Improving Image Classification Across Domains by Solving Multiple Self-Supervised Tasks

Bagnoli Giulio
giulio.bag@gmail.com

Abstract

One thing all machine learning scholars agree on is that a secret to getting a model that works well is that we need to train it on a huge dataset. Some of them are extremely expensive to collect and it is not always possible to recover a large amount of data. Aware of this problem, two solutions have been proposed: Domain Shift, which consists in to train the model on data distributions similar to those on which it will operate, and Self Supervised Learning, which is a paradigm that attempts to get the machines to derive supervision patterns from the data itself, without human involvement. The proposed approach combines these two strategies by building a model capable of learning features independent of the domain from which the images come. This knowledge is then applied to the dataset on which the model will operate. To extrapolate these features various techniques have been proposed that have been tested individually and in groups, reporting significant improvements compared to the simple use of Domain Shift strategies. Contrary to the standard approach of Self Supervised Learning, in our idea, the network trains on the main task and on the secondary ones in parallel. Both contribute to the model's weights update.

1. Introduction

In a supervised learning context, models need a huge amount of labeled samples to correctly perform their tasks. This data is not only very expensive to acquire, but it also has to belong to the same distribution of the test samples. Therefore it is necessary to study techniques that allow training the models partially on unlabeled data belonging to other distributions. The usage of data that comes from a different source belongs to a set of approaches called "domain shift". In this paper, we face this problem using multiple sources and exploiting self-supervised techniques.

Domain shift is a sub-discipline of machine learning that deals with scenarios in which a model trained on a source distribution is used in the context of a different, but related, target distribution - e.g. a model that was trained on photos and is used to classify paintings. In general, domain shift uses data from one or more source domains to build a model able to solve new tasks in a target domain. The level of relatedness between the source and target domains usually determines how successful the adaptation will be.

The cases in which it is possible to apply domain shift could be split according to different aspects such as the number of source domains, the type of available data from the target domain, or the presence at train time of the test set without labels. This last aspect is used to divide the experiments in this paper. We have considered two different scenarios:

- Unsupervised Domain Adaptation (DA): we know the unlabeled target data at train time and we use it to improve the model.
- Domain Generalization (DG): we do not know the target data at training time, we only have one or more source domains different from the target one.

Self-supervised techniques start from unlabeled data and, after removing part of the inherent information from them, ask the model to predict it back. In this way, we are forcing the network to find useful information in terms of semantics (shapes, relations between parts, etc...). Contrary to the standard approach, in which first we train the model to find the hidden information through the unlabeled data and then use the weights obtained in the previous network to initialize a new one, in the proposed experiments we train a network that learns at the same time how to solve a classification problem and a self-supervised one. This secondary task helps the network to learn spatial features, independent from the appearance of the image.

In the perform work it is given particular attention to which one of the implemented self-supervised tasks is the most appropriate to help the model in its classification task. We observed the following ones:

- Jigsaw Puzzle: the image is divided into an arbitrary number of patches that are randomly recombined; the goal of the model is to find the pieces in the correct order.

- Jigsaw Puzzle Style Transfer: as in the standard Jigsaw task, but each patch is transformed using a different style; the goal of the model is still to find the correct order of the pieces.

- Odd-One-Out: we substitute one of the patches with a random one belonging to an image of the same class and source of the image under analysis; the goal of the model is to find the unrelated piece.

- Rotation: the image is rotated and the model has to define the rotation angle.

2. Related Works

Self-Supervised Learning. Several works have studied domain adaptation problems in depth. The solutions can be divided in three categories: *model-based*, *feature-level* and *data-level*. In the first group, we can find both shallow and deep learning methods that build over multi-task learning [14] or low-rank network parameter decomposition [15]. While in the second group there are methods that search for an embedding space where images of the same class but different sources are projected nearby [16]. The data-level approach tries to augment the source domain to get closer to the target one. This strategy is used in [21] in which models are trained on simulated random images, indeed with enough variability in the simulator, the real world may appear to the model as just another variation. Augmentation can also be obtained with domain-guided perturbations of the source instances like in [19]. Our work proposes an alternative solution in which we exploit the extracted features used for some self-supervised tasks to attach the domain adaptation problem jointly training both self-supervised and main classifiers.

Self-supervised learning is a widely explored field and there is a wide range of tasks that can be exploited. The usage of a **jigsaw puzzle** as a self-supervised task has already been exploited on multiple works [1, 17]. Moreover, this task is also used in other fields such as computer graphics and image editing [20, 2]. The problem can be formulated as a classification task or can be solved by exploiting greedy strategies, among them, we have the usage of minimum spanning tree algorithm in [7]. Contrary to [3], in this paper, not all possible permutations are considered but only a subset of them. The algorithm by which the permutations are chosen is explained in [17]. The **style transfer** problem has origin in texture synthesis and transfer. The first strategies to attach the problem were based on a histogram matching on linear filter responses [11] and non-parametric sampling [4, 6]. The first technique that achieved good results using convolutional layers were Gays *et al.* in [9]. Their framework is based on a slow optimization process that iteratively updates the image to minimize a content loss and a style loss computed by a loss network. To make the process faster a common workaround is to replace the optimization process with a feed-forward neural network that is trained to minimize the same objective [13]. However, the above feed-forward methods are limited in the sense that each network is tied to a fixed style. Since we needed a model able to operate on different styles and on a huge amount of images we decided to use the model developed in [12]. It reaches a speed comparable to the fastest existing approach, without the restriction to a pre-defined set of styles. The **odd-one-out** is a self-supervised task usually used for self-supervised video representation learning. In this kind of task, the network takes several video sequences as input and its goal is to identify the odd one [5]. This idea can be reflected in images passing to the network several patches and asking it to identify the unrelated one. Eventually, the **rotation** was introduced as a self-supervised task. It is a non patch-based task and in [10] Gidaris *et al.* demonstrates its efficiency.

In our work, all these self-supervised tasks at first are evaluated and compared and then tested together in order to find the best combination that achieves the highest classification score.

3. Beyond the JiGen Approach

Starting from the same basic idea of [1], we would like to train a model capable of further improving the results already achieved. To do this, since solving the puzzle has brought benefits, the starting intuition is to apply other different self-supervised tasks or a combination of them. Let's consider the case in which we exploit a single self-supervised task. Just as it happens for JiGen, the purpose of the network is not only to measure the error of the main classification problem, by comparing the predicted labels with the ground truth, but it has to also satisfy a second condition that depends on the chosen self-supervised task. This idea can be extended in order to perform several self-supervised tasks at the same time: the network has to satisfy several extra conditions, one for each task, in addition to continuing to perform the main classification problem, which remains unchanged. Thanks to the fact that all these tasks do not need the original image label, we can use them even for unsupervised domain adaptation, exploiting the unlabeled target data available at training time. In detail, the self-supervised tasks that were used to support the classification in this work are:

Jigsaw Puzzle This task is the same exploited by JiGen architecture. The input images are divided into patches using an $n \times n$ squared grid and then these patches are shuffled. The new patch indexes are not chosen randomly from the $n^2!$ possible permutation, but they are select from a restricted set of P permutations generated using the Hamming distance-based algorithm, in order to maximize the distance

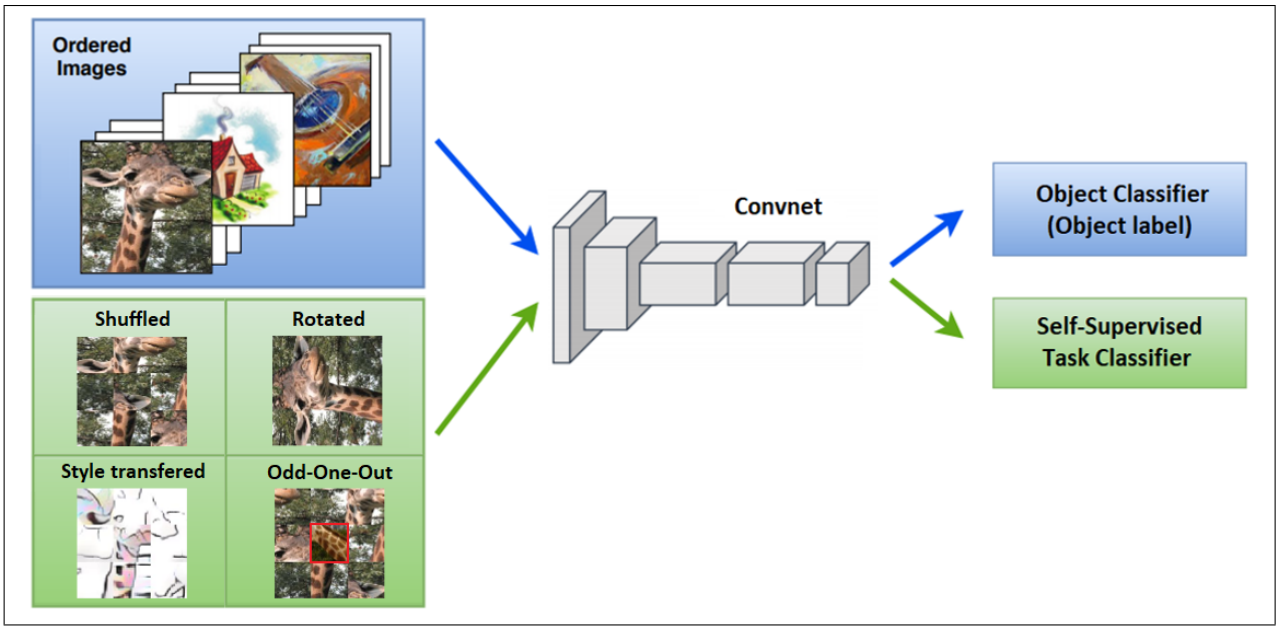

Figure 1. Illustration of the developed architecture. Starting from the images of PACS we apply some changes in order to remove some relevant information that is to be recovered by solving a self-supervised task. The network is then trained simultaneously on the main problem and the self-supervised tasks to achieve a better classification score.

between them. Eventually, we obtain our self-supervised task, which is a classification problem where the label is the permutation index selected to shuffle the image patches.

Jigsaw Puzzle Style Transfer It is a more challenging variation of the previous task. The only addition is that, after splitting the image, a style transformation is applied to each patch. The style to be applied is chosen differently depending on whether we are in a DG or in a DA context. In the former, the style to transfer is taken from an image of a different source domain while in the latter it is taken from a target domain image. Moreover, the transformation takes place starting from the work developed in [12]. This style transfer model is trained again based on the context. For DG only the source domains are used to train the network while for DA also the target domain is considered.

Odd-One-Out For this task the images are split in patches, as for the jigsaw puzzle, but, before shuffling them according to the selected permutation, one of the patches is replaced with another one from a different image. The replacement is chosen from an image of the same domain and class as the original one, in order to make the task more challenging. Eventually, we obtain a self-supervised task that is a classification problem where the label is the index of the foreign patch.

Rotation To perform this task, the input images are simply rotated by 0, 90, 180 or 270. Unlike the other tasks, there is no need for subdivision into patches. The self-supervised task consists of a classification task where the label is one of the four possible orientation angles applied to the image.

Implementation Details This work studies both DG and DA context but it uses a different basic network structure for each one. In the former, we build our architecture over AlexNet, while in the latter we modify a ResNet18 network. ResNet18 is notoriously better than AlexNet so it is conceivable to achieve better results in the DA, not only thanks to the availability of the images of the target domain but also thanks to the better basic structure of the network used. In this work, these networks have multiple final layers, one for each self-supervised task plus one of the main classification problem.

In DG case, each branch produces its unique loss value that is computed as a cross entropy loss. These are combined together through a weighted sum and used in the backpropagation step. The weight of the sum for the t-th task is denoted by α_t. Moreover, each task has another parameter that is the *data bias* β_t, that in this work indicates the percentage of images in a batch dedicated to the task to which it refers (note that, with respect to [1], our parameters β work in the opposite way). Noteworthy, the main classification branch does not have these parameters because its α value is always set to 1 and its β value is the difference between 1 and the sum of the others β. These parameters are tuned during the train. From a mathematical point of view we describe the components that influence the final loss (L_{DG}): if S are the domains on which the model (h) operates, we assume that each of them has N_i labeled im-

ages $\{(x_j^i, y_j^i)\}_{j=1}^{N_i}$, where x_j^i indicates the image and y_j^i the class. We also assume that to each task is assigned a different and disjoint group of images $\{(z_{j_t}^i, p_{j_t}^i)\}_{j=1}^{N_{i_t}}$, where N_{i_t} is the cardinality of the set of images reserved for the task t for the domain i, $z_{j_t}^i$ indicates the transformed image and $p_{j_t}^i$ the task label. Consequently, the cardinality of the set of images reserved for the main classification problem for the domain i is $N_{i_c} = N_i - \sum_t N_{i_t}$. Eventually, let θ_f be the set of parameters of the part in common to each task, θ_c the branch parameters of the main task and θ_{s_t} the parameters of the t-th secondary task, we can write the L_{tot} as:

$$L_{DG} = \sum_{i=1}^{S} \Big(\sum_{j=1}^{N_{i_c}} L_c(h(x_j^i|\theta_f, \theta_c), y_j^i) + \sum_{t=1}^{3} \alpha_t \cdot \sum_{k_t=1}^{N_{i_t}} L_t(h(z_{k_t}^i|\theta_f, \theta_{s_t}), p_{k_t}^i) \Big)$$

For unsupervised DA, since the α parameter may differ between the source and target losses, each branch of the network produces two different cross entropy losses, that contribute to the weighted sum as in DG case. Moreover, there is also an empirical entropy loss computed on the target images which are not used for any self-supervised task $\{x_j, y_j\}_{j=1}^{N_{target_c}}$. This value is multiplied for an empirical entropy weight and summed to the final loss. The empirical entropy loss for each image is equal to:

$$L_E(x_j) = \sum_{y \in Y} h(x_j|\theta_f, \theta_c) \log\{h(x_j|\theta_f, \theta_c)\}$$

where Y is the set of the possible class label in our domains.

All the other possible learning process parameters are fixed: SGD used, 30 epochs, batch size 128, empirical entropy loss weight $\eta = 0.1$, learning rate 0.001 and stepped down to 0.0001 after 80% of the training epochs.

The two jigsaw puzzle tasks have also two parameters related to the task itself, the dimension of the permutation set P and the grid size $n \times n$, that are tuned too, the Odd One Out task has only the grid size as extra parameter and the rotation task has no extra parameters related to their performance.

4. Experiments

Code and pretrained models are available in this GitHub repository.

Dataset. We use PACS, a dataset which covers 7 classes and 4 domains, to evaluate all the considered self-supervised tasks in a multi-domain context (Photo, Art Paintings, Cartoon and Sketches). We performed the experiments using three domains out of four as sources and the remaining one as target, as explained in the experimental processes of [1, 15].

4.1. Replicating JiGen Results

Comparison with Baselines. We begin the experiments using our network setup based on the approach described in [1], to replicate the behavior and the results of both the *Deep All* and JiGen techniques proposed in the paper. Our network is a modified version of a standard classification architecture: it has several output layers, the usual one plus one for each self-supervised task. In order to launch our simulations, we disabled the unnecessary tasks by setting their α and β parameters to 0. The situation in which all these parameters are set to 0 corresponds to *Deep All*, which is equivalent to use a standard network pretrained on ImageNet. While setting only the α and β parameter of the jigsaw task, our network emulates the behaviour of JiGen. The results obtained from these experiments are used as a double baseline in order to understand if other self-supervised tasks, or a combination of them, can improve the results achieved in other studies.

In order to confirm that our network was correctly configured and inline with the original work, we first replicated the *Deep All* experiment on PACS dataset. Then, following the work of [1] (which expanded on [17] and [18]), we set up a jigsaw puzzle solver for 9 shuffled patches from images decomposed by a regular 3×3 grid. For each image, the network, with probability β_{jigsaw}, decides whether or not to shuffle it and then applies the Jigsaw task on all of them. Non-shuffled image labels correspond to the permutation index related to the correct patches order. Instead, the classification task is only applied to non-shuffled images. Additionally, in DA, since target images are used also during training, the classification task exploits them computing the entropy loss, which can be weighted in the loss sum. In all these experiments and in the following ones, this weight is always set to 0.1. Table 1 reports the obtained results. The accuracy values reached for each domain are comparable to the baseline ones. Consequently, the overall accuracy over PACS is similar too, with a difference below $\pm 0.5\%$, for both *Deep All* and jigsaw puzzle task approaches.

Learning Parameters Ablation. Once the layer related to the jigsaw puzzle task has been activated, the results mentioned above are obtained through an ablation study. We keep the jigsaw hyperparameters fixed with a 3×3 patch grid and $P = 31$ labels for the jigsaw puzzle task (30 shuffled plus 1 non-shuffled). Since experiments in [1] suggest that having a higher percentage of shuffled image could lead to generally worse performances, our studies focus only on some values in the $[0.1, 0.6]$ range for β, while we do not set any hard limit on α which values are sampled in the $[0.1, 1]$ range. For example, by setting the data bias to a lower value, we provide to the network more ordered

Task	Art	Cartoon	Sketch	Photo	Avg.
DG - AlexNet					
Deep All	66.68	69.41	60.02	89.98	71.52
JiGen	67.63	71.71	65.18	89.00	73.38
Deep All	66.45	70.12	61.82	89.30	71.92
Jigsaw	68.60	71.29	**65.60**	89.60	**73.77**
Rotation	67.77	70.43	65.15	**90.29**	73.41
Style Transfer	65.72	71.97	62.74	89.79	72.56
Odd One Out	68.01	**72.14**	58.06	89.40	71.90
Jig + Rot	**68.70**	71.50	64.46	89.82	73.62
Jig + Odd	67.38	71.45	64.64	88.74	73.05
Rot + Odd	65.38	71.97	60.75	89.04	71.78
Jig + Rot + Odd	68.45	71.24	64.41	87.96	73.02
DA - ResNet 18					
Deep All	77.85	74.86	67.74	95.73	79.05
JiGen	85.08	81.28	81.50	97.96	86.46
Deep All	78.56	76.15	65.41	96.34	79.11
Jigsaw	85.97	81.87	78.06	97.61	85.88
Rotation	89.23	86.40	82.32	97.82	**88.94**
Style Transfer	84.77	80.84	73.40	97.25	84.60
Odd One Out	88.06	**86.97**	**82.46**	96.65	88.53
Jig + Rot	**90.04**	82.59	78.75	97.50	86.97
Jig + Odd	85.21	82.38	69.05	97.48	83.53
Rot + Odd	89.89	82.64	78.52	98.01	87.26
Jig + Rot + Odd	88.04	85.96	77.40	**98.21**	87.40

Table 1. Results obtained on PACS, compared to the JiGen ones. Each column correspond to a different target domain. The best results are highlighted in bold. The top reports Domain Generalization results, while the bottom reports multi-source Domain Adaptation results.

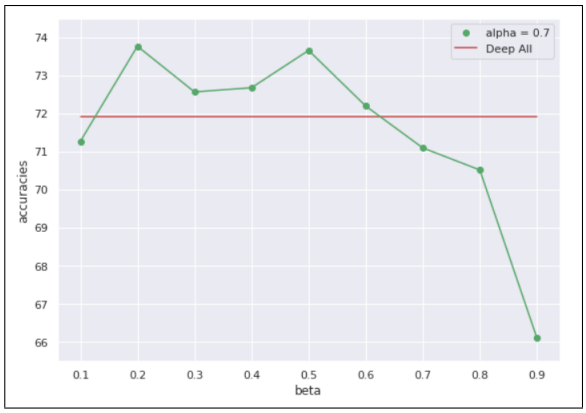

Figure 2. Ablation results in DG for the jigsaw puzzle task. The plot shows the accuracy as β varies, keeping α fixed. The red line represents the *Deep All* baseline.

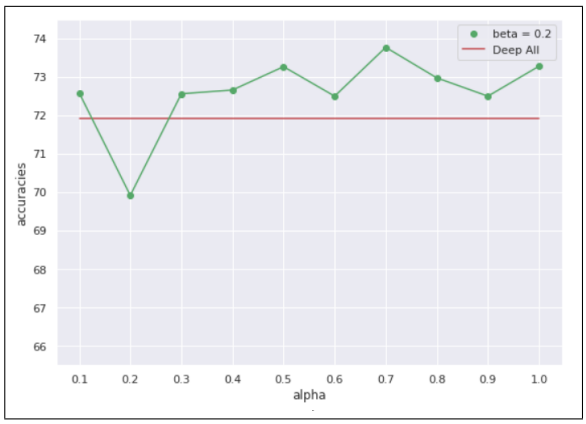

Figure 3. Ablation results in DG for the jigsaw puzzle task. The plot shows the accuracy as α varies, keeping β fixed. The red line represents the *Deep All* baseline.

than shuffled images, thus dedicating to the main classification branch of the network the biggest part of the input dataset. On the other hand, with a higher β value we obtain the opposite situation - i.e. the jigsaw puzzle task receives the biggest amount of images.

Ablation results for DG on PACS are reported in figure 2 and 3. As we can see, the best accuracy, highlighted in top of table 1, is obtained with $\beta = 0.2$ and $\alpha = 0.7$. These values are different from the ones discovered in [1], in particular, we reserve a lower amount of images for the jigsaw puzzle task. Noteworthy the fact that for $\beta \geq 0.7$, the more the value is increased the more the accuracy decreases below the baseline. This is justified by the fact that there is too limited a quantity of images destined for the main classification task, not enough to train it correctly. Regarding the ablation on α, the differences between accuracies reported in the second plot are quite small. This means that for the value of β used, the impact of α on the result is not very significant.

In DA context, the tested hyperparameters during ablation are again selected from the same range of values as the DG case. In DA there is the possibility to choose two different values for α_s and α_t, which represent respectively source and target images loss weight. However, in our experiments, only equal values are selected and used in the simulations. The highest accuracy reached for this task, reported in bottom of table 1, is obtained using $\beta = 0.6$, $\alpha_s = 0.2$ and $\alpha_t = 0.2$.

Jigsaw Puzzle Parameters Ablation. Setting α and β to their best result combination for DG, we ran another ablation test on the two parameters related to this task: grid size and patch number. For the first one, we tested both a higher (4×4) and a lower (2×2) size. The results obtained are reported in table 2 and they show how both instances perform worse than the original 3×3 setting, although they still maintain higher accuracies compared to *Deep All* one. Regarding the second parameter, changing the permutation number to higher ($P = 50$ and $P = 100$) or lower ($P =$

DG - AlexNet					
Grid Size	**Art**	**Cartoon**	**Sketch**	**Photo**	**Avg.**
2	64.55	72.4	**66.39**	88.98	73.08
3	**68.60**	**71.29**	65.60	**89.60**	**73.77**
4	67.63	69.65	63.81	87.90	72.24

Table 2. Ablation results for the jigsaw puzzle task in DG. The table shows the accuracies obtained varying grid size parameter.

DG - AlexNet					
Num Perm	**Art**	**Cartoon**	**Sketch**	**Photo**	**Avg.**
5	68.60	70.05	64.52	88.80	72.99
10	66.45	69.15	62.92	**90.48**	72.25
30	**68.61**	**71.29**	**65.60**	89.60	**73.77**
50	66.45	70.61	62.68	90.12	72.46
100	65.62	70.35	63.68	89.82	72.36

Table 3. Ablation results for the jigsaw puzzle task in DG. The table shows the accuracies obtained varying number of permutation parameter.

5 and $P = 10$) values displays a situation similar to the previous one: the new results are lower compared to the original setting ones, but they are still higher than the *Deep All* baseline. All the accuracies obtained for the ablation of this parameter are reported in table 3.

4.2. Testing Other Self-Supervised Tasks

Using Rotation instead of Jigsaw Puzzle. After observing the improvements obtained by using the jigsaw task, we asked ourselves if similar results could be obtained with a simpler self-supervised task. We opted for a rotation one where the network has to recognize if an image has been rotated and with which angle. So, for this task, the angle constitutes the label and it can take on four different values. Following this, we disabled the patch grid system and rotated the full images. The experiments were conducted by maintaining a similar range of values used for the jigsaw task, with a grid search that involved $\alpha = \{0.3, 0.7\}$ and $\beta = \{0.2, 0.4\}$. The experiments reported significant improvements in results for DA, compared to the jigsaw task ones, achieving an average improvement of 3% over our jigsaw puzzle task and 2.5% over the original JiGen. Bottom of table 1 fifth row shows the best accuracy obtained with the best combination of hyperparameters, that is $\beta = 0.2$ and $\alpha = 0.7$. Moreover, this accuracy value is the best one obtained among all the experiments although none of the accuracies for the single domain is the best one when compared with those obtained with the support of other self-supervised tasks.

On the other hand, in DG context, the results obtained with the same range of values as DA for the hyperparameters grid search are only on par with *Deep All*. With the aim of trying to replicate the improvements obtained on DA also on DG, we hypothesized that the issue may reside in the value assigned to these parameters. Then we decided to start a fine-tuning process and broaden our range of values for the grid search. The new experiments however obtained results slightly worse than those of the jigsaw puzzle task but at better than the *Deep All*. The best accuracy reported in bottom of table 1, fifth row, is obtained with $\beta = 0.55$ $\alpha = 0.75$. Furthermore, it can be noticed how mid-to-low values of β tend to perform with better average accuracies.

Using Odd-One-Out instead of Jigsaw Puzzle. Another task compared to the jigsaw was Odd-One-Out. For its implementation, we have continued to use the 3×3 patch grid. The model causally mixes the image, choosing one of the permutations used in the case of the jigsaw task, and swaps one patch with another one taken from a different image. The source image of this belongs to the same domain and the same class as the image under analysis. The model's task is to determine the index of the external patch. For the experiments, we tested the same 2×2 grid search with $\alpha = \{0.3, 0.7\}$ and $\beta = \{0.2, 0.4\}$ used for the rotation task. The obtained DG results maintained a similar pattern to ones obtained on *Deep All*. It is interesting to notice an impressive drop that sketch shows compared to the jigsaw. In DA case the results are higher with a similar 2% increase against the original JiGen task and 2.5% increase to our jigsaw result. The best result is obtained for both DG and DA on $\beta = 0.2$ and $\alpha = 0.7$. This, combined with results obtained on the rotation task, may indicate that simpler tasks provide a better way for the network to extract hidden features from the target images when using domain adaptation, thus providing better results at test time.

Adding Style Transfer to Jigsaw Puzzle Task. The next step aims to observe what would happen if we modify the jigsaw task adding an increased random factor, instead of completely substitute it with another task. To do that, we used the style transfer technique, described in [12], which consists of taking the style-features from a given set of images and applying them to a different one. Translating this to our approach, led us to two ideas: on one side, in DG context, to re-distribute the style features of multiple source domains to the source training images could help the network to get a more generalized learning process. On the other side, in DA context, applying the target domain style-features to the source images could help the network in obtaining more indirect information regarding it. In order to do this, we first took an encoder and a decoder, both pretrained on ImageNet, and we fine-tuned them on a given set of PACS images. In DG we fine-tuned a model for each combination of 3 domains, since the target one is not available at training time, while in DA we trained a single model that exploits the images of all PACS domains. The parameters of AdaIN used for this fine-tuning phase were the default ones.

Then, our network used these models to apply the style

transfer to a random set of our training images in the following way: each training image, during the jigsaw puzzle task pre-processing step, can be randomly shuffled (depending on β_{jigsaw}) and when it happens a random style based on another domain is applied to each patch. The label for this self-supervised task is still the permutation label. Eventually, all images go again through the jigsaw classification task, but only the non-shuffled ones, which also avoid the style transfer are sent to the main classification problem.

In DG context, we conducted a 2×2 grid search with $\alpha = \{0.3, 0.7\}$ and $\beta = \{0.2, 0.4\}$: the obtained results are heterogeneous, but they never reach the base jigsaw task. The best results, shown in top of table 1, sixth row, is obtained with $\alpha = 0.7$ and $\beta = 0.2$. In particular, such as in the Odd-One-Out case, sketch seems to be the domain that suffers the most, compared to the jigsaw.

In DA, we applied only the target domain style to all the images reserved for the self-supervised task. We used the same range of values for the grid search and all the simulations show almost identical results, never reaching the jigsaw puzzle ones but still higher than *Deep All*. The best accuracy computed is shown in table 4, first row, and it was retrieved with $\alpha = 0.3$ and $\beta = 0.4$. However, it is interesting to note how, in contrast with sketch and cartoon which suffered a heavy performance blow, photo obtained a significant improvement compared to the original jigsaw task, which could depend by the style encoder and decoder pretraining on ImageNet.

Lastly, we decided to merge the two approaches and to repeat the DA experiments. This means we used the style of every domain and we applied them to both source and target training images. The results, in this case, are better than the ones previously obtained with the usage of only the target domain. The best result is shown in bottom of Table 1, sixth row, and in Table 4, second row, and it was obtained with $\alpha = 0.7$ and $\beta = 0.2$. Table 1 underlines that sketch is the only domain in which accuracy is worse compared to the one obtained for the original jigsaw puzzle task, though with less intense effect with respect to the other tasks. The fact that this second approach performed better than the first one was unexpected. Intuitively, we are led to think that applying the style of target images, available at training time in DA, might be the main cause of an improvement in performance. Unfortunately, in doing so we apply only one style to the train images, instead of 4 different ones as happens following the second approach described. Hence, the results obtained can be motivated by saying that using multiple styles helps to generalize the set of train images better, and this has proven to be a better factor than the availability of the target style alone.

DA - ResNet 18					
Style	**Art**	**Cartoon**	**Sketch**	**Photo**	**Avg.**
Target domain	83.66	79.44	70.14	97.18	82.60
All domains	**84.77**	**80.84**	**73.40**	**97.25**	**84.60**

Table 4. Comparison between results of jigsaw puzzle task in multi-source Domain Adaptation, obtained using as style to transfer only the target one or a random one from all the domains.

4.3. Combining multiple Self-Supervised Tasks

After testing every task on its own, we decided to combine them. The tested combinations are a mix of the jigsaw puzzle, rotation and odd-one-out tasks, both in couples and all three combined. For each set of experiments, depending on the activated tasks, we set their respective β parameter to the same value, keeping $\alpha = 0.7$. Since in our work different tasks use a disjointed set of images, the tested β were kept low, in the range $[0.1, 0.3]$ for the 2-task combinations and in the range of $[0.1, 0.2]$ for the 3-task one, in order to still maintain a sufficiently good amount of original images for the main classification problem. Regarding DG, almost all the combinations provided better results than single self-supervised task usage. The best combination is the one that combined rotation and jigsaw tasks, assigning $\beta = 0.1$ for both tasks. Top of table 1 shows the results. In DA case, in which results are reported in bottom of the same table, all task combinations provided better results compared to the Deep All baseline and most of them even surpassed the normal jigsaw puzzle task. The best combination was the one that combines all the three tasks, again using $\beta = 0.1$ for all of them.

5. Conclusion

The approach described in this paper proposes a solution to the problem of classification across domains. The intuition of using a self-supervised task to support the classification, already known in the literature, has been extensively tested by trying various types of different self-supervised tasks. This approach has been applied in two different contexts, DG and DA. In the first case, the various tasks analyzed have a heterogeneous behavior. The jigsaw puzzle task turns out to be the best, replicating the scores already obtained in other papers. In particular, the addition of the style transfer to this type of task does not bring any improvement. The rotation allows increasing the accuracy in ways equivalent to the jigsaw. The odd-one-out does not seem to affect the classification results, which remain at the baseline level. In the case of the DA, all the tasks exceed the baseline value, in particular, rotation and odd-one-out overcome the jigsaw puzzle, obtaining better results. Eventually, when looking at the combination of multiple tasks, we obtain good results in almost all the cases, even if a low level of fine-tuning is performed. A better fine-tuning phase

could increase the results even over the best ones, retrieved with a single self-supervised task, because the more task we exploit the more the model should be able to generalize.

On a final note, to improve the results obtained it could be interesting to try others self-supervised tasks not mentioned in this work or to combine our approach with another existing one, such as DANN [8].

References

[1] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. 2, 3, 4, 5

[2] T. S. Cho, S. Avidan, and W. T. Freeman. The patch transform. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (32):1489–1501, 2009. 2

[3] R. S. Cruz, B. Fernando, A. Cherian, and S. Gould. Visual permutation learning. *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[4] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *SIGGRAPH*, 2001. 2

[5] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. 2

[6] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: example-based adaptive patch sampling for unsupervised style transfer. *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[7] A. C. Gallagher. Jigsaw puzzles with pieces of unknown orientation. *Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 8

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[10] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. 2

[11] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. *SIGGRAPH*, 1995. 2

[12] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. 2, 3, 6

[13] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European Conference on Computer Vision (ECCV)*, 2016. 2

[14] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. *European Conference on Computer Vision (ECCV)*, 2012. 2

[15] D. Li, Y. Yang, Y.-Z. Song, and T. M.Hospedales. Deeper, broader and artier domain generalization. *International Conference on Computer Vision (ICCV)*, 2017. 2, 4

[16] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. *International Conference on Computer Vision (ICCV)*, 2017. 2

[17] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *European Conference on Computer Vision (ECCV)*, 2016. 2, 4

[18] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. Boosting self-supervised learning via knowledge transfer. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[19] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. *In International Conference on Learning Representations (ICLR)*, 2018. 2

[20] D. Sholomon, O. E. David, and N. S. Netanyahu. A generalized genetic algorithm-based solver for very large jigsaw puzzles of complex types. *Conference on Artificial Intelligence (AAAI)*, 2014. 2

[21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, , and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *International Conference on Intelligent Robots and Systems (IROS)*, 2017. 2