61	79.90
Ablation 2	No	Full source	68.02	86.34	93.24	86.04	90.54	67.39	81.93
Ablation 3	No	Weighted source	67.52	88.20	94.59	84.47	95.95	66.88	82.94
Ablation 4	Full source	No source	62.42	87.28	93.87	83.79	93.72	63.43	80.75
Ablation 5	Full source	Full source	68.79	86.96	94.59	85.09	93.92	68.15	82.92
Ablation 6	Full source	Weighted source	68.15	**89.44**	**95.45**	86.95	**97.30**	68.15	84.24
Ablation 7	Weighted source	No source	63.06	87.58	93.23	86.34	93.24	64.33	81.30
Ablation 8	Weighted source	Full source	66.88	85.09	93.24	85.71	94.59	66.88	82.07
Full LBI (ours)	Weighted source	Weighted source	**70.70**	**89.44**	93.24	87.58	95.95	**70.06**	**84.50**

(Cl), Product (Pr), and Real-World (Rw). Each category has an average of about 70 images and a maximum of 99 images. Office31 contains 4,110 images belonging to 31 classes and 3 domains, including Amazon website (A), web camera (W), and digital SLR camera (D). The number of images in domain A, W, and D is 2817, 795, and 498 respectively. ImageCLEF consists of 1,800 images from 3 datasets (domains) with 12 classes, including Caltech-256 (C), ILSVRC2012 (I), and Pascal VOC2012 (P). We split OfficeHome, Office31, and ImageCLEF into train/validation/test sets with a ratio of 5:3:2, 6:2:2, and 6:2:2 respectively. For each dataset, we perform domain adaptation (Long et al., 2015a) studies: one domain is selected as source and another domain is selected as target; data in the source domain is leveraged to help with model training in the target domain. In the learning-by-ignoring framework, source data is used as pretraining data and target data is used as finetuning data.

4.2. Baselines

We compare with the following baselines. For each baseline, we experimented with three domain adaptation (DA) settings: 1) unsupervised DA, where the labels of source examples are used for DA and the labels of target examples are not; 2) semi-supervised DA, where the labels of target examples are used for DA and the labels of source examples are not; 3)

Table 4: Accuracy (%) on the OfficeHome dataset.

			Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar
DAN-1 (Long et al., 2015a)			30.86	46.25	54.19	33.54	41.22	45.03	33.33
DAN-2 (Long et al., 2015a)			67.66	86.77	72.74	57.51	87.59	69.50	62.14
DAN-3 (Long et al., 2015a)			68.46	**87.94**	75.53	59.05	85.13	72.29	60.29
DANN-1 (Ganin et al., 2016b)			33.37	43.56	55.75	40.95	47.66	50.50	37.24
DANN-2 (Ganin et al., 2016b)			67.09	83.37	73.63	55.56	82.79	69.83	58.64
DANN-3 (Ganin et al., 2016b)			68.80	85.60	72.96	60.08	86.53	72.74	59.88
CDAN-1 (Long et al., 2017b)			37.49	49.77	59.55	41.56	51.99	54.41	41.77
CDAN-2 (Long et al., 2017b)			68.23	84.43	73.41	60.49	84.66	71.84	59.67
CDAN-3 (Long et al., 2017b)			67.31	86.65	73.52	61.93	84.66	71.06	61.93
MME-1 (Saito et al., 2019)			28.11	40.40	52.40	29.84	35.71	40.34	29.22
MME-2 (Saito et al., 2019)			68.23	84.66	70.73	55.35	85.01	66.37	58.44
MME-3 (Saito et al., 2019)			69.60	85.01	71.96	56.97	84.66	67.93	52.47
	Pretrain	Finetune							
Ablation 1	No	No source	66.40	87.47	73.18	65.23	87.24	72.96	64.40
Ablation 2	No	Full source	68.69	85.36	74.75	60.08	86.07	72.29	59.47
Ablation 3	No	Weighted source	68.00	86.42	75.08	62.14	86.53	73.52	63.37
Ablation 4	Full source	No source	69.71	**87.94**	**76.65**	67.28	88.17	76.54	65.02
Ablation 5	Full source	Full source	67.54	85.13	74.75	60.08	84.54	71.17	61.11
Ablation 6	Full source	Weighted source	68.00	84.66	74.64	61.73	86.3	71.84	63.17
Ablation 7	Weighted source	No source	**70.51**	87.70	76.20	**69.75**	**89.34**	**77.65**	**68.52**
Ablation 8	Weighted source	Full source	67.66	85.25	74.41	61.52	86.77	73.30	61.52
Full LBI (ours)	Weighted source	Weighted source	69.03	84.66	75.42	61.32	85.83	73.52	62.76

			Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Average
DAN-1 (Long et al., 2015a)			28.69	52.96	51.03	33.37	65.69	43.01
DAN-2 (Long et al., 2015a)			70.40	74.19	65.23	68.11	86.07	72.33
DAN-3 (Long et al., 2015a)			70.51	75.75	63.79	69.14	86.89	72.90
DANN-1 (Ganin et al., 2016b)			33.37	56.54	52.67	42.74	70.73	47.09
DANN-2 (Ganin et al., 2016b)			65.60	72.29	60.70	68.00	86.07	70.30
DANN-3 (Ganin et al., 2016b)			68.80	73.18	64.61	68.46	87.70	72.45
CDAN-1 (Long et al., 2017b)			41.14	58.21	56.58	41.83	73.54	50.65
CDAN-2 (Long et al., 2017b)			69.49	73.41	63.58	67.54	88.06	72.07
CDAN-3 (Long et al., 2017b)			**70.74**	72.63	65.23	69.94	87.35	72.75
MME-1 (Saito et al., 2019)			26.97	48.04	46.71	33.71	61.71	39.43
MME-2 (Saito et al., 2019)			67.31	69.50	59.26	66.51	86.18	69.80
MME-3 (Saito et al., 2019)			69.14	73.07	62.96	68.80	85.83	70.70
	Pretrain	Finetune						
Ablation 1	No	No source	68.91	73.85	60.70	69.14	86.77	73.02
Ablation 2	No	Full source	67.77	74.75	66.16	69.60	87.47	72.71
Ablation 3	No	Weighted source	69.94	74.53	65.02	**70.17**	89.23	73.66
Ablation 4	Full source	No source	69.71	76.20	**68.96**	69.37	88.06	75.30
Ablation 5	Full source	Full source	66.74	71.84	64.61	66.63	87.00	71.76
Ablation 6	Full source	Weighted source	66.97	74.19	65.43	67.54	88.41	72.74
Ablation 7	Weighted source	No source	70.63	**77.65**	67.90	69.83	**90.52**	**76.35**
Ablation 8	Weighted source	Full source	67.43	74.19	68.31	68.34	86.18	72.91
Full LBI (ours)	Weighted source	Weighted source	69.94	74.41	67.28	67.89	86.18	73.19

supervised DA, where both the labels of source examples and target examples are used for DA.

- **DAN** (Long et al., 2015a), which matches mean embeddings of different domain distributions in a reproducing kernel Hilbert space.
- **DANN** (Ganin et al., 2016b), which uses adversarial learning to learn representations so that source domain and target domain are not distinguishable.
- **CDAN** (Long et al., 2017b), which conditions adversarial adaptation on discriminative information.

- **MME** (Saito et al., 2019), which is a semi-supervised domain adaptation method based on minimax entropy.

4.3. Experimental Settings

We use ResNet-34 (He et al., 2016) pretrained on ImageNet (Deng et al., 2009) as the backbone of our methods. Our models were trained using the Adam (Kingma and Ba, 2014) optimizer, with a batch size of 64, a learning rate of 1e-4 for the feature extractor, a learning rate of 1e-3 for the classifier, a weight decay of 5e-4, for 50 epochs. The learning rate was decreased by a factor of 10 after 40 epochs. In learning by ignoring, γ is set to 1 for all datasets; λ is set to 7e-3 for OfficeHome, 5e-4 for Office31, and 3e-3 for ImageCLEF.

4.4. Results

Table 2 shows the results on the Office31 dataset. In the $A \rightarrow B$ notion, A denotes the source dataset and B denotes the target dataset. As can be seen, our proposed LBI method achieves better averaged accuracy than the domain adaptation (DA) baselines including DAN, DANN, CDAN, and MME. The major reason is that our method automatically identifies source examples that are not suitable for pretraining and excludes them from the pretraining process. In contrast, these DA baselines adapt all source examples into the target domain and lack the ability of explicitly identifying source examples that are not suitable for domain adaptation. Table 2 shows the results on the ImageCLEF dataset. Our proposed LBI achieves better average accuracy than the DA baselines, thanks to its mechanism of ignoring source examples that are not suitable for helping with the learning on the target domain. Table 4 shows the results on the OfficeHome dataset. LBI outperforms all DA methods, which further demonstrates the effectiveness of our proposed learning-by-ignoring framework. On this dataset, using source dataset for finetuning leads to worse performance. This is probably because the source domains in this dataset have large domain shifts with the target domains. Therefore, using source data directly for finetuning may not be proper. However, using non-ignored source data for pretraining is helpful.

4.5. Ablation Studies

We perform ablation studies to check the effectiveness of individual modules in our framework. Unless otherwise notified, γ was set to 1 for all three datasets; λ was set to 7e-3 for OfficeHome, 5e-4 for Office31, and 3e-3 for ImageCLEF.

- Ablation setting 1: no pretraining, finetune on target data only. We ignore the source dataset and directly train a model on the target dataset. This is equivalent to setting both λ and γ in LBI (Eq.(5)) to 0.
- Ablation setting 2: no pretraining, finetune on target data and all source examples. There is no pretraining; we combine the source dataset and the target dataset as a single dataset, then train a model on the combined dataset. This is equivalent to setting λ to 0 and setting all ignoring variables in B to 1 in LBI (Eq.(5)).
- Ablation setting 3: no pretraining, finetune on target data and weighted source examples. There is no pretraining; the model is directly trained on all target examples

and selected source examples. This is equivalent to setting the tradeoff parameter λ to 0 in the LBI framework (Eq.(5)).

- Ablation setting 4: pretrain on all source examples, finetune on target data only. We first train a model M_1 on the full source dataset. Then we train a model M_2 on the target dataset. When training M_2, we encourage its weights to be close to the optimally trained weights of M_1 by minimizing their squared L2 distance. This is equivalent to setting γ to 0 and setting all ignoring variables in A to 1 in LBI (Eq.(5)).
- Ablation setting 5: pretrain on all source examples, finetune on target data and all source examples. This setting is similar to ablation 4, except that M_2 is trained on the target and full source dataset. This is equivalent to setting all ignoring variables in A and B to 1 in LBI (Eq.(5)).
- Ablation setting 6: pretrain on all source examples, finetune on target data and weighted source examples. This setting is similar to ablation 4, except that M_2 is trained on the target and weighted source dataset. This is equivalent to setting all ignoring variables in A to 1 in LBI (Eq.(5)).
- Ablation setting 7: pretrain on weighted source examples, finetune on target data only. This is equivalent to setting the tradeoff parameter γ to 0 in the LBI framework (Eq.(5)).
- Ablation setting 8: pretrain on weighted source examples, finetune on target data and all source examples. In this setting, all source examples are used for finetuning, without ignoring any of them. This is equivalent to setting all ignoring variables in B to 1 in the LBI framework (Eq.(5)).
- Ablation study on λ. We explore how the performance varies as the tradeoff parameter λ in Eq.(5) increases. In this study, the other tradeoff parameter γ in Eq.(5) is set to 1. We report the average accuracy on the validation set. The experiments are conducted on ImageCLEF and OfficeHome. Please refer to the supplements for results on OfficeHome.
- Ablation study on γ. We explore how the performance varies as γ increases. We report the average accuracy on the validation set. The experiments are conducted on ImageCLEF and OfficeHome. In this study, the other tradeoff parameter λ is set to 3e-3 for ImageCLEF and 7e-3 for OfficeHome. Please refer to the supplements for results on OfficeHome.

Table 2 shows the ablation study results on the Office31 dataset. From this table, we make the following observations. **First**, during pretraining, ignoring certain source examples is better than using all source examples. For instance, ablation 7 achieves better average accuracy than ablation 4 where the former performs pretraining on weighted source data and the latter uses all source examples for pretraining. The rest settings of ablation 7 and 4 are the same. As another example, our proposed full LBI framework achieves higher average accuracy than ablation 6 where the former performs ignoring of certain source examples during pretraining while the latter does not. The rest settings of full LBI are the same as those of ablation 6. The reason is that due to domain shift between source and target, some source examples are largely different from the target examples; if using such

source examples for pretraining, the pretrained encoder may be biased to the source distribution and cannot well represent target examples. Our proposed approaches automatically identify source examples that have large domain discrepancy with the target and remove them from the pretraining process. Consequently, the pretrained encoder with ignoring is better than the one pretrained using all source examples without ignoring. **Second**, when using source examples to finetune the encoder (together with target examples), it is beneficial to ignore some source examples in the finetuning process. As can be seen, in terms of average accuracy, the full LBI framework performs better than ablation 8 where the former ignores certain source examples during finetuning while the latter does not. The rest settings of these two methods are the same. Similarly, ablation 6 outperforms ablation 5; ablation 3 achieves higher average accuracy than ablation 2. Again, the reason is that due to domain shift, some source examples are not suitable for finetuning the encoder of the target dataset. Our proposed ignoring approach excludes such source examples from finetuning, hence achieving better performance than not using ignoring. **Third**, without the ignoring mechanism, using source examples for finetuning is harmful. For example, ablation 2 which uses all source samples for finetuning achieves lower average accuracy than ablation 1 which uses no source example for finetuning. Similarly, ablation 5 achieves worse average accuracy than ablation 4; ablation 8 performs less well than ablation 7. However, once we add the ignoring mechanism and uses non-ignored source examples for finetuning, the resulting average accuracy is higher than not using any source examples for finetuning, as can be seen from the average accuracy results that ablation 3 outperforms ablation 1; ablation 6 outperform ablation 4; and the full LBI method outperforms ablation 7. This further demonstrates the effectiveness of our proposed ignoring mechanism, which can 1) identify sources examples that are close to the target domain and use them for finetuning; 2) recognize source example having large domain shifts from the target and exclude them from finetuning. In contrast, the baseline methods can only do the black-or-white choice: either using all source examples for finetuning or not using any source example at all, which hence leads to inferior performance.

Table 3 shows the ablation results on ImageCLEF. From this table, we make similar observations as those in Table 2. **First**, in pretraining, the ignoring mechanism which identifies source examples that have large domain shifts from the target and excludes them from the pretraining process achieves better performance than not using ignoring. This is evidenced from the average accuracy results that ablation 7 outperforms ablation 4; our full LBI framework achieves better accuracy than ablation 6. **Second**, during finetuning, ignoring certain source examples works better than using all source examples. This is demonstrated by the average accuracy results that the full LBI outperforms ablation 8; ablation 6 outperforms ablation 5; and ablation 3 outperforms ablation 2.

Table 4 shows the ablation results on OfficeHome. The observations from these results are similar to those made in Table 2 and 3. **First**, performing ignoring on source during pretraining is beneficial, as seen from the average accuracy results that ablation 7 outperforms ablation 4; ablation 8 outperforms ablation 5; and the full LBI outperforms ablation 6. **Second**, performing ignoring on source during finetuning is advantageous than not using ignoring, as demonstrated by the average accuracy results that ablation 3 performs better than ablation 2; ablation 6 outperforms ablation 5; and the full LBI framework outperforms ablation 8.

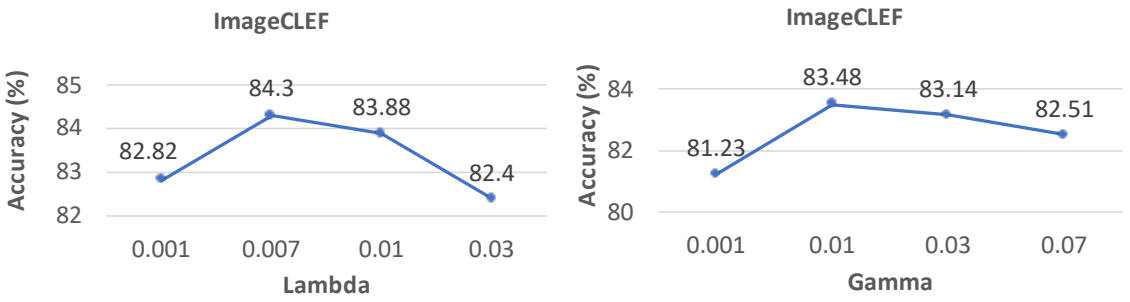

Figure 3: (Left) How accuracy changes as λ increases. (Right) How accuracy changes as γ increases.

Figure 3(Left) shows how the average accuracy varies as the tradeoff parameter λ increases. As can be seen, when λ increases from 0.001 to 0.007, the accuracy increases. This is because a larger λ results in a stronger effect of pretraining. The pretrained encoder captures information of non-ignored source examples and such information is valuable for finetuning. However, if λ continues to increase, the accuracy starts to decrease. The reason is that an excessively large λ leads to too much emphasis on the pretrained encoder and pays less attention to the finetuning on the target data. Figure 3(Right) shows how the average accuracy varies as the tradeoff parameter γ increases. When γ increases from 0.001 to 0.01, the accuracy increases. This is because a larger γ imposes a stronger utilization of non-ignored source examples as additional data for finetuning. However, further increasing γ leads to a worse accuracy. The reason is that if γ is too large, the source data will dominate target data.

5. Conclusions

In this paper, we propose a novel machine learning framework – learning by ignoring (LBI), motivated by the ignoring-driven learning methodology of humans. In LBI, an ignoring variable is learned for each pretraining data example to identify examples that have significant domain difference with the target distribution. We formulate LBI as a three-level optimization problem which consists of three learning stages: an encoder is trained by minimizing a weighted pretraining loss where the loss of each data example is weighted by its ignoring variable; another encoder is finetuned where during the finetuning this encoder is encouraged to be close to the previously pretrained encoder; the ignoring variables are updated by minimizing the validation loss calculated using the finetuned encoder. We conduct experiments on various datasets where the results demonstrate the effectiveness of our method.

References

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM, 2009.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Corbin A Cunningham and Howard E Egeth. Taming the white bear: Initial costs and eventual benefits of distractor inhibition. *Psychological science*, 27(4):476–485, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *2010 ieee 51st annual symposium on foundations of computer science*, pages 329–338. IEEE, 2010.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016a.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018a.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998, 2018b.

Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015a.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105, 2015b.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017a.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017b.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017c.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217, 2017d.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

Henning Müller, Paul Clough, Thomas Deselaers, and Barbara Caputo. *ImageCLEF: experimental evaluation in visual information retrieval*, volume 32. Springer Science & Business Media, 2010.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8050–8058, 2019.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.