Improving Domain Adaptation By Source Selection
Kevin Bascol, Rémi Emonet, Elisa Fromont

To cite this version:
Kevin Bascol, Rémi Emonet, Elisa Fromont. Improving Domain Adaptation By Source Selection. 2019. hal-02136378

HAL Id: hal-02136378
https://hal.archives-ouvertes.fr/hal-02136378
Submitted on 22 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
IMPROVING DOMAIN ADAPTATION BY SOURCE SELECTION

Kevin Bascol†‡, Rémi Emonet†, Élisa Fromont⋆

† Laboratoire Hubert Curien UMR 5516, Univ Lyon, UJM-Saint-Etienne F-42023, Saint-Etienne, France
⋆IRISA/INRIA rba, Uni Rennes 35042 Rennes cedex, France
‡Bluecime inc. 38330 Montbonnot Saint-Martin, France

ABSTRACT
Domain adaptation consists in learning from a source data distribution a model that will be used on a different target data distribution. The domain adaptation procedure is usually unsuccessful if the source domain is too different from the target one. In this paper, we study domain adaptation for image classification with deep learning in the context of multiple available source domains. We propose a multisource domain adaptation method that selects and weights the sources based on inter-domain distances. We provide encouraging results on both classical benchmarks and a new real world application with 21 domains.

Index Terms— Domain Adaptation, Negative Transfer, Deep Learning, Image Classification.

1. INTRODUCTION AND RELATED WORK
Domain adaptation [1] consists in learning from a (labeled) source data distribution, a model that will be used on a different (but related and often unlabeled) target data distribution. Many real world tasks require the use of domain adaptation simply because of a lack of (target) labeled data or because of some shift between the source and the target data distribution that prevents from successfully using the learned model on the target data. When using deep learning, the most common domain adaptation algorithmic setting is to construct a common representation space for the two domains while keeping good performance on the source labeling task. This can be achieved through the use of adversarial techniques where feature representations from samples in different domains are encouraged to be indistinguishable [2], [3]. Whatever the technique, the domain adaptation procedure is usually unsuccessful if the source domain is too different from the target one. In [4] for example, the authors have empirically identified positive and negative transfer situations. We study domain adaptation for image classification with deep learning in the context of multiple available source domains and when no label are available on the target domain.

Notations We consider having $D$ source domains. Data of the $i$th domain are noted $Z_i$ and are composed of examples $X_i$ and labels $Y_i$. The target domain is given the index $j$.

A number of related works propose to select or weight (elements of) the source domains in order to improve the test accuracy on the target domain but none of these works explicitly evaluate and propose solutions to overcome the effect of negative transfer during the adaptation process. For example, the work from [5] considers transfer learning from only one source domain and when the target task is a sub-task of the source task (as for us, no target label is available). They also extend the work of [2] but they decompose the domain classifier according to each source classes. During the adaptation phase, each target example is weighted according to the class-domain classifier loss. The works from [6], [7], [8] and [9] tackle the problem of multi-source domain adaptation but their selection scheme makes use of a few target labels and is used to select one single source domain. The unpublished work from [10] is the closest to ours. The authors propose to select multiple domains according to four possible distances (the $\chi^2$-divergence, the Maximum Mean Discrepancy, the Wassertein distance and the Kullback-Liebler divergence) and according to the classification performance on each single source domain. Both the distance and the performance features are weighted by a parameter $\beta$ computed as:

$$\beta = \arg \min_{\beta \geq 0} \sum_{i=1}^{D} \sum_{k=1;k \neq i}^{D} |\xi(Z_i, Z_k) - \beta f(Z_i, Z_k)|$$

with $f$ the set of considered features and $\xi(Z_i, Z_k)$ the performance of the classifier trained on $Z_i$ and tested on $Z_k$. The authors show that on a homogeneous dataset, their method is better than randomly selecting the domains but not better than when using all of them. However, on a heterogeneous dataset, selecting the sources with their proposed distance is better than both selecting all the domains and selecting them randomly. Note that to optimize $\beta$, $D$ classifiers should be trained which can be costly in practice (especially for deep neural networks). Besides, the authors do not provide any criterion to set the number of selected sources.

In Section 2, we show our strategy to automatically select the best sources to avoid negative transfer during do-
main adaptation. Section 3 shows extensive experiments and promising results on both classical benchmarks and a new real world application related to ski-resort chairlift security. We conclude in Section 4.

2. DOMAIN SELECTION AND WEIGHTING

Considering a target domain \( j \) and \( D \) source domains, we propose an approach that automatically computes a weight vector \( p^j \in \Delta^{D-1} \subset \mathbb{R}^D \) (probability simplex) and uses \( p^j \) to reweight the domains (when sampling minibatches) during “domain-adversarial training [2]”. This training phase is usually done to fine-tune a pre-trained network. The proposed approach is modular as we decompose the computation of the domains weight vector \( p^j \) in three configurable steps:

1. the distance vector \( d^j = \{d^j_i\}_{i=1}^D \) is computed (distance of each source domain, \( i \), to the target one, \( j \)),
2. it is mapped to a score vector \( s^j = \text{score}(d^j) \),
3. it is normalized to a probability vector \( p^j = s^j / \sum_i s^j_i \).

Computing pairwise dataset distances We focus here on a distance based on optimal-transport but other distances could be considered. For instance, one could use the average minimal Euclidean distance (\( d^j_i = \text{average}_{x \in X_i} \min_{y \in X_j} \|x - y\| \)) or a distance based on auto-encoder where an autoencoder \( AE_j \) is trained with the points of domain \( j \) and \( d^j_i = \text{average}_{x \in X_i} \|x - AE_j(x)\|^2 \) (average squared reconstruction error).

The optimal transport problem aims at finding the minimal cost for transforming a data distribution into another one [11]. This minimal cost is defined as the sum of the (probability) mass to displace multiplied by the given displacement price (for instance the euclidean distance). The minimal cost is called the Wasserstein distance and constitutes a distance between distribution. In our discrete case (samples of points), the distance can be expressed as:

\[
d^j_i = \sum_{x \in X_j} \sum_{y \in X_i} \gamma^*_{x,y} C_{x,y} \tag{2}
\]

where \( C \) is a distance matrix between all pair of elements of the two domains. The optimal transport plan \( \gamma^* \) is obtained by solving the optimal transport problem:

\[
\gamma^* = \arg\min_{\gamma \in \Pi(\hat{\mu}_j, \hat{\mu}_i)} \langle \gamma, C \rangle_F \tag{3}
\]

where \( F \) is the Frobenius inner product and the constraint set \( \Pi(\hat{\mu}_j, \hat{\mu}_i) = \{\gamma \in \mathbb{R}^{||X_j| \times |X_i||} \mid \gamma 1 = \hat{\mu}_j \text{, } \gamma^T 1 = \hat{\mu}_i \} \) ensures that the transport plan \( \gamma \) does not create or remove mass (by ensuring that the marginal distributions \( \hat{\mu}_j \) and \( \hat{\mu}_i \) are preserved).

Two set of points, even if drawn from the same distribution, will exhibit a variable non-zero Wasserstein distance. To compensate for this sampling-induced bias and variance, we normalize the Wasserstein distance \( d^j_i \) by subtracting the mean and dividing by the variance, of \( d^j_i \), obtained by sampling subsets of the source domain and computing their Wasserstein distance.

Transforming distances into scores Possible score functions include the inverse distances (\( \frac{1}{d^j_i} \)) or the inverse squared distances (\( \frac{1}{d^j_i^2} \)). Here, we focus on the negative exponential scoring function:

\[
s^j_i = e^{-\lambda d^j_i} \tag{4}
\]

The parameter \( \lambda \) allows a smooth interpolation between putting all the weight on the closest domain and having a uniform distribution of all domains. Thanks to this parameter, we will be able to control the variety of the subset of domains we are considering (see below).

Ensuring training set variety In case of many source domains and when some of them have a very small number of training examples, it becomes important to avoid selecting too few domains in the process (e.g., a single one). For generalization (i.e. avoiding overfitting) and transfer purpose, the training set should exhibit enough variety. For domain sizes \( n \in \mathbb{N}^D \), a probability vector \( p^j \) and a draw of \( N \) training examples, it becomes important to avoid selecting too few domains in the process (e.g., a single one). For generalization (i.e. avoiding overfitting) and transfer purpose, the training set should exhibit enough variety. For domain sizes \( n \in \mathbb{N}^D \), a probability vector \( p^j \) and a draw of \( N \) training samples (with replacement), we define the training set variety \( \lambda \) as the expected number of distinct samples we will use for training. The variety can be approximated as:

\[
\text{variety}(p^j, n, N) \approx \sum_i n_i \cdot \left[ 1 - \left( 1 - \frac{p^j_i}{n_i} \right)^N \right] \tag{5}
\]

Our probability vector \( p^j \) depends on the \( \lambda \) parameter. As such, by varying \( \lambda \) from \( \infty \) to 0, we can move from a minimal variety (sampling from the closest domain \( i \), getting a diversity of approximately \( n_i \), the number of examples in this closest domain) to a maximal one (using a uniform \( p^j \), getting a variety of \( N - N \left( \frac{N-1}{N} \right)^N \approx 0.63N \) in case of a balanced \( n \)). In the experiments, we consider one “epoch” (with replacement) of \( N = \sum_i n_i \) samples. We use \( p^j \) and \( n \) to find the highest value of \( \lambda \) for which the variety is below a target variety.

3. EXPERIMENTS

Our backbone classifier is a residual network (ResNet50) [12] pre-trained on the ImageNet dataset [13]. We report the average test-accuracy on the target domain using a 5-fold cross validation procedure. We use the following names to report the model performance:

- **Target (only)**: models trained on the labeled target dataset. Note that this is an ideal but unrealistic situation since, in our actual applications, there is no target label. This setting requires the target domain to be split into training/test sets.
– **Only near:** models trained using only the nearest domain (according to our distance measure) to the target one;
– **Only far:** models trained using only the farthest domain (according to our distance measure) to the target one;
– **LODO:** models trained using all domains but the target one, using domain adaptation on the remaining target domain, without using our domain selection method;
– **w/o near:** models trained using all domains but two: the target and the closest domain to the target one (according to our distance measure) are not used.
– **w/o far:** models trained using all domains but two: the target and the farthest domain to the target one (according to our distance measure) are not used.
– **OURS:** models trained by weighting the source domains with our method, using the variety criterion (with a target variety of half its maximum value).

### 3.1. Datasets

**Office-Caltech (O-C) [14]** is a classical domain adaptation benchmark with four domains: Amazon (A), DSLR (D), Webcam (W) and Caltech (C). It is composed of the 10 classes (Backpack, Bike, Calculator, Headphones, Keyboard, Laptop, Monitor, Mouse, Mug, and Video-projector) common between Office-31 [15] and Caltech256 [16].

**ImageNet-Caltech (I-C).** To control the discrepancy between each domain and validate our chosen domain similarity measure, we have designed a dataset using images from Caltech101 [17] (C1), Caltech256 [16] (C2) and from ImageNet [13] (IN) with mixed labels (bird, car, chair, dog, and person, following, among others, [18, 19]) described in Table 1. This dataset is composed of three different types of domain. The “Good” (G) domains are created with the true original classes, the “Bad” (B) domains are created with different but similar original classes, and the “Random” (R) domains are created with randomly chosen classes in the corresponding datasets. With this design, the “Good” domains are expected to be closer to each other since the original labels are the same. The “Bad” domains should be farther away from the “Good” ones and the “Random” datasets should be the farthest (and are expected to be far from each others too).

**Blucime.** In [20], we introduced an image dataset for the classification of risky situations on chairlifts. The task is to detect if a chairlift vehicle is empty, with passengers in a safe situation, or with passengers in an unsafe situation (security railing not put down, children alone, ...). The images come from 21 different chairlifts: we consider that each chairlift represents a different domain. This dataset is currently not publicly available, so the actual sky resort names are replaced by letters in the corresponding performance table.

### 3.2. Results

In Figure 1, we show the probabilities we obtain using the selection process described in Section 2 on the 3 datasets.

<table>
<thead>
<tr>
<th>Domains</th>
<th>Class</th>
<th>Dataset Classes</th>
<th>Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_C1</td>
<td>bird</td>
<td>“pigeon”, “flamingo”, “bus”, “rooster”, “emu”</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“car_side”</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“chair”, “windsor_chair”</td>
<td>0.118</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“dalmatian”</td>
<td>0.067</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“Faces”, “Faces_easy”</td>
<td>0.07</td>
</tr>
<tr>
<td>G_IN</td>
<td>bird</td>
<td>“birds” (56)</td>
<td>1.456</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“motorcar” (8)</td>
<td>1.496</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“chair” (3)</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“domestic dog” (117)</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“individual” (2)</td>
<td>1.5</td>
</tr>
<tr>
<td>B_C1</td>
<td>bird</td>
<td>“butterfly”, “dragonfly”</td>
<td>1.59</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“Motorbikes”</td>
<td>0.798</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“grand_piano”</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“cougar_body”</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“buddha”</td>
<td>0.85</td>
</tr>
<tr>
<td>B_C2</td>
<td>bird</td>
<td>“024.butterfly”</td>
<td>1.12</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“072.fire-truck”, “178.school-bus”</td>
<td>0.216</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“011.billiards”</td>
<td>0.278</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“105.horse”</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“038.chimp”; “090.gorilla”</td>
<td>0.322</td>
</tr>
<tr>
<td>B_IN</td>
<td>bird</td>
<td>“butterfly” (6)</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“motortrunk” (9)</td>
<td>1.494</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“dining_table, board”</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“domestic cat” (5)</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“apes” (5)</td>
<td>1.5</td>
</tr>
<tr>
<td>R_C1</td>
<td>bird</td>
<td>“brian”</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“chandelier”</td>
<td>1.07</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“watch”</td>
<td>0.239</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“ketch”</td>
<td>0.114</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“bonsai”</td>
<td>0.128</td>
</tr>
<tr>
<td>R_IN</td>
<td>bird</td>
<td>“saltshaker”</td>
<td>1.473</td>
</tr>
<tr>
<td></td>
<td>car</td>
<td>“hiddler crab”</td>
<td>1.182</td>
</tr>
<tr>
<td></td>
<td>chair</td>
<td>“brown bear”</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>dog</td>
<td>“banana”</td>
<td>0.149</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>“dough”</td>
<td>1.249</td>
</tr>
</tbody>
</table>

Table 1: Classes used in each dataset to create the different domains (G = “good”; B = “bad”; R = “random”; C1: from Caltech101; C2: from Caltech256; IN: from ImageNet). In parentheses: number of classes composing the superclass that has been used (e.g. 56 classes of birds).

Fig. 1: Probabilities used to reweight the different domains: ImageNet-Caltech (top-left), Office-Caltech (bottom-left), Blucime (right).

In most datasets, only up to three domains have a selection probably greater than 0.1 and more than half the source domains are totally unused. For instance, during training with "Bad" ImageNet as the target domain, 67.2% of the training images come from "Bad" Caltech256, 21.6% from "Good" ImageNet, 6.7% from "Good" Caltech101, and only 4.6% from the three other source domains.
In Table 2, we show the results on the Office-Caltech (O-C, first five columns) and ImageNet-Caltech (I-C, last nine columns) datasets. On both datasets, using only the nearest source domains is beneficial compared to using the farthest one (+5.49 accuracy points on O-C and +44.21 pts on I-C) which suggests that our distance is meaningful. On Office-Caltech, using the target domain as source (Target line) gives worst performance than using the nearest source (-2.23 pts), which can be explained by the lack of training data which induces overfitting phenomena. Since ImageNet-Caltech contains more data, the Target setting is much more suited, as expected, than the Only nearest one (+36.78 pts).

Using the LODO setting (all source domains are used during training), we observe better performance than with the Only nearest and Only farthest ones thanks to a more diverse training set (respectively, on O-C, +0.46 pts and +5.95 pts, on I-C, +6.73 pts and +50.94 pts). This means that there is a real trade-off between the domain selection and the number of remaining training data. However, if we remove the nearest source domain, the performance becomes worst than for the LODO setting (-5.12 pts on O-C and -6.41 pts on I-C), on Office-Caltech we even get worst performance than using the Only nearest setting (-4.66 pts). If we remove the farthest source domain, we do obtain better performance than with the LODO setting (+1.64 pts). However, if we ignore the results on the "Random" domains (which are close to random by design), on average, the LODO setting gives 91.67% of accuracy, w/o farthest 92.58%, and our approach allows us to get the best accuracy performance of 93.14% which confirms the relevance of our approach.

In Table 3, we show the results on the Bluecime dataset. Due to the page limit, we only detail the results on 4 domains (the same ones as in [20]) and give the averaged results over the 21 domains. As with the other datasets, we obtain better results by selecting the source domains (+1.12 pts). This shows that the proposed method works well even when there are much more source domains to select from.

### 4. CONCLUSION

We have shown that unsupervised domain adaptation can be improved by selecting and weighting a good subset of the sources that are the most similar to the target domain. Our approach weights the sources according to the Wasserstein distance between unlabeled domain distributions and according to the variety of the data in the selected sources. Extensive experiments showed the relevance of our proposed weighting scheme. Future work involves exploring and reporting the behavior of our approach with different settings (e.g. combination of distances, scoring functions, variety criterion), that were left out due to the page limit.

### Acknowledgements

This work is supported by the FUI MIVAO project.

### 5. REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan, “A theory of learning from different domains;”


[6] Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye, “Multisource domain adaptation and its application to early detection of fatigue,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 4, pp. 18, 2012.


