Backprop Induced Feature Weighting for Adversarial Domain Adaptation with Iterative Label Distribution Alignment

Thomas Westfechtel[1], Hao-Wei Yeh[1], Qier Meng[1], Yusuke Mukuta[1,2], Tatsuya Harada[1,2]
[1]The University of Tokyo [2]RIKEN
Tokyo, Japan
{thomas,yeh,qiermeng,mukuta,harada}@mi.t.u-tokyo.ac.jp

Abstract

The requirement for large labeled datasets is one of the limiting factors for training accurate deep neural networks. Unsupervised domain adaptation tackles this problem of limited training data by transferring knowledge from one domain, which has many labeled data, to a different domain for which little to no labeled data is available. One common approach is to learn domain-invariant features for example with an adversarial approach. Previous methods often train the domain classifier and label classifier network separately, where both classification networks have little interaction with each other. In this paper, we introduce a classifier-based backprop-induced weighting of the feature space. This approach has two main advantages. Firstly, it lets the domain classifier focus on features that are important for the classification, and, secondly, it couples the classification and adversarial branch more closely. Furthermore, we introduce an iterative label distribution alignment method, that employs results of previous runs to approximate a class-balanced dataloader. We conduct experiments and ablation studies on three benchmarks Office-31, Office-Home, and DomainNet to show the effectiveness of our proposed algorithm.

1. Introduction

One of the major problems of deep learning is that a large amount of labeled training data is required to achieve accurate models. The process to label the training data is tedious and requires a huge amount of manual labor. One branch of research to overcome this problem is unsupervised domain adaptation (UDA). UDA tries to leverage an already existing labeled (source) dataset to an unlabeled (target) dataset that shares some similarities with the source dataset. For example, computer generated pictures of CAD data could be used to recognize objects in the real world. However, since there is a huge difference, also called domain gap, between the source and target data, simply training on the source dataset yields suboptimal results.

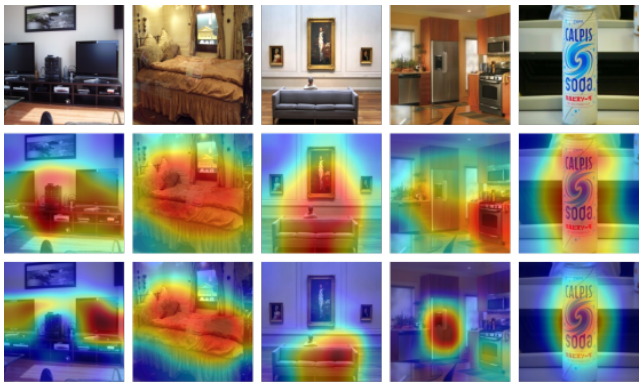


Figure 1: Heatmap of the features that are aligned in the adversarial network. The first row shows the original image. The second row shows the heatmap for DANN [6], while the last row shows the heatmap for our proposed backprop-induced weighting method. The network is adapted on the task Art to RealWorld of the OfficeHome dataset. Without using the weighting, the adversarial network focuses on large parts of the image, including the background. The weighting lets the adversarial network focus mainly on the foreground object.

One way to overcome the domain gap is to train domain invariant features, features that are available both in the source and the target domain. A common approach is to employ adversarial training, where a domain classifier is trained to estimate the specific domain of the samples while the feature extractor is trained to make the extracted features indistinguishable for the domain classifier. However, this approach usually disregards the classifier in the adaptation process. Furthermore, features of the background of the pictures, that have no influence on the classification also get adapted, which may result in lower accuracy. To overcome this problem, we propose a backprop-induced weighting of the feature space. We backpropagate the classifica-

tion loss back to the feature space and create a weighting vector based on the backpropagated gradients. The benefits of this approach are two-fold. Firstly, as shown in [21] the gradients can be seen as a class attention score for different features, meaning that important features for the classification are emphasized for the adversarial adaptation, while less important features (i.e. the background) are attenuated. Secondly, through the backpropagation, the classifier and domain discriminator are implicitly coupled. In Fig. 1 it can be seen that the backprop-induced weighting emphasizes the foreground objects while attenuating the background.

Another problem for domain adaptation is the label distribution shift (LDS). While a source dataset may be created in an equally distributed way, some classes may be overrepresented or underrepresented in the target dataset. This misalignment leads to adaptation errors early on, which can accumulate over the training. A balanced dataloader that loads training data for each class equally would mitigate this problem, but since the labels for the target data are not known, this is not feasible. One possible solution is to estimate pseudo-labels for the target dataloader during the training process and employ them to balance the dataloader. However, since the pseudo-labels are noisy especially in the beginning of the training, this strategy may lead to an accumulation of early errors. In contrast, we propose an iterative adaptation paradigm. We train the network from scratch for several runs, where each new run uses the predicted labels from the previous runs to achieve a more balanced dataloader (with the first run using a random dataloader). The main benefit of this approach is that relatively accurate pseudo-labels are available even during the beginning of the training of a run.

Our main contributions are:

- We introduce a backpropagation-guided weighting of the feature space. This lets us emphasize the important features for classification, and it also couples the classifier to the adversarial domain adaptation process more closely.

- We introduce an iterative label shift alignment paradigm to approximate a balanced target dataloader. The iterative paradigm allows us to circumvent the accumulation of early misalignment errors by providing reliable pseudo-labels even in the beginning of the training of a run.

- We show the effectiveness of our algorithm on three datasets (Office-31, Office-Home, and DomainNet) and further evaluate the contribution of the different methods in ablation studies.

2. Related Work

A common approach for unsupervised domain adaptation (UDA) is to extract domain invariant features, features that are shared across the domains. This can be achieved with an adversarial approach. Domain-adversarial neural network (DANN) [6] employs a domain classifier to distinguish whether the image belongs to the source or target domain. By employing a gradient reversal layer the feature extractor is trained to extract features that are indistinguishable for the domain classifier, thus extracting domain invariant features. Conditional domain adversarial network (CDAN) [16] extends this method by multilinear conditioning the domain classifier with the classifier predictions. Many adversarial methods build up on DANN or CDAN. [18] introduces a spectral adaptation to CDAN. [8] adds group- and class-wise domain classifiers to DANN and synchronizes the gradient between the different domain classifiers. Moving semantic transfer network [26] extends DANN with a moving semantic loss. The method creates class representations for both domains and each class in the feature space, which are updated with each sample during the training process. The distance between the source and target feature representation of a class is used as domain adaptation loss.
Our method builds up on DANN. Furthermore, we also use the moving semantic loss of [26] in our paper, however, inspired by current clustering methods [10] [23] we further add a loss to penalize similar representation for different classes.

Recent research has included attention mechanism of neural networks to the domain adaptation task. [21] has shown for general purposes that the gradients of the classifier can be used to visualize the attention of the network with regard to the classification output. For domain adaptation, [13] proposed a domain conditioned channel attention mechanism. Probably most similar to our proposed backprop-induced feature weighting is [11]. In their work, the authors focused on identifying regions that can be adopted better. In particular, they backpropagate the domain predictive uncertainty to emphasize well-adopted regions for the classifier.
In contrast to that, in our work we backpropagate the classification error to emphasize features that are important for the classification while attenuating irrelevant features. This lets the adversarial branch focus more on the relevant part of the image.

Besides the domain gap, another problem for the adaptation process is that the label distribution of the source and target dataset can be vastly different. [9] presented a sampling-based implicit alignment approach. [25] introduced a balance factor to mitigate the difference in the label distribution. [28] introduced class-specific auxiliary weights to overcome the varying class prior probability.

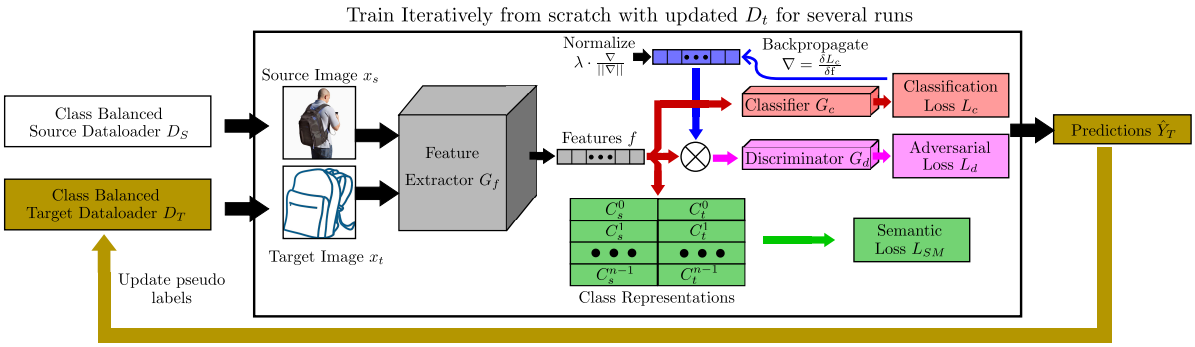


Figure 2: The pipeline of our proposed method. We employ three losses to train the network. For the adversarial loss, we weigh the feature space based on the importance for the classifier. In particular, we backpropagate the classification loss to the feature layer, normalize the gradients and employ it as a weighting vector. Furthermore, after training the network for a single run, the predicted labels of the target domain are used to initialize dataloader of the next run to achieve a class-balanced dataloader.

[29] proposed a weighting mechanism based on the domain similarity and the prediction uncertainty of each sample. However tackling the label and domain shift problem simultaneously can lead to an error build-up from early misalignment, especially since it's hard to robustly estimate target labels early on in the training. To circumvent this error build-up, we chose an iterative process where the network is trained for several consecutive runs from scratch. Each new run employs the pseudo-labels of the previous runs. This means that the dataloader has access to reliable pseudo-labels even in the early training stages of a run.

3. Methodology

In unsupervised domain adaptation, the task is to mitigate the domain shift between a source and target domain. For the source domain $\mathcal{D}_s$ a set of n_s labeled samples $\mathcal{D}_s = (x_{i,s}, y_{i,s})_{i=1}^{n_s}$ is given, where $x_{i,s}$ donates a sample with the corresponding label $y_{i,s}$. For the target domain $\mathcal{D}_t$ only the samples are given without any labels $\mathcal{D}_t = (x_{i,t})_{i=1}^{n_t}$. The goal is to estimate the labels for the target domain $y_{i,t}$ by exploiting the shared feature space that is similar, but different. In our work, we tackle the vanilla or closed-set setting, where the source and target domain have identical label classes $\mathcal{C}_s = \mathcal{C}_t$.

In our work, we employ a combination of three losses, the classification loss for samples of the source domain L_c, an adversarial loss L_d, and a moving semantic loss L_{MS}. For the adversarial loss, we combine the adversarial network introduced in DANN [6] with our proposed backprop-induced weighting. The moving semantic loss is based on MSTN [26], however, we expand it with an inter-class loss that enlarges the distance between different classes.

Furthermore, we introduce an iterative strategy to tackle the label distribution shift. In particular, the network is trained from scratch for several runs, where each run employs the predictions for the target data of the previous runs in order to achieve a class-balanced dataloader. The pipeline of our method can be seen in Fig. 2. Furthermore, the pseudo-algorithm for our method can be seen in Alg. 1.

3.1. Backpropagation induced importance weighting

Our network consists of a feature extractor G_f that maps an input x_i into features space f_i

$$f_i = G_f(x_i) \tag{1}$$

From the feature space f_i, the network splits into two branches. The first branch, the classification branch G_c, estimates the class label probability $p(y_i)$ for the given features f_i.

$$p(y_i) = G_c(f_i) \tag{2}$$

For the second branch, we employ a domain classifier network G_d introduced by DANN [6]. G_d aims to distinguish the respective domain based on the feature space. The loss is calculated as the binary cross entropy loss between the prediction and the ground truth:

$$L_d = L_{BCE}(G_d(f_i), d_i) \tag{3}$$

In the backpropagation step, the domain loss is inverted in a gradient reversal layer, so that the feature extractor G_f is trained to extract features that are indistinguishable for the domain classifier, thus extracting domain-invariant features. Our approach targets the input of this branch. We argue that not all features within the feature space f_i are equally important for the classification task. Therefore we propose to weigh the feature space based on its importance for the classifier. In this way, the adversarial domain classifier network prioritizes adapting features that are of importance for the classification. To estimate the importance of the features for the classification task, we first calculate the classifier loss L_c for the current sample.

$$L_c = L_{CE}(p(y_i), \hat{y}_i) \tag{4}$$

Where L_{CE} depicts the cross entropy loss and $\hat{y}_i$ depicts the ground truth label in case of the source domain and a pseudo label in case of the target domain. For the pseudo labels of the target domain, we simply select the class with the highest probability based on the classification score of the classifier.

$$\hat{y}_i = \text{argmax}(p(y_i)) \tag{5}$$

In the next step, the loss is backpropagated to the feature layer. And the gradients ∇ for the layer are estimated.

$$\nabla_i = \frac{\partial L_c}{\partial f_i} \tag{6}$$

We normalize the gradients and use them as an importance weighting for the features space. To approximately preserve the norm of the feature space, we introduce a scaling factor λ.

$$f_i' = \lambda \cdot f_i \circ \frac{\nabla_i}{||\nabla_i||} \tag{7}$$

The main point of our proposed method is, that instead of aligning the distributions based on the feature space f_i generated by the feature extractor G_f, we align the weighted feature space f_i'. The domain adaptation loss L_d is calculated based on f_i' instead of f_i.

$$L_d = L_{BCE}(G_d(f_i'), d_i)) \tag{8}$$

3.2. MSTN

For the third loss of our method, we use a moving semantic transfer loss L_{MS}. This loss is based on MSTN [26]:

$$L_{MSTN} = \sum_{k=1}^{K} \Phi(C_s^k, C_t^k) \tag{9}$$

where C_s^k and C_t^k are the moving centroids of the classes in feature space for source and target data respectively. Φ is a distance measure. L_{MSTN} aligns the class representations of source and target data within the feature space. For the calculation of the centroids, we do not use the importance weighted features space. Inspired by current deep-clustering-based methods [10] [23] we extend the loss to also enlarge the distance between centroids of different class

$$L_{MS} = \sum_{k=1}^{K} \Phi(C_s^k, C_t^k) + \sum_{k=1}^{K} \sum_{j \neq k}^{K} \Theta(C_s^k, C_s^j) + \Theta(C_s^k, C_t^j) + \Theta(C_t^k, C_t^j) \tag{10}$$

The cosine similarity between the centroids is used as function Θ and the L2-distance is used for Φ.

Algorithm 1: Algorithm of our method.

Input: Source and target dataset: $\mathcal{D}_s, \mathcal{D}_t$

/* Iteratively train network from scratch for r_{max} runs */

for $r_i = 0; r_i < r_{max}$ **do**

- Initialize class-balanced dataloader D_s and D_t
- Initialize network
- /* Train network */
- **for** $j = 0; j < j_{max}$ **do**
  - /* Load samples and insert in network */
  - Get samples $x_s, y_s, x_t, p(\hat{y}_{t,a})$
  - $f_s, p(y_s), f_t, p(y_t) \leftarrow G_c(G_f(x))$
  - /* Aggregate and bootstrap target probability */
  - $p(y_{t,a}) = (r-1)p(y_{t,a}) + r \cdot p(y_t)$
  - $p(y_{t,b}) = u(n)p(y_{t,a}) + (1-u(n))p(\hat{y}_{t,a})$
  - $\tilde{y}_t = \text{argmax}(p(y_{t,b}))$
  - /* Moving semantic loss */
  - Update moving semantics
  - $C_{s,t}^k \leftarrow f_s, y_s, f_t, \tilde{y}_t$
  - Calculate semantic loss L_{MS}
  - /* Adversarial loss */
  - Calculate $L_{c,s} = L_{CE}(p(y_s), y_s)$
  - Calculate $L_{c,t} = L_{CE}(p(y_t), \tilde{y}_t)$
  - Backpropagate $L_{c,s}$ and $L_{c,t}$
  - Normalize gradients and weight features
  - $f_{s,t}' = \lambda \cdot f_{s,t} \circ \frac{\nabla_{s,t}}{||\nabla_{s,t}||}$
  - Calculate adversarial loss L_d
  - $L_d = L_{BC}(G_d((f_{s,t}'), d_{s,t})$
  - /* Total loss */
  - $L = L_{c,s} + \sigma(i) \cdot L_d + \sigma(i) \cdot L_{MS}$
  - Backpropagate L and update
- /* Update predictions for dataloader */
- Calculate predictions for target data
- $\hat{p}(y_T) \leftarrow G_c(G_f(x_T))$
- $p(\hat{y}_{T,a}) = (q-1) \cdot p(\hat{y}_{T,a}) + q \cdot \hat{p}(y_T)$

3.3. Iterative label distribution shift alignment

In order to tackle the label distribution shift, we propose an iterative process, where we train our network for several runs from scratch. Each run uses the results of the previous runs to initialize a class-balanced dataloader. As we do not have any label estimates for the target dataset during the first run, the first run employs a random dataloader. This means that the dataloader loads data of the target dataset randomly regardless of their respective class. In consecutive runs, the class-balanced dataloader loads iteratively one sample for each class in random order. The classes are de-

Table 1: Accuracy results on Office-31 dataset. The best results are displayed in bold and the runner-up results are underlined. BIWAA displays the results of our method after the first run and BIWAA-I the results after 10 runs.

Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg
ResNet-50 [7]	68.4	96.7	99.3	68.9	62.5	60.7	76.1
DAN [15]	68.5	96.0	99.0	67.0	54.0	53.1	72.9
DANN [6]	82.0	96.9	<u>99.1</u>	79.7	68.2	67.4	82.2
MCD [20]	88.6	98.5	**100.**	92.2	69.5	69.7	86.5
MSTN [26]	91.3	98.9	**100.**	90.4	72.7	65.6	86.5
CDAN+E [16]	94.1	98.6	**100.**	92.9	71.0	69.3	87.7
MDD [30]	94.5	98.4	**100.**	93.5	74.6	72.2	88.9
GVB-GD [5]	94.8	98.7	**100.**	<u>95.0</u>	73.4	73.7	89.3
DCAN [13]	95.0	97.5	**100.**	92.6	77.2	74.9	89.5
GSDA [8]	**95.7**	**99.1**	**100.**	94.8	73.5	74.9	89.7
ASAN [18]	95.6	98.8	**100.**	94.4	74.7	74.0	90.0
BIWAA	94.8	<u>99.0</u>	**100.**	94.6	<u>74.8</u>	<u>75.0</u>	89.7
BIWAA-I	<u>95.6</u>	<u>99.0</u>	**100.**	**95.4**	**75.9**	**77.3**	**90.5**

termined based on the predictions of the previous runs.

3.4. Bootstrap and aggregating label

To further improve the accuracy we also employ an aggregating label strategy and bootstrap the training. Aggregating labels means that not only the current prediction is used to estimate the pseudo-label, but also previous predictions of the same sample. Each time the label probability for a sample i is estimated $p(y_i)$ we update an aggregate version $p(y_{i,a})$ of it. This increases the reliability of the prediction for the target data. The strategy is used within a single run, but also between runs to update the predictions for the dataloader D_T. Within a single run, the aggregate label is updated by a factor r each time the respective target data is loaded:

$$p(y_{i,a}) = (r-1) \cdot p(y_{i,a}) + r \cdot p(y_i) \tag{11}$$

Similarly, the predictions for the dataloader D_T are updated by a factor q after each run:

$$p(\hat{y}_{i,a}) = (q-1) \cdot p(\hat{y}_{i,a}) + q \cdot p(\hat{y}_i) \tag{12}$$

Furthermore, we use the predictions of the previous runs to bootstrap the training. While the predictions gained from previous runs are noisy, they still provide relatively accurate labels, particularly compared to label estimates during the early stages of the training. Therefore, we exploit the predictions of previous runs to bootstrap the training. However, overly relying on the label predictions of the previous runs would limit the possible progress of the current training. Therefore the bootstrap strategy is only used in the early stages of the training and fades out in later stages. In particular, we employ an exponential function combines the labels prediction from the previous runs $p(\hat{y}_{i,a})$ with the label prediction of the aggregate label $p(y_{i,a})$ of the current run to a bootstrapped version $p(y_{i,b})$.

$$p(y_{i,b}) = u(j) \cdot p(y_{i,a}) + (1 - u(j)) \cdot p(\hat{y}_{i,a}) \tag{13}$$

where j is the iteration of the current run, and u is the exponential function, which is as follows:

$$u(j) = \begin{cases} 2/(1 + e^{-10 \frac{j}{j_t}}) - 1 & \text{if } j \leq j_t \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

j_t depicts the number of iterations with the bootstrap function active. The bootstrapped prediction $p(y_{i,b})$ is used to estimate the pseudo label for the backpropagation-induced weighting and for updating the class representations.

4. Experiments

We evaluate our proposed method on three different domain adaptation benchmarks, Office-31, Office-Home, and DomainNet. We show that we can improve the baselines significantly. In ablation studies, we further investigate the contribution of the different parts of our proposed algorithm.

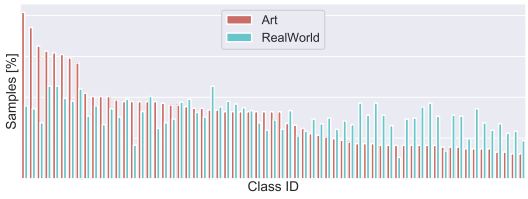
4.1. Setups

Office-31 [19] is the most popular dataset for real-world domain adaptation. It contains 4,110 images of 31 categories. The domains are Amazon (A), Webcam (W), and DSLR (D). We evaluate all six possible adaptation tasks.

Office-Home [24] is a more challenging benchmark than Office-31. It contains 15,500 images of 65 categories. The domains are Art (A), Clipart (C), Product (P), and Real-World (R). We evaluate all twelve possible adaptation tasks.

Table 2: Accuracy results on Office-Home dataset. The best results are displayed in bold and the runner-up results are underlined. BIWAA displays the results of our method after the first run and BIWAA-I the results after 10 runs.

Method	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg
ResNet-50 [7]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN [15]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN [6]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
MCD [20]	48.9	68.3	74.6	61.3	67.6	68.8	57.0	47.1	75.1	69.1	52.2	79.6	64.1
MSTN [26]	49.8	70.3	76.3	60.4	68.5	69.6	61.4	48.9	75.7	70.9	55.0	81.1	65.7
CDAN+E [16]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
MDD [30]	54.9	73.7	77.8	60.0	71.4	71.8	61.2	53.6	78.1	72.5	60.2	82.3	68.1
ASAN [18]	53.6	73.0	77.0	62.1	73.9	72.6	61.6	52.8	79.8	73.3	60.2	83.6	68.6
GSDA [8]	**61.3**	76.1	79.4	65.4	73.3	74.3	65.0	53.2	80.0	72.2	**60.6**	83.1	70.3
GVB-GD [5]	57	74.7	79.8	64.6	74.1	74.6	65.2	55.1	81.0	74.6	59.7	**84.3**	70.4
DCAN [13]	54.5	75.7	**81.2**	67.4	74.0	**76.3**	67.4	52.7	80.6	74.1	59.1	83.5	70.5
BIWAA	54.2	76.3	79.5	66.2	73.1	74.1	66.1	54.2	80.1	74.4	58.4	83.2	70.0
BIWAA-I	56.3	**78.4**	**81.2**	**68.0**	**74.5**	75.7	**67.9**	**56.1**	**81.2**	**75.2**	60.1	83.8	**71.5**



(a) Label histogram for Office-Home: Art→Realworld

(b) Label histogram for DomainNet: Clipart→Painting

Figure 3: Example of label distribution for different datasets.

In Fig. 3a the label distribution for the domains Art and Real-World can be seen.

DomainNet [17] is a large scale dataset with about 600.000 images from 6 different domains and 345 different classes. However, as some domains and classes have a considerable amount of mislabeled data, we follow [22] and only use a subset of 40 commonly seen classes from the four domains of Real World (R), Clipart (C), Painting (P), and Sketch (S). We evaluate all twelve possible adaptation tasks. DomainNet consists of the largest label distribution shift of the three datasets. An example for the domains Clipart and Painting can be seen in Fig. 3b.

Implementation details: We built up our implementation on the DANN implementation of [16]. We use the ResNet-50 [7] architecture as backbone for all of our experiments. Same as in [16] we increase the learning rate by a factor of 10 for all layers that are trained from scratch. We further adopt the learning rate annealing strategy and the progressive discriminator learning strategy $\sigma(i)$. For the aggregate label functions we use $t = 0.8$ and $s = 0.9$, we employ the bootstrap function for the first 500 iterations: $n_t = 500$. A normalization factor of $\lambda = 10$ was employed for all experiments.

4.2. Results

Results for Office-31: The results are shown in Tab. 1. Our method outperforms all compared methods in most tasks. Only in the case of D→W and A→W GSDA performs slightly better than our method. The overall average accuracy is 0.5%pts higher than the runner-up method. Even without using the iterative label shift alignment, our method already achieves state-of-the-art results with the same average accuracy as GSDA and is only being outperformed by ASAN.

Results for Office-Home: As shown in Tab. 2 our algorithm achieves higher average accuracy than all other methods, performing 1.0%pts better than the runner-up method. We achieve in 8 out of 12 tasks the best highest accuracy, and for 3 tasks the second highest accuracy.

Results for DomainNet: In Tab. 3 the results for DomainNet can be seen. Even without the iterative training strategy, our method already outperforms the other methods. Using the iterative strategy our method improves to an average score of 79.88%, just below 80%. Our method outperforms other methods in most of the tasks, only for the tasks R→C, C→R and P→R InstaPBM [12] performs slightly better. However, overall we achieve about 2%pts increase in the average score.

Table 3: Per class accuracy results on DomainNet dataset. The best results are displayed in bold and the runner-up results are underlined. BIWAA displays the results of our method after the first run and BIWAA-I the results after 10 runs.

Method	R→C	R→P	R→S	C→R	C→P	C→S	P→R	P→C	P→S	S→R	S→C	S→P	Avg
ResNet-50 [7]	58.84	67.89	53.08	76.70	53.55	53.06	84.39	55.55	60.19	74.62	54.60	57.78	62.52
BBSE [14]	55.38	63.62	47.44	64.58	42.18	42.36	81.55	49.04	54.10	68.54	48.19	46.07	55.25
PADA [1]	65.91	67.13	58.43	74.69	53.09	52.86	79.84	59.33	57.87	76.52	66.97	61.08	64.48
MCD [20]	61.97	69.33	56.26	79.78	56.61	53.66	83.38	58.31	60.98	81.74	56.27	66.78	65.42
DAN [15]	64.36	70.65	58.44	79.44	56.78	60.05	84.56	61.62	62.21	79.69	65.01	62.04	67.07
ETN [2]	69.22	72.14	63.63	86.54	65.33	63.34	85.04	65.69	68.78	84.93	72.17	68.99	73.99
BSP [4]	67.29	73.47	69.31	86.50	67.52	70.90	86.83	70.33	68.75	84.34	72.40	71.47	74.09
DANN [6]	63.37	73.56	72.63	86.47	65.73	70.58	86.94	73.19	70.15	85.73	75.16	70.04	74.46
COAL [22]	73.85	75.37	70.50	89.63	69.98	71.29	89.81	68.01	70.49	87.97	73.21	70.53	75.89
InstaPBM [12]	**80.10**	**75.87**	70.84	**89.67**	70.21	72.76	**89.60**	74.41	72.19	87.00	79.66	71.75	77.84
BIWAA	77.17	73.20	73.01	88.06	70.73	72.84	88.38	77.31	74.67	87.01	**80.57**	71.94	77.91
BIWAA-I	79.93	75.24	**75.35**	87.93	**72.07**	**75.71**	88.87	**77.81**	**76.66**	**88.78**	80.49	**74.49**	**79.44**

Table 4: Effect of backprop-induced feature weighting. The average accuracy over all tasks is displayed for the Office-31 (O31) and the Office-Home (OH) dataset. BG stands for the proposed backprop-induced weighting.

Method	O31	OH
AFN [27]	85.7	67.3
AFN+BG	87.5 (+1.8)	69.6 (+2.3)
DANN [6]	82.2	57.6
DANN+BG	87.7 (+5.5)	67.3 (+9.7)
CDAN+E [16]	87.7	65.8

5. Ablation studies

Effect of backprop induced weighting:
In this part, we examine the effect of the backprop-induced weighting. For the evaluation, we use the original code of the respective methods and only introduce the backprop-induced weighting to it. We applied the modification to DANN [6] and AFN [27]. AFN is based on the observation that the feature norms of the target domain are much smaller than for the source domain. The algorithm progressively adapts the feature norms of the two domains.

The results can be seen in Tab. 4. In the case of AFN, the weighting improves the accuracy by 1.8%pts for the Office-31 and 2.3%pts for the Office-Home dataset. In the case of DANN, the results are improved by 5.5%pts and 9.7%pts respectively. We would like to further point out that our method achieves similar results as CDAN+E [16] on Office-31 and outperforms it on Office-Home by 1.5%pts. As many methods build up on CDAN+E we think it worth noticing that our method performs better. Comparing DANN, CDAN, and our method to each other, DANN tries to match the distribution of f_i which ignores discriminative information of the classifier. CDAN tries to match the distribution of $(f_i \otimes p_i)$ which is discriminative but requires many dimensions. In contrast our proposed method tries to match the distribution of $\lambda \cdot f_i \circ \frac{\nabla_i}{||\nabla_i||}$, with $\nabla_i = \frac{\partial L_c}{\partial f_i}$, which is both discriminative and compact.

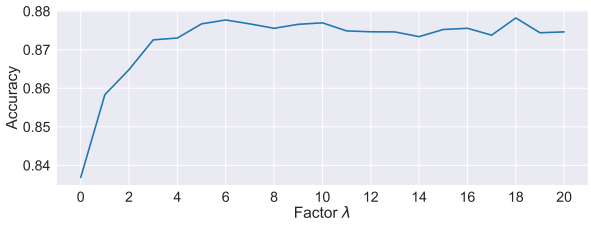
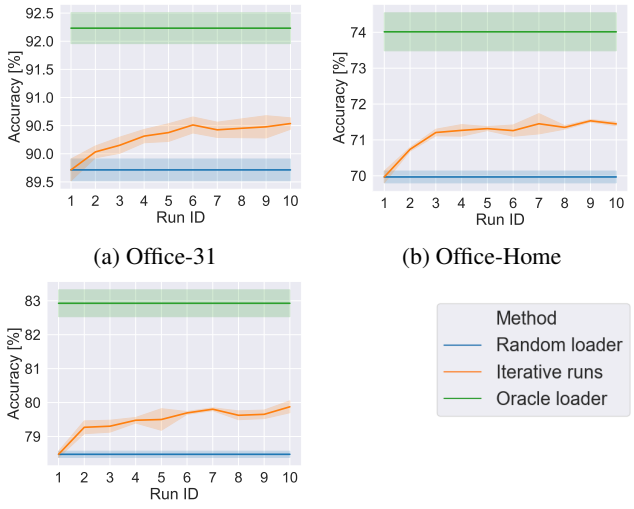


Figure 4: Influence of the normalization factor λ on the accuracy for the Office-31 dataset. We ran the experiment for three different seeds for all factors.

Influence of normalization factor λ:
In this part, we investigated the influence of the normalization factor λ on the adaptation task. We only employed the backprop-induced weighting strategy and did not use the semantic loss L_{MS} nor other optimizations. We ran 3 experiments for each λ in a range between 0 and 20, where 0 means that the backprop-induced weighting was not used. The results are plotted in Fig. 4. It can be seen that the accuracy improves steeply until a factor of about $\lambda = 6$, and is relatively stable for higher values. In our experiments, a value of $\lambda = 10$ is used for all of our experiments.

Accuracy over runs:
In this part, we show how the accuracy increases over iterative runs. Fig. 5 displays the average accuracy for the three datasets over 10 iterative runs. The lower bound in the figure employs a random data loader (same as the first run of our method), and the upper bound employs an oracle



(a) Office-31 (b) Office-Home

(c) DomainNet

Figure 5: Accuracy over consecutive runs on the three datasets. The random dataloader serves as baseline, while the oracle version serves as upper bound. We ran our experiment for 10 iterative runs each time with three different random seeds. The average accuracy and the standard deviation are plotted.

data loader that employs the ground truth data to load data class-balanced. It should be noted that the oracle dataloader does not use the ground truth data during the estimation of the pseudo labels within the main code. It can be seen that during the first few runs the accuracy increases significantly and slowly for further runs. Currently, our method improves the baseline of a random dataloader by about 0.8%pts for Office-31, 1.5%pts for Office-Home, and 1.41%pts for DomainNet. It can be clearly seen that the oracle dataloader achieves higher accuracies, meaning there is still a large potential for further improvements. So far our algorithm achieves about $1/3$ of the potential improvement of the oracle dataloader. It has to be stated that the problem of overcoming a domain shift and label shift at the same time is a difficult problem and it is therefore unclear how close it is possible to come close to the results of the oracle version.

Bootstrap and label aggregation:
In this part, we examine the contribution of the different optimization strategies. Tab. 5 shows the contribution of the bootstrap and label aggregation strategies. The bootstrap strategy has arguably the highest impact, improving the accuracy by about 0.25%pts on its own. Using all three strategies achieves the best results with an increase of 0.31%pts over not using any of these strategies.

6. Discussion and Limitations

In this work, we introduced a backprop-induced feature weighting. We combined the method with a moving semantic transfer loss, that aligns the semantics for the same class while disentangling semantics between different classes. We further introduced an iterative label distribution alignment paradigm. Our results show that we outperform the baselines and achieve top results on three different datasets, namely Office-31, Office-Home, and DomainNet.

Table 5: Effect of the bootstrap and label aggregation strategy. We report the average score over all tasks of the Office-31 dataset. For each configuration, three seeds are trained for 10 consecutive runs. A_r depicts whether the aggregation is active during a single training, A_i the aggregation for iterative runs, and B bootstrap strategy.

A_r	A_i	B	Accuracy
			90.23
✓			90.31
	✓		90.30
✓	✓		90.30
		✓	90.48
✓		✓	90.49
	✓	✓	90.38
✓	✓	✓	90.54

The backprop-induced feature weighting allows the adaptation network to focus on the important foreground objects. However, this process makes use of pseudo-labels which are inherently noisy. While we presented some strategies (aggregate labels and bootstrap) to overcome this problem, it would be interesting to see how an approach [3] that progressively extends the target data from easy to hard samples would influence the algorithm.

We showed that our iterative label distribution alignment strategy increases performance significantly. However, there is still a large potential for improvement when compared to the oracle data-loader. We believe that this potential gives a valuable hint on where domain adaptation methods can still be improved.

The advantage of our iterative label distribution alignment strategy is that it can rely on relatively robust pseudo-labels even early in the training. However, this comes at a computational cost as the network is trained for several runs from scratch. One future direction of study is to overcome the domain shift and label distribution shift simultaneously.

Acknowledgment

This work was partially supported by JST AIP Acceleration Research JPMJCR20U3, Moonshot R&D Grant Number JPMJPS2011, CREST Grant Number JPMJCR2015, JSPS KAKENHI Grant Number JP19H01115, and JP21K17799 and Basic Research Grant (Super AI) of Institute for AI and Beyond of the University of Tokyo.

References

[1] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[2] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang. Learning to transfer examples for partial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2985–2994, 2019.

[3] Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 627–636, 2019.

[4] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International conference on machine learning*, pages 1081–1090. PMLR, 2019.

[5] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Chi Su, Qingming Huang, and Qi Tian. Gradually vanishing bridge for adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12455–12464, 2020.

[6] 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, 2016.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Lanqing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. Unsupervised domain adaptation with hierarchical gradient synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4043–4052, 2020.

[9] Xiang Jiang, Qicheng Lao, Stan Matwin, and Mohammad Havaei. Implicit class-conditioned domain alignment for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 4816–4827. PMLR, 2020.

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

[11] Vinod Kumar Kurmi, Shanu Kumar, and Vinay P Namboodiri. Attending to discriminative certainty for domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 491–500, 2019.

[12] Bo Li, Yezhen Wang, Tong Che, Shanghang Zhang, Sicheng Zhao, Pengfei Xu, Wei Zhou, Yoshua Bengio, and Kurt Keutzer. Rethinking distributional matching based domain adaptation. *arXiv preprint arXiv:2006.13352*, 2020.

[13] Shuang Li, Chi Liu, Qiuxia Lin, Binhui Xie, Zhengming Ding, Gao Huang, and Jian Tang. Domain conditioned adaptation network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11386–11393, 2020.

[14] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[15] 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, 2015.

[16] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1647–1657, 2018.

[17] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[18] Christoph Raab, Philipp Vath, Peter Meier, and Frank-Michael Schleif. Bridging adversarial and statistical domain transfer via spectral adaptation networks. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[19] 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.

[20] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3723–3732, 2018.

[21] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[22] Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: an empirical odyssey. In *European Conference on Computer Vision*, pages 585–602. Springer, 2020.

[23] Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8725–8735, 2020.

[24] 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.

[25] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer

learning. In *2017 IEEE international conference on data mining (ICDM)*, pages 1129–1134. IEEE, 2017.

[26] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pages 5423–5432. PMLR, 2018.

[27] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1426–1435, 2019.

[28] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2272–2281, 2017.

[29] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2720–2729, 2019.

[30] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019.