Domain Mixture: An Overlooked Scenario in Domain Adaptation

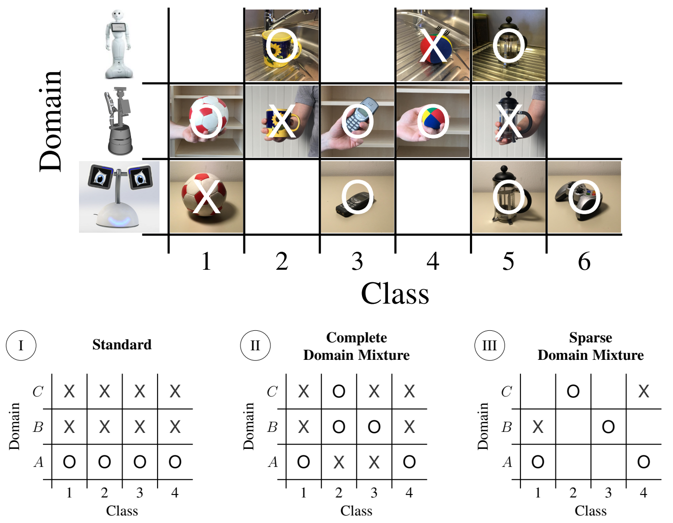
1st Sebastian Schrom
Control Methods and Robotics Lab
TU Darmstadt
Darmstadt, Germany
sebastian.schrom@rmr.tu-darmstadt.de

2nd Stephan Hasler
Honda Research Institute
name of organization (of Aff.)
Offenbach, Germany
email address

***Abstract*—An image based object classification system trained on one domain usually shows decreased performance for other domains if data distributions differ significantly. There exist various domain adaptation approaches that improve generalization between domains. However, those approaches consider during transfer only the restricted setting where supervised samples of all competing classes are available from the source domain. We investigate here the more open and so far overlooked scenario, where during training only a subset of all competing classes is shown in one domain and another subset in another domain. We show the unexpected tendency of a deep learning classifier to use the domain origin as a prominent feature, which is resulting in a poor performance when testing on samples of unseen domain-class combinations. With an existing domain adaptation method this issue can be overcome, while additional unsupervised data of all unseen domain-class combinations is not essential. First results of this overlooked scenario are extensively discussed on a modified MNIST benchmark.**

I. Introduction

Classifying objects from camera images is a very important function for future intelligent systems. There is a natural variance in the appearance of images that show objects, which is strongly influenced by e. g. the environmental conditions or the hardware they are captured with. A good classifier has to be invariant to such variations. In general, given a task $\mathcal{T}$, which is defined by a label space $\mathcal{Y}$, we like to predict the label $y_i \in \mathcal{Y}$ for a certain image. With a set of n sample images $X = \{x_1, ..., x_n\} \in \mathcal{X}$, where $\mathcal{X}$ defines the multi-dimensional feature space of all possible image variations, and corresponding labels $Y = \{y_1, ..., y_n\} \in \mathcal{Y}$, the goal of training a classifier is to approximate the actual conditional probability distribution $P(Y|X)$. This can be achieved by optimizing the parameters of the classifier during training. When training and test images differ systematically with respect to data distributions, both datasets are said to come from different domains. In this case, training and test domain are referred to as source and target domain, with their belonging datasets (X^s, Y^s) and (X^t, Y^t). X^s and X^t, define a subspace of $\mathcal{X}$ and therefore cover only a subset of all possible image variations, which can lead to a poor generalization among the domains when training on a single domain. The application of a system trained on the source domain to an unknown target domain is called domain transfer and methods that improve generalization to the unknown domains are referred to as domain adaptation (DA). Both, source and target domain come with their related tasks $\mathcal{T}^s = \{\mathcal{Y}^s,\ P(Y^s|X^s)\}$ and $\mathcal{T}^t = \{\mathcal{Y}^t,\ P(Y^t|X^t)\}$, which are identical in classic DA, i. e. $\mathcal{T}^s = \mathcal{T}^t$ [1]–[3].



	1	2	3	4
C	X	X	X	X
B	X	X	X	X
A	O	O	O	O

	1	2	3	4
C	X	O	X	X
B	X	O	O	X
A	O	X	X	O

	1	2	3	4
C		O		X
B	X		O	
A	O			O

Fig. 1. *Top*: Distribution of object samples in an interactive object learning setting of different service robots. The robots represent the domains. 'O' describe supervised samples, 'X' unsupervised samples, empty fields missing samples. *Bottom*: DA scenarios, the standard scenario (I) and the overlooked domain mixture scenario in the two configurations (II), (III) investigated in this paper.

The standard case that is investigated in DA is the scenario in which training images are available for all classes of the source domain. In Fig. 1 (I), there are three domains d, with A as the source domain. There are supervised samples for all four classes in the source domain, represented as 'O' and unsupervised samples for the remaining domains, represented as 'X', of which only the domain label d_i is known.

The standard DA scenario postulates a highly regular assignment of supervised and unsupervised data in relation to domains. There are, however, relevant applications where these assumptions are not met. Consider a scenario of an ensemble of service robots, performing object-related tasks in a building and capable of object learning in interaction with users [4]. Each robot may have a limited work space and special camera hardware, but still encounter similar objects across the building and receive partial or overlapping supervised object training on some encounters. Additionally, unsupervised object images are collected without labels. The overall goal is now to optimize a global, domain invariant, object classifier for all robots. We call this underinvestigated DA scenario, which is visualized in Fig. 1 (II & III), the *domain mixture* scenario. We distinguish between the complete (II) and the sparse (III) domain mixture scenario, where the domain-class space is completely or sparsely covered by samples. Since source and target domain here cannot be easily differentiated we are also using the terms *source* and *target* dataset in the following.

In this paper we first categorize related approaches on the standard DA scenario based on the availability of label information and the experimental treatment of multiple domains. We first obtain baseline results of standard DA using an established DA approach and the standard DA scenario (I). We show that DA can have a negative influence if the target dataset is implicitly included in the source dataset. After that, we investigate extensively the complete domain mixture scenario (II) and analyze the effect of DA in relation to the optimal distribution of labels in the domain vs. class space for good generalization. We find that applying DA is essential for avoiding poor generalization to unseen domain-class combinations. Finally, we investigate the relevance of unsupervised samples 'X' in the sparse domain mixture scenario (III). The results show that even without any unsupervised samples DA leads to improved performance.

II. Related Work

In general previous DA approaches assume that the source dataset includes one or several domains that are fully covered by supervised samples for all classes. This allows to call the dataset *source* domain(s), [5]–[7]. The existing DA approaches can be categorized based on the availability of labeled data in the target domain(s). The three groups here are supervised, semi-supervised and unsupervised DA.

In the supervised DA approaches, [5], [6], [8] at least some supervised training samples for all classes in the target domain are available. This separates the group from optimization with a standard training set where domains are not considered. Often the limited target data is used to align features generated by the feature extractor during optimization with the ones generated by the same classes from the source domain [6].

In the semi-supervised case, [9]–[11], only a subset of classes from the target domain is covered by supervised samples and the rest with unsupervised samples, which makes the generalization over the entire domain-class space more challenging. The target data can be used for instance to preserve relationships between this subset of classes across source and target similar to supervised DA, combined with a domain confusion loss which is used to align the extracted features from source and target in general [11].

Most approaches, e. g. [7], [8], [12]–[14], can be categorized into unsupervised DA (see Fig. 1 (I)). Here, the target domain is covered only by unsupervised data, which makes the transfer from source to target domain classes difficult. Since here class preserving methods can not be used at all, a common approach using deep architectures is to introduce additional cost functions attached after a main feature extraction path which are rating the general distance in feature space of source and target domain samples [5], [15]–[17].

Besides the categorization regarding the availability of labels in the target domain, DA approaches can also be categorized by how entire domains are distributed over source and target dataset in the experiments. Often only the transfer from a single source domain to a single different target domain is investigated, as e.g. [18]. Other approaches extend this by including several source domains in the source dataset and evaluating it on a single or several target domains in the target dataset [9], [19], [20]. Rarely the case is investigated where a domain from the target dataset was already included among several other domains in the source dataset [21], or where source and target domains are the same.

The domain mixture scenario that we investigate in this paper (see Fig. 1 (II & III)) shares with all other DA approaches the target to achieve a good generalization over the entire domain-class space. A difference, however, is the fact that there is no domain, of which supervised training samples for all competing classes are available. Here, the notions source and target domain are invalid. Regarding the categorization based on the availability of the labeled data in the target dataset the domain mixture scenario can be assigned to unsupervised DA since there is no labeled data in our target dataset. Therefore we chose to use the approach from [15] as an exemplary unsupervised DA method. Similar unsupervised approaches could also be used.

III. Experiments

Datasets: We use the MNIST dataset [22], defined as S, together with the MNIST-M [15] dataset, M, as domains d for evaluation, $d \in \{S, M\}$ (Fig. 2). The M dataset was originally generated by randomly extracting patches from the BSDS500 dataset [23] while inverting pixels of the digit on each patch. Compared to S this makes the M dataset more complex, since there can also be digits with dark pixels on a bright background together with hard edges within the shape of a digit as shown in the examples. Overall, the variance of digit representations in M is significantly greater and therefore the S samples might be covered in great parts by M. Given the task of classifying the digit on each sample, the label space is defined as $\mathcal{Y} = \{0, 1, ..., 9\}$. In the experiments we clearly separate the pre-defined train and test sets of MNIST. We are aware that other typical DA datasets, [13], [24]–[26] provide more challenging domain transfer scenarios, however, we started with several hundred optimization runs on the MNIST data to reveal the general difficulties of the domain mixture scenario.

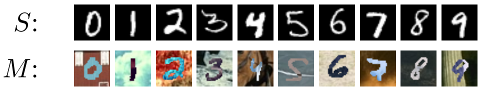


Fig. 2. Datasets for evaluation: Standard MNIST (S) and MNIST-M (M).

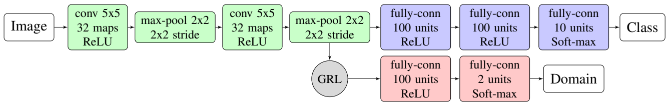


Fig. 3. DA architecture similar to [15]. The shared feature extraction path (green) is followed by the label classifier (blue) and the domain classifier (red) which is connected via a gradient reversal layer (GRL).

TABLE I
STANDARD DA EXPERIMENTS WITH THE MNIST DATASETS. THE RESULTS SHOW THE ACCURACY IN PERCENT ON THE TARGET TEST SETS.

I			
Source:	S	M	$S+M$
Target:	M	S	$S+M$
Source only	54.6	97.6	97.8
DA [15]	76.3	96.3	96.5
Train on Target	96.2	99.2	97.8

Model Architecture and Evaluation Setup: As an exemplary unsupervised DA method we use the deep architecture from Ganin et al. [15], shown in Fig. 3. It has a shared feature extraction path and two independent subsequent classification paths. While the upper path predicts the class label, the lower path predicts the domain d_i of each sample. In this architecture the domain invariance of the features is achieved by the domain path with its gradient reversal layer (GRL) at the connection of the shared feature path and the domain classifier. The GRL reverses the gradient from the domain classifier and therefore both together inhibit to learn domain specific features in the shared feature path during backpropagation.

During batch-training each batch is composed of one half supervised samples from the source dataset of which the labels y_i and d_i are known and one half unsupervised samples from the target dataset of which only d_i is known. The entire batch is used to train the domain classifier, while the label classifier is optimized only with the supervised part. The shared feature path is equally influenced by the gradients coming from both subsequent branches during backpropagation. To support finding a good solution, the influence of the domain path on the shared feature path is adapted over time via the adaptation factor λ, by which the gradient in the GRL is multiplied. For that, λ is smoothly increased from 0 to 1 during training.

In all experiments the architecture was trained until convergence using the update strategy for the learning-rate μ and the adaptation factor λ as described in [15]. Similar to [15] we also used an initial learning-rate of $\mu = 0.01$ and applied dropout in the fully-connected layers. The batch-size was chosen differently with 64 samples and soft-max was applied to the output of both classifiers. Instead of Caffe we implemented the architecture with the deep learning framework TensorFlow.

A. Standard Domain Adaptation (I)

We will first reproduce selected results from [15] which will later be used for comparison to our newly defined domain mixture scenario. For the standard DA we repeat each experiment ten times and report the averaged accuracy.

The results of our standard DA experiments, corresponding to Fig. 1 (I) with ten classes and two domains are shown in Table I. Here the standard notions of source and target *domain* are valid, since entire domains are covered by supervised training samples for all competing classes. For each transfer we report three results, the DA result following the method of [15] and two baselines without DA. These are 'Source only', where training is performed only on the training data of the source domain, and 'Train on Target', where training is done only on the training data of the target domain. The reported numbers are the accuracy on the test set of the target domain.

For the classic transfer $S \rightarrow M$, with S being the source domain and M the target domain, the results are given in column 1. The very low accuracy of only 54.6 % for 'Source only' reflects that the architecture is not forced to learn domain invariant filters. This is due to the low variance in appearance of the digits in the source domain samples. Applying DA in this scenario leads to a great gain in performance to an accuracy of 76.3 %. Here, we can see the positive influence of the domain classifier path on the feature extractor, which guides the features to become more invariant of domains. We could not fully reproduce the results for this experiment as given in [15], where they reached 81.5 %, and trace this back to differences in the deep learning framework. However, the general tendency of the results is visible and sufficient for further experiments. The upper baseline 'Train on Target' shows with 96.2 % accuracy which performance can be reached if training and test are done on M.

For the transfer $M \rightarrow S$, a very high accuracy can already be reached without DA when training only on M, as shown in the second column of Table I. This is due to the fact that the appearance of samples from S is already covered in great parts by M. Using additionally DA, the average performance even slightly reduces, which can be traced back to the influence of the additional loss from the domain path. In the given domain constellation, the application of the approach of [15], and possibly of similar unsupervised DA methods in general, puts an unneeded constraint on the features learned in the shared feature path and hinders to learn optimal features.

In the final standard DA experiment we want to investigate a rather unusual transfer with $(S + M) \rightarrow (S + M)$. The target is to evaluate the effects of applying DA although source and target domain are identical, which can occur when details about dataset distributions are unknown. The performance of the baseline results, which of course are the same, show the expected behavior by lying approximately in the middle between 'Train on Target' of $S \rightarrow M$ and $M \rightarrow S$. Again, for DA the accuracy drops slightly in comparison to the baselines. We explain this once more with the additional, unnecessary constraint that is put on the features by the gradient coming from the domain classifier. To summarize the results from this and the previous transfer, we can see, that DA, as used here, can have a negative influence if being unaware that source and target domain are the same or one domain is implicitly included in another domain, as it was the case with $M \rightarrow S$.

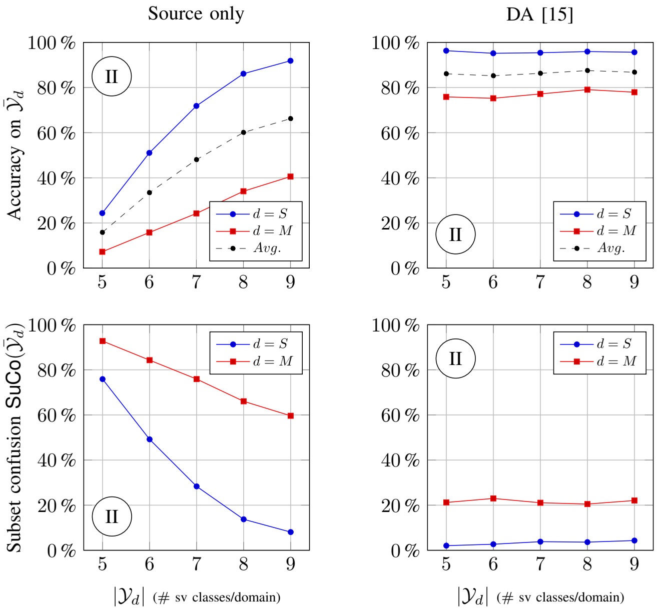


Fig. 4. 'Source only' and DA results for the accuracy on $\bar{\mathcal{Y}}_d$ and the subset confusion $\mathsf{SuCo}(\bar{\mathcal{Y}}_d)$. During optimization of the architecture, the training set, i. e. source dataset for DA, was composed of supervised (sv) samples from S for the classes $\mathcal{Y}_S$, and sv samples from M for $\mathcal{Y}_M$. For DA all domain-class combinations that were not in the source dataset were involved unsupervised during optimization.

B. Complete Domain Mixture Scenario (II)

After investigating the standard DA scenario (I), we evaluate the more open domain mixture scenario in its complete configuration illustrated in Fig. 1 (II). The target here is to see how it behaves with and without DA and to answer the question how labels should be distributed in the domain-class space. In these experiments, the source dataset is composed of supervised samples from the subset of classes $\mathcal{Y}_S \subseteq \mathcal{Y}$ from the S dataset and the subset of classes $\mathcal{Y}_M \subseteq \mathcal{Y}$ from the M dataset. We ensure that the union of both covers all digit classes, i. e. $\mathcal{Y}_S \cup \mathcal{Y}_M = \mathcal{Y}$. Further, we keep both always of the same cardinality, i. e. $p = |\mathcal{Y}_S| = |\mathcal{Y}_M|$. According to these constraints, the possible cardinalities can only be in the range of $5 \leq p \leq 10$. For $p > 5$ this means there are supervised samples for certain classes from both domains in the training set. The case $p = 10$ has already been covered by the standard DA experiment $(S + M) \rightarrow (S + M)$.

The target dataset consists of unsupervised samples for all domain-class combinations not in the source dataset, i. e. $\bar{\mathcal{Y}}_S = \mathcal{Y} \setminus \mathcal{Y}_S$ and $\bar{\mathcal{Y}}_M = \mathcal{Y} \setminus \mathcal{Y}_M$. The subsets $\bar{\mathcal{Y}}_S$ and $\bar{\mathcal{Y}}_M$ also define the test classes on which we evaluate our optimized architectures, with and without DA. The evaluation was done dependent on the number of supervised classes per domain $p = |\mathcal{Y}_d|$ with $d \in \{S, M\}$. For every cardinality we did 35 experiments, choosing each time random classes in $\mathcal{Y}_d$.

The results for the 'Source only' and DA experiments are shown in the upper two plots of Fig. 4. Looking at the results of 'Source only' for a subset size of $|\mathcal{Y}_d| = 5$, where we do not have any overlapping classes of the two domains in the training set, we can see a remarkably bad performance in accuracy on the test samples of the classes $\bar{\mathcal{Y}}_d$, i. e. the domain-class combinations that were not involved during optimization at all. The accuracy over the entire test set (Avg.) is only at 15.8 %. This weak performance is surprising since one could expect that the classifier already knows how to handle samples from both domains, as during training the classifier has seen already all domains and all classes. In comparison to the standard DA experiment $S \rightarrow M$ for 'Source only', where the training set only included a single domain with all classes, the expected accuracy for the experiments here would have been higher.

Due to the unexpected weak performance, we investigated the errors that were done by the optimized classifier further and found, that the wrong classified digit samples of the test set classes $\bar{\mathcal{Y}}_d$ of one domain are mostly mixed up with classes $\mathcal{Y}_d$ of the same domain that were shown during training. This is obviously because the classifier is using the domain characteristics as an additional powerful feature that allows a simplified classification of the given training set. For the training data, it is sufficient to identify the domain origin of each sample, i. e. to split the data into two groups for the two domains, and then based on that, classify which digit is shown. This simplifies the classification, since after the pre-grouping, the classification only needs to be made among five classes instead of ten, for the case of $|\mathcal{Y}_d| = 5$. However, this learned pre-grouping is harmful when it comes to the classification of the unseen domain-class combinations in the test set as we can see at the poor accuracy of our experiments.

To quantify the influence of this pre-grouping, we define the subset confusion $\mathsf{SuCo}(\bar{\mathcal{Y}}_d)$ for a given test set of the classes $\bar{\mathcal{Y}}_d$ with samples i, the given ground-truth labels y_i and the predicted labels y'_i as:

$$\mathsf{SuCo}(\bar{\mathcal{Y}}_d) = \frac{|\{i \mid y_i \in \bar{\mathcal{Y}}_d,\ y'_i \in \mathcal{Y}_d\}|}{|\{i \mid y_i \in \bar{\mathcal{Y}}_d\}|} \qquad (1)$$

Here, a value of 0 means that all the predicted classes of the test samples are in the correct subset $\bar{\mathcal{Y}}_d$. However, this does not necessarily mean that the prediction was correct. A value of 1 would mean that all test samples were mixed up with classes from the subset that was included in the source dataset. In this case no prediction is correct.

In the bottom plots of Fig. 4 we show the subset confusion for our experiments. For the case of $|\mathcal{Y}_d| = 5$ the subset confusion is very high and explains the low accuracy of this constellation. Comparing the graphs of the accuracy and the subset confusion, we can see that the subset confusion is basically the vertically flipped graph of the accuracy. This indicates, that most classification errors are due to the confusion of the subsets and therefore due to the learned pre-grouping.

With an increasing number of supervised classes per domain $|\mathcal{Y}_d|$, the accuracy on the unseen domain-class samples goes up to 66 % over all test samples. We assume this is mainly caused by the increased overlap of classes in $\mathcal{Y}_S$ and $\mathcal{Y}_M$ in the training set. With such an overlap the gain of the classifier using the domain origin as additional feature is reduced and leads to the classifier focusing more on the actual shape feature of the digits, which in turn also effects features learned for the domain exclusive classes. In general, the experiments here showed again the difference in difficulty of the two used domains since the accuracy on the unseen domain-class samples of S is always higher in comparison to M.

Using the DA method [15] we can see at the accuracy shown in the top right plot of Fig. 4 that already at $|\mathcal{Y}_d| = 5$, without any overlap in supervised samples, the average performance is with 86 % very high. This result is in line with the expectations by being exactly in the middle of the standard experiments $S \rightarrow M$ and $M \rightarrow S$ when DA is applied. The domain classifier path obviously largely inhibits to learn the mentioned pre-grouping of the training samples. The results for the increased cardinalities show that the accuracy is approximately independent of an increasing overlap of classes in $\mathcal{Y}_S$ and $\mathcal{Y}_M$. This is surprising since a similar effect to the experiments without DA could have been expected.

A general conclusion that can be drawn from the complete domain mixture experiments (II) and the standard DA experiments (I) with regard to the service robot setting from Fig. 1 is that if the variance in appearance of classes within the domains is known, then providing the labels to the domain with the highest variance achieves the best performance (see $S \rightarrow M$ vs. $M \rightarrow S$). This does not necessarily have to be combined with DA. However, if there is no information about the variance in class representations within the domains and goal is to keep the minimal performance as high as possible, it is better to distribute the labels among the different domains and combine this with DA to avoid the pre-grouping effect.

C. Sparse Domain Mixture Scenario (III)

Since in the complete domain mixture scenario (II) all domains are already represented with supervised samples 'O' in the training set, the question arises how many or if any additional unsupervised samples 'X' are necessary at all. The domain classifier can already be trained only on the supervised samples included in the training set. Furthermore, the sparse configuration of the domain mixture scneario (III) is closer to the introduced real-world example. In our previous experiments, $\bar{\mathcal{Y}}_d$ always represented all unsupervised (usv) domain-class combinations during training, i. e. $\bar{\mathcal{Y}}_d = \bar{\mathcal{Y}}_{d_{\text{usv}}}$. Now we define $\bar{\mathcal{Y}}_d = \bar{\mathcal{Y}}_{d_{\text{usv}}} \cup \bar{\mathcal{Y}}_{d_{\text{unseen}}}$, where $\bar{\mathcal{Y}}_{d_{\text{unseen}}}$ stands for the domain-class combinations that were not shown during optimization at all. We investigated the necessity of additional unsupervised samples for the fixed number of supervised classes per domain $|\mathcal{Y}_d| = 5$ and did experiments for $|\bar{\mathcal{Y}}_{d_{\text{usv}}}| \in \{0, ..., 5\}$.

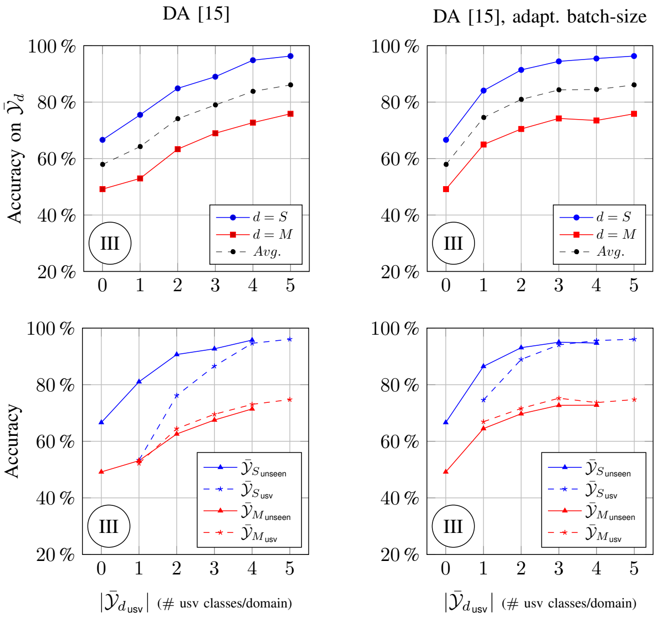


Fig. 5. Investigation of the necessity of unsupervised samples in the chosen DA approach for a fixed number of supervised classes per domain $|\mathcal{Y}_d| = 5$. The right plots show the result for an adapted batch-size, where the appearance frequency of domain-class combinations in the batch is balanced. *Bottom*: Split up of the upper results into actual unseen and unsupervised (usv) domain-class combinations during training.

The results for these experiments are shown in the plots of Fig. 5. Here $|\bar{\mathcal{Y}}_{d_{\text{usv}}}| = 5$ corresponds to the result of (II) with DA and $|\mathcal{Y}_d| = 5$. In the top left plot we can see at $|\bar{\mathcal{Y}}_{d_{\text{usv}}}| = 0$, the case where no additional unsupervised data is added. If we compare the average accuracy of 57.9 %, with the average accuracy of 15.8 % for 'Source only' at $|\mathcal{Y}_d| = 5$ in Fig. 4, which has the same training set constellation, we can see a clear gain in accuracy. This supports the use of the DA method in the domain mixture scenario, even without additional unsupervised data. The domain path of the architecture is obviously capable of partially breaking up the pre-grouping, only with the help of the supervised samples included in the source dataset. Overall, we can see from this plot also the expected increase in accuracy with an increasing number of unsupervised domain-class combinations.

Another outcome of these experiments is shown in the bottom left plot of Fig. 5. Here we split up the test accuracy into $\bar{\mathcal{Y}}_{d_{\text{usv}}}$ and $\bar{\mathcal{Y}}_{d_{\text{unseen}}}$. For the samples of the unsupervised classes of M, the performance is independent of $|\bar{\mathcal{Y}}_{d_{\text{usv}}}|$ constantly slightly better than for the samples from the unseen classes. This is an expected result since the unsupervised classes were already involved in the optimization of the architecture. Surprisingly, for the S domain, we can see that the accuracy on $\bar{\mathcal{Y}}_{S_{\text{usv}}}$ is constantly below $\bar{\mathcal{Y}}_{S_{\text{unseen}}}$. This effect is very pronounced for small $|\bar{\mathcal{Y}}_{d_{\text{usv}}}|$. We explain this effect by the simplicity of the S dataset on which already using samples from M a good accuracy can be achieved (see Table I, column 2). Therefore, adding a domain classifier with an additional constraint on the learned domain-class combinations can possibly have a negative influence.

To investigate the reason for this unexpected result further, we had a look at the batches used during training. As mentioned in the introduction of the architecture, the second half of each batch is filled up with samples of the available unsupervised samples, of which only the domain label d_i is known. For the experiments here, this can lead to a strong imbalance among all domain-class combinations when training the domain classifier. This means that for a batch-size of

64 samples and ten supervised domain-class combinations, each supervised combination appears on average 3.2 times in the first half of the batch. If we follow for the experiments here the procedure to fill up the second half of each batch with the unsupervised domain-class combinations, we generate an imbalance within the batch regarding the appearance frequency. With less unsupervised than supervised domain-class combinations the unsupervised samples appear with a higher frequency, which can have a misleading effect during optimization. Therefore, we chose a heuristic to determine the batch-size for each experiment individually. To keep the average appearance of 3.2 times per domain-class combination we increase the batch-size starting from 32 samples by approximately 3.2 samples for each additional unsupervised combination. According to the increase of the number of unsupervised classes per domain $|\bar{\mathcal{Y}}_{d_{\text{usv}}}|$ this leads to the batch-sizes $BS \in \{32, 38, 45, 51, 58, 64\}$. Note that this heuristic does not necessarily define the optimum solution. The results in the right plots of Fig. 5 show that the general accuracy for all affected experiments can further be improved by adapting the batch-size. Nevertheless, as shown in the bottom right plot, the described effect that $\bar{\mathcal{Y}}_{S_{\text{usv}}}$ is constantly below $\bar{\mathcal{Y}}_{S_{\text{unseen}}}$ could only be reduced but not avoided completely.

Strictly speaking the applied adaptation of the batch-sizes would also have been necessary for the DA experiments in Fig. 4, since for $|\mathcal{Y}_d| > 5$ such an imbalance exists as well. However, note, this adaptation is hardly to realize in a real-world scenario when the number of represented classes in the unsupervised data is unknown.

IV. Conclusion

In this paper we investigated the so far overlooked scenario of domain mixture in DA, where the assumption is that several domains are represented with supervised samples in the training set, but no domain is fully represented by supervised samples for all competing classes. We elaborated that this is a relevant real world scenario that can occur for instance in human-multi-robot interaction. Given the domain mixture scenario in the domain-class space, we showed with the example of MNIST data that a CNN classifier pre-groups the training samples by their domain origin and uses this as a strong feature for classification, which makes a generalization to unseen domain-class combinations hardly possible. However, when using the DA method of [15], which makes use of additional unsupervised domain-class samples of the unseen combinations, we saw that the pre-grouping can be inhibited, even without any overlapping supervised samples from both domains. Further we showed, that not all possible unsupervised domain-class combinations need to be included during training. Therefore we state, if being limited to labeling only certain domain-class combinations, and the goal is to achieve always the highest minimal performance, it is better to distribute the labels among the different domains, i. e. generating the domain mixture scenario, and to combine this together with DA. As we have shown in our experiments, doing so is only harmful if one of the domains covers already a great variance in class appearances. Then it is most beneficial to label only the classes of the domain with higher variance. In further experiments it needs to be investigated whether the results also hold for datasets where the domain separation is not as clear as in the investigated MNIST datasets.

References

[1] G. Csurka, “A comprehensive survey on domain adaptation for visual applications,” in *Domain Adaptation in Computer Vision Applications.* Springer, Cham, 2017, pp. 1–35.

[2] A. Arnold, R. Nallapati, and W. W. Cohen, “A comparative study of methods for transductive transfer learning,” in *ICDM*, 2007, pp. 77–82.

[3] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[4] S. Hasler, J. Kreger, and U. Bauer-Wersing, “Interactive incremental online learning of objects onboard of a cooperative autonomous mobile robot,” in *ICONIP*, 2018, pp. 279–290.

[5] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *CoRR*, vol. abs/1412.3474, 2014.

[6] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, “Unified deep supervised domain adaptation and generalization,” in *ICCV*, 2017, pp. 5716–5726.

[7] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” in *CVPR*, 2017, pp. 95–104.

[8] A. Rozantsev, M. Salzmann, and P. Fua, “Beyond sharing weights for deep domain adaptation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 801–814, 2019.

[9] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, “Revisiting batch normalization for practical domain adaptation,” in *ICLR (Workshop)*, 2017.

[10] B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *CVPR*, 2012, pp. 2066–2073.

[11] J. Hoffman, E. Tzeng, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” in *Domain Adaptation in Computer Vision Applications.* Springer, 2017, pp. 173–187.

[12] B. Sun and K. Saenko, “From virtual to reality: Fast adaptation of virtual object detectors to real domains,” in *BMVC*, 2014.

[13] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *ECCV*, 2010, pp. 213–226.

[14] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, “Unsupervised visual domain adaptation using subspace alignment,” in *ICCV*, 2013, pp. 2960–2967.

[15] Y. Ganin and V. S. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *ICML*, 2015, pp. 1180–1189.

[16] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, “Domain separation networks,” in *NIPS*, 2016, pp. 343–351.

[17] B. Sun and K. Saenko, “Deep CORAL: correlation alignment for deep domain adaptation,” in *TASK-CV*, 2016.

[18] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *CVPR*, 2017, pp. 2962–2971.

[19] J. Hoffman, B. Kulis, T. Darrell, and K. Saenko, “Discovering latent domains for multisource domain adaptation,” in *ECCV*, 2012.

[20] R. Gopalan, R. Li, and R. Chellappa, “Domain adaptation for object recognition: An unsupervised approach,” in *ICCV*, 2011, pp. 999–1006.

[21] S. Schrom and S. Hasler, “Effects of domain awareness in generalizing over cameras in road detection,” in *Machine Learning Reports*, 03/2017.

[22] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[25] T. Tommasi and T. Tuytelaars, “A testbed for cross-dataset analysis,” in *ECCV Workshops*, 2014, pp. 18–31.

[26] A. Bergamo and L. Torresani, “Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach,” in *NIPS*, 2010, pp. 181–189.