Self supervision for Domain Generalization and Adaptation

Michele Filippini, Pier Giorgio Mingoia, Pietro Recalcati and Enrico Tortasso
Politecnico di Torino
{s275902, s276390, s276963, s277950}@studenti.polito.it

Abstract

The ability to apply previous knowledge to new topics is typical of our species and in the past it was impossible to think that a machine could replicate it. The first generation of neural networks relied on low-level features for the recognition of images, producing models with no ability to adapt to new domains. JiGen tried to solve this issue by pairing a supervised task like classification with a self supervised one, in order to force the network to learn to recognize higher level patterns. We tried to improve this solution by using style transfer on the input images and by substituting the jigsaw task with rotation recognition. The vast majority of our tests showed similar or better results with respect to the original version of JiGen.

1. Introduction

Nowadays, the application of deep learning models to real world problems in different contexts is becoming more and more pervasive. The amount of information collection systems around us is constantly increasing, producing huge quantities of data which, although similar in content, are diverse in appearance. As an example, a picture of an object taken during the day is different from one taken during nighttime. To fully exploit all the available data and to be able to better perform in unknown conditions it is necessary to modify existing deep models. We added a pretext self-supervised task to an image classifier in order to use information from unlabelled samples to improve the generalization ability of our network across domains.

2. Related work

Domain Generalization and Adaptation While Deep Neural Networks have been shown to deliver very good results when applied to object recognition tasks on a given homogeneous data set, they are usually not as performing when dealing with styles and distributions which are different from the ones they have been trained on. *Domain Adaptation (DA)* can be defined as the setting in which a model trained on a *source* domain is developed to correctly classify objects coming from a different one, referred to as *target*. In the most common scenario a small set of unlabeled target samples is available at training time, while in the context of *Domain Generalization (DG)* the goal is to be able to learn a model which can be applied to any unknown target.

Several strategies have been proposed to tackle these problems, mainly devoted to reduce the gap between different distributions in order to consequently decrease the effort required by the domain shift. As an example, a simple shallow learning technique aims at minimizing a distance measure proposed in [6] between the projections of samples belonging to different domains in a new feature space. Similarly, the most common deep learning solution ([4]) involves the addition to the object classifier of a domain discriminator, which is trained in an adversarial fashion by reversing its gradients before updating the weights.

Self-supervised learning Supervised learning is one of the most diffused task in machine learning works but it shows several disadvantages related to the data labeling. Successful data labeling is a workflow challenge: the need to manage enough workers to process the large volume of unstructured data and the need to ensure high quality across such a large workforce.

In the Domain Adaptation setting it is fairly common for target training samples to be unlabelled, causing the need for a network able to learn without human supervision. The main idea behind self-supervised learning is to introduce some *perturbation* in the unlabeled images and ask the network to detect or correct it.

Existing solutions range from regeneration of missing image portions [15], grayscale image colourization [11], relative patch location [3] to jigsaw puzzles solution [14] and many more. All of these tasks are helpful in learning visual representations and can be exploited to improve classification performance or to train a network when a limited amount of labeled samples are available. A common practice is to train the model on the self-supervised task and

later substitute (or add) a final layer to perform classification, either fine-tuning the previous ones for the new task or keeping them frozen. A different approach involves multi-task learning [2]: the self supervised task and the classifier share a network portion and are trained at the same time, allowing each task to act as a regularizer for the other one. Differently from the fine-tuning solution, the latter prevents the network from forgetting what it had previously learned from the supporting task and encourages each layer to focus on a single task.

Solving a *jigsaw puzzle* is a simple yet effective way of providing self-supervision in the domain adaptation setting: spatial location of patches is part of the information contained in every image and does not require humans to manually provide labels. Furthermore, a strong knowledge of the relative position of object parts is needed in order to successfully fulfil the task. This mitigates the tendency of networks to focus on very few features during classification, therefore improving the ability to generalize across different domains.

Noroozi and Favaro ([14]) defined the task as finding the permutation corresponding to the order of tiles in a scrambled image (*Figure* 1). In their work they show how applying transfer learning from a network trained on jigsaw puzzles improves the performance of a standard AlexNet on ImageNet classification. Following the strategy proposed by Carlucci *et al*. in [1], we verified that using jigsaw puzzles as an auxiliary task at training time leads to improvements in the adaptation and generalization abilities of a classifier.

Gidaris *et al*. ([5]) proposed *rotation recognition* as a new unsupervised task useful to learn image features in ConvNets. Training images are rotated by a random integer multiple of 90°(0°, 90°, 180°, 270°), which is treated as a label, and the network is required to predict the correct angle. Similarly to jigsaw puzzles solution and relative position recognition, this forces the model to learn details about the object structure and shape while keeping the self-supervised task very simple.

3. Method

In this section we want to introduce the characteristics of the JiGen network and the modifications applied to it to better understand the results shown in the experiments section.

JiGen Network JiGen can be described as an end-to-end architecture that learns simultaneously how to generalize across domains and about spatial co-location of image parts. It implements the concept of multitask learning to reach the final result of being able to generalize across different domains.

Starting from [14] as described in [1] JiGen moves the patch re-assembly at the image level and formalizes the jigsaw task as a classification problem over recomposed images with the same dimension as the original one. In this way object recognition and patch reordering can share the same network backbone. JiGen is composed of three different parts: there is a convolutional network which is the backbone and there are two fully connected layers, one related to object recognition and the other one related to patch reordering. In DG settings the chosen architecture is AlexNet, in DA it is ResNet-18.

Figure 1: Example of a normal image and its jigsaw puzzle transformation.

Given an image x as input to the network the first basic objective of JiGen to solve classification task is to minimize the loss formula $\mathcal{L}_c(h(x|\theta_f, \theta_c), y)$ that measures the error between the true label y and the label predicted by the deep model function h parametrized through θ_f and θ_c. These parameters define the feature embedding space and the final classifier, respectively for the convolutional and the fully connected parts of the network. The second basic objective of JiGen is related to the jigsaw task solution. To train the network to be able to solve this kind of task it also receives as input some images decomposed using a regular $n \times n$ grid of patches, which are then shuffled and re-assigned to one of the n^2 grid positions. Out of the all possible permutations only a subset P of them are finally selected, following the Hamming distance based algorithm ([14]). Considering a shuffled image z and the related permutation index p , the network second basic objective is to minimize the jigsaw loss $\mathcal{L}_p(h(z|\theta_f, \theta_p), p)$. The deep model function has the same structure of the one presented before and they share the parameters θ_f. The final fully connected layer dedicated to permutations recognition is instead different and is parametrized by θ_p. Considering images belonging to S different domains, with each domain containing N_i different images, where x_j^i indicates the $j\text{-}th$ image from the $i\text{-}th$ domain and y_j^i its class label. Considering also in each domain K_i shuffled images, where z_k^i indicates the shuffled sample and p_k^i the related permutation index, we trained the network to obtain the optimal model which is able to satisfy this condition:

$$\arg \min_{\theta_f, \theta_c, \theta_p} \sum_{i=1}^{S} \sum_{j=1}^{N_i} \mathcal{L}_c(h(x_j^i|\theta_f, \theta_c), y_j^i) +$$

$$\sum_{k=1}^{K_i} \alpha \mathcal{L}_p(h(z_k^i|\theta_f, \theta_p), p_k^i)$$

Both the losses described before, $\mathcal{L}_p$ and $\mathcal{L}_c$, are defined as standard cross entropy loss. The classification loss is not influenced by the shuffled images, since it would make object recognition tougher. The jigsaw loss is instead calculated on all the images, since an ordered image can be seen as the result of a shuffle with all the patches positioned in the correct order. At test time only the object classifier is used to make predictions on new target images.

JiGen applied to Unsupervised Domain Adaptation We made experiments regarding both DA and DG condition. As described before, Domain Adaptation is different from Domain Generalization since unlabeled target images are available at training time. Thanks to the unsupervised nature of the jigsaw puzzle task we can extend it to new target images. In this settings for the target images following [1] we minimized the classifier prediction uncertainty through the empirical entropy loss

$$\mathcal{L}_E(x^t) = \sum_{y \in \mathcal{Y}} h(x^t|\theta_f, \theta_c) log\{h(x^t|\theta_f, \theta_c)\}$$

while for the shuffled target images we continued to optimize the jigsaw loss $\mathcal{L}_p$.

Implementation details JiGen network [1] presents two parameters related to how we defined the jigsaw task and three related to the learning process. The first two are respectively the size of the grid $n \times n$ used to decompose and create shuffled images and the cardinality of the patch permutations subset P. We decided to keep this two parameters fixed, 3×3 for grid size and $P = 30$, for the majority of experiments, as declared in the JiGen paper ([1]), but we also modified them to understand how network performance changes based on different values.

The remaining parameters are the weights α and η, related to the jigsaw loss and the entropy loss when included in the optimization process for unsupervised domain adaptation. The final parameter related to learning process is β, that corresponds to the percentage of ordered images in each image batch. For example $\beta = 0.6$, means that for each batch 60% of the images are ordered, while the remaining 40% are shuffled. During our experiments, α and β parameters where chosen by cross-validation on a 10% of subset of the source images. In Domain Adaptation settings we assumed the η parameter fixed to the value of 0.1.

Our JiGen model is trained with SGD solver, 30 epochs, batch size 128, learning rate set to 0.001 and stepped down to 0.0001 after 80% of the training epochs. We used simple data augmentation protocol by randomly cropping the images to retain between 80-100$\%$ and randomly applied horizontal flipping. Following [13] we converted image tiles to grayscale with fixed 10% probability and we applied random crop to each tile as suggested in [14]

[1] Code available at https://github.com/PierGiorgioMingoia/MLAI_project

4. Proposed variations

Rotation task

Figure 2: Example of the 180° rotation.

Rotation is the first variation we made. What we wanted to study through this variation is how the performances change when instead of jigsaw we use rotation as self supervised task. The second objective of the network, in this setting, is no longer to minimize the jigsaw loss as described before but to minimize the rotation loss $\mathcal{L}_r(h(x|\theta_f, \theta_r), y)$. The rotation is implemented as a classification task, as we did for the jigsaw. The possible rotations of images are 0°, 90°, 180°, 270°. An example can be seen in *Figure* 2. The rotated images, as well as the not rotated ones, are associated to a specific label y which represent the rotation applied.

AdaIN Network

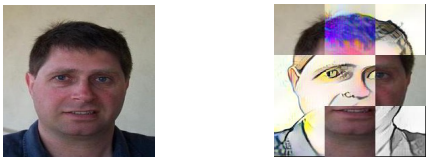

Figure 3: Example of jigsaw puzzle composed by different styles of the same images using AdaIN

The second variation we proposed is to use the AdaIN network during the decomposition of the images in different tiles. AdaIN network is a very powerful model able to perform style transfer in real time. This network is composed of the first few layers of a fixed VGG-19 network to encode the content and style images. An AdaIN layer is used to perform style transfer in the feature space. A decoder is learned to invert the AdaIN output to the image spaces [8]. As we described before and as we are going to demonstrate in following sections JiGen network gives very good results in image classification also when images belong to different domains. To reinforce the learning procedure of the network we decided to exploit the AdaIN model in or-

der to apply domain shifting to each single tile of the image. At the end of this procedure we obtained an image which is composed of different tiles each of these belonging to a random domain between the input ones as shown in *Figure* 3. Since we made experiments both in DG and DA settings, we developed five different pre-trained models of AdaIN to obtain more precision in domain shifted images.

5. Experiments

We analyzed the behaviour of the the JiGen neural network described before when varying the hyperparameters α (weight of the jigsaw loss classifier) and β (the percentage of non shuffled images). We also tuned α_t (weight of the jigsaw loss for the target domain) in the Domain Adaptation setting.

For the experiments related to Domain Generalization we used AlexNet [9] (8 layers), while for Domain Adaptation the adopted network was ResNet-18 [7]. Both of them were pre-trained on the ImageNet dataset as in [16], which proved this to result in a better overall performance.

Our starting point was a single task classifier, denoted as Deep All, that classifies images from different domains. We then modified it to perform a jigsaw task, making it a multi-tasks classifier. The network performs differently on the various domains. For example Photo is the most accurate one because the models we use are pretrained on ImageNet. Our goal is to obtain better results with respect to Deep All.

5.1. Dataset

Figure 4: Examples from PACS dataset.

We used PACS dataset [10]. It contains 9991 images distributed across 4 domains (Photo, Art Painting, Cartoon and Sketches as in *Figure* 4) and grouped in 7 different classes (dog, person, giraffe, elephant, house, horse and guitar). This dataset was proved to have larger domain shift than others [12].

5.2. JiGen Domain Generalization

We performed a grid search (*Figure* 5) in order to find the couples of (α, β) that provide the best average accuracy in validation across all target domains. After the first run we noticed that the margin values of 0.1,0.2 and 1.0 could be avoided since they always lead to bad results. We took the best 2 couples, we carried on further experiments to compare them and we identified the best one as (0.6,0.6), that performed better on tests.

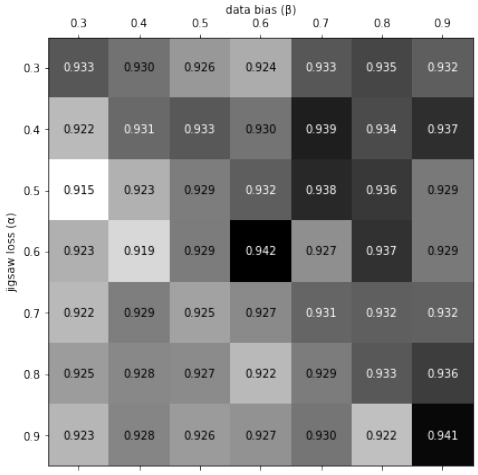

Figure 5: Grid search for JiGen Domain Generalization tuning.

After the choice of the best parameters we decided to evaluate our network compared to Deep All and [1]. As seen in *Table* 1, we obtained results comparable to the ones of previous works. This proves that the jigsaw task leads to overall better results with respect to the Deep All.

JiGen-DG		Art_paint.	Cartoon	Sketch	Photo	Avg.
AlexNet						
[1]	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	64.79	70.64	60.50	**89.26**	71.30
	JiGen	**67.17**	**71.72**	**63.99**	88.84	**72.93**

Table 1: Multi-source Domain Generalization results on PACS obtained as the average over three repetitions for each run. We used $\alpha = 0.6$ and $\beta = 0.6$.

Ablation study As we can see in *Figure* 6, varying β has a great impact on the overall performance of our network. For low values we obtained results that are worse when compared to the Deep All. This is because when β is small we pass to the network a high number of puzzle images, making the network focus on the jigsaw classification even though the main objective is still class recognition.

In the second graph instead we can see that for mid-high values of β the choice of a small value for α leads to an overall loss in performance. This is likely due to feeding the network with a relatively big quantity of puzzle images while not giving an appropriate weight to the jigsaw loss. This makes the self supervised task lose some relevance, leading to a loss in the overall accuracy.

We also decided to treat the number of tiles the images were divided in as another hyperparameterer. We wanted to see how the network performed with different tiles numbers

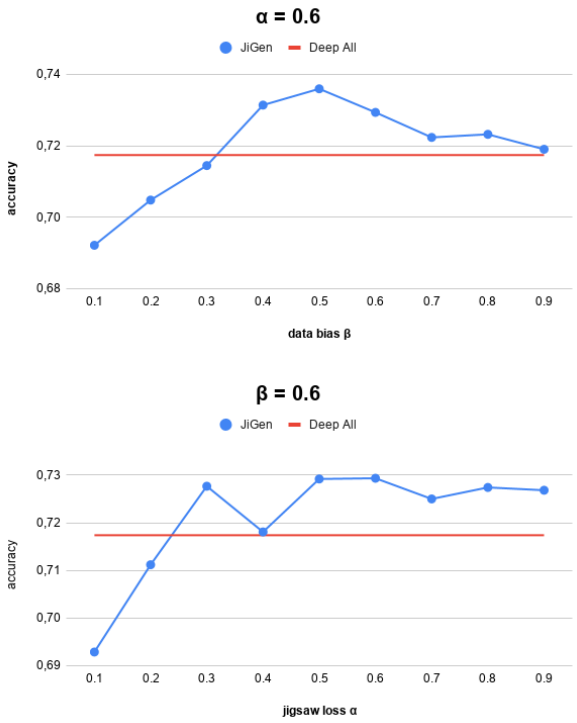

Figure 6: These graphs show the results from the ablation on α and β in JiGen Domain Generalization. The red line is the Deep all accuracy while the blue line is the global average accuracy over all target domains obtained varying only one of the hyperparameters at a time.

JiGen-DG	Art_paint.	Cartoon	Sketch	Photo	Avg.
AlexNet					
Deep All	64.79	70.64	60.50	89.26	71.30
2x2	**68.75**	70.64	62.44	88.76	72.65
3x3	67.17	**71.72**	**63.99**	88.84	**72.93**
4x4	66.55	70.63	62.50	**89.50**	72.29

Table 2: Multi-source Domain Generalization results on PACS varying the number of tiles of an image, obtained as the average over three repetitions for each run.

(*Figure* 7). For the 2×2 case we had a number of possible permutations equals to $4! = 24$ while for 3×3 and 4×4 we decided to limit the number to 30. For the last 2 cases we used the Hamming distance as a method to find the most diverse permutations. The best performing was 3×3, as shown in *Table* 2.

5.3. JiGen Domain Adaptation

We repeated the same hyperparameter tuning procedure used for Domain Generalization, while keeping $\alpha_t = 0.7$ as shown in *Figure* 8. This value was taken from [1] and used

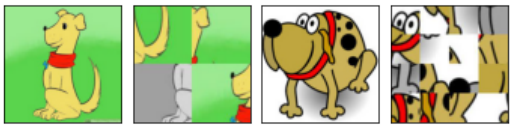

Figure 7: Examples of 2x2 and 4x4 jigsaw puzzles.

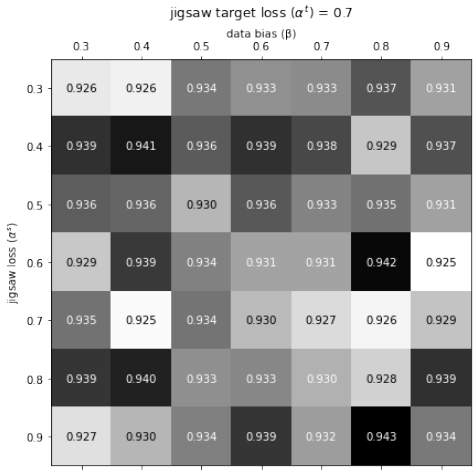

Figure 8: Grid search for JiGen Domain Adaptation tuning.

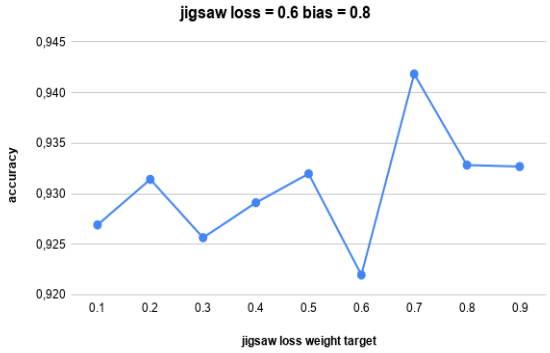

Figure 9: Best validation for different values of α_t for Domain Adaptations.

JiGen-DA		Art_paint.	Cartoon	Sketch	Photo	Avg.
ResNet-18						
[1]	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	79.83	75.30	67.81	95.86	79.70
	JiGen	**85.28**	**81.72**	**78.14**	**98.48**	**85.91**

Table 3: Multi-source Domain Adaptation results on PACS obtained as the average over three repetitions for each run. We used $\alpha_s = 0.6$ and $\beta = 0.8$ and $\alpha_t = 0.7$.

as a starting point. We then selected the best couple (α_s, β) and then looked for the best value of α_t as shown in *Figure* 9. We then settled for (α_s, β, α_t) equal to (0.6,0.8,0.7).
We then compared again our results with the ones obtained in [1], as seen in *Table* 3. The increase in performance was more relevant with respect to the Domain Generalization case.

Ablation study As in the DG setting, the variation of beta is the one that causes the biggest fluctuation in performance (see *Figure* 10). Choosing a low value for β means feeding too many puzzle images to the network, resulting in a worse overall performance on the classification task. On the contrary we see that as long as at least one between α_s and α_t is chosen to be a significant value, the variation of the other carries little difference to the results. In both cases, however, we can see a tendency to a worse accuracy when α_s or α_t grows towards big values. This is probably due to putting too much weight on the jigsaw loss, thus worsening the classification task performance.

As already done in Domain Generalization we analyzed how the number of tiles affects the performances of the network. As shown in *Table* 4 we obtained the best results for 4×4 setting.

JiGen-DA	Art_paint.	Cartoon	Sketch	Photo	Avg.
ResNet-18					
Deep All	79.83	75.30	67.81	95.86	79.70
2x2	85.09	81.66	75.66	98.16	85.14
3x3	**85.28**	81.72	74.14	**98.48**	85.91
4x4	84.34	**82.19**	**79.14**	98.46	**86.03**

Table 4: Multi-source Domain Adaptation results on PACS varying the number of tiles of an image, obtained as the average over three repetitions for each run.

5.4. Rotation

In the attempt to find an alternative to JiGen that was still capable of bringing good results we implemented the rotation self supervised task as described in the above section. As done with JiGen network we studied the effects of this variation both in Domain Adaptation and Generalization settings. Both in DA and DG we used the same approach as with JiGen in the hyperparameter tuning, picking the 2 best couples for each setting that give the best validation values as shown in Figure 11 (for DG) and in Figure 12 (for DA). Between the two couples we chose the one that, considering three runs, gave the best average test results across all different domains. Following this procedure the couple selected for DG is (α, β) equals to (0.4,0.9) and for DA (α_s, β, α_t) equal to (0.9,0.9,0.7). As shown in *Table* 5, results obtained in Domain Generalization setting are slightly better than the ones obtained through the JiGen network, while those obtained in Domain Adaptation are strongly better than JiGen.

Rotation	Art_paint.	Cartoon	Sketch	Photo	Avg.
AlexNet					
Deep All	64.79	70.64	60.50	89.26	71.30
JiGen	67.17	**71.72**	**63.99**	88.84	72.93
RotDG	**69.07**	70.50	63.06	**89.60**	**73.06**
ResNet-18					
Deep All	79.83	75.30	67.81	95.86	79.70
JiGen	85.28	**81.72**	78.14	**98.48**	85.91
RotDA	**88.98**	81.62	**81.98**	98.18	**87.69**

Table 5: Multi-source Domain Generalization $\alpha = 0.4$ and $\beta = 0.9$, Domain Adaptation $\alpha_s = 0.9$ and $\beta = 0.9$ and $\alpha_t = 0.7$ results on PACS comparing JiGen and rotation task, obtained as the average over three repetitions for each run.

Ablation study As shown in the first graph of *Figure* 15, in the DG setting, keeping fixed value of α for very low value of β we obtain lower or similar performances with respect to Deep All. Increasing the number of straight images given as input to the network while having small value of α seems to stabilize the accuracy over the deep all value.

In the second graph instead we observe a decreasing behaviour in results varying values of α still remaining above Deep All performance. The parameter β fixed to 0.9 means that the network is mainly focused on the classification task since the majority of images given as input are straight. Since few images are rotated, the rotation task does not provide a highly reliable contribution to the training of the network. For this reason we can observe that high values of α lead to performances below the overall mean.

In the Domain Adaptation setting, as seen in *Figure* 14, the behaviour obtained by varying each of the hyperparameters is better than Deep All and quite stable throughout the whole ablation process.

5.5. AdaIN

The second variation we proposed is the insertion of the AdaIN network in the tile-decomposition of images. We trained the AdaIN network on different subsets of the PACS dataset to obtain the five models needed to run our experiments. All the results of this version are reported in (*Table* 6). As we did with the Rotation task we evaluated the performances of this kind of variation both in DA and DG settings. Since in Domain Adaptation both source and target images are used in the training phase of the network we decided to use an AdaIN model trained on all images of PACS. We made this choice with the objective of having a model able to perform style transfer between all different domains.

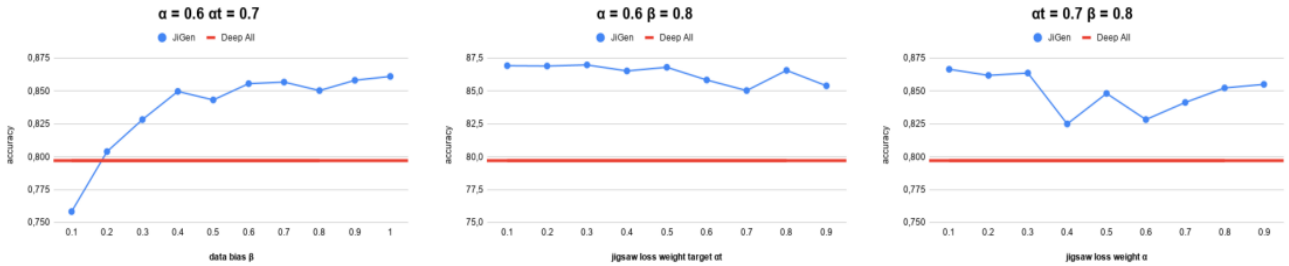

Figure 10: These graphs shows the results from the ablation on α_s, β and α_t in JiGen Domain Adaptation. The red line is the Deep All accuracy while the blue line is the global average accuracy over all target domains obtained varying only one of the hyperparameters at a time.

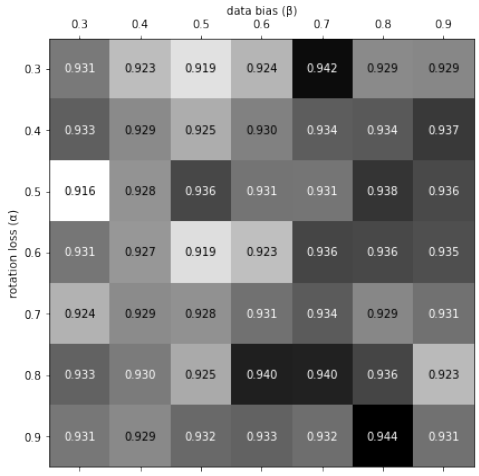

Figure 11: Grid search for Rotation Domain Generalization tuning.

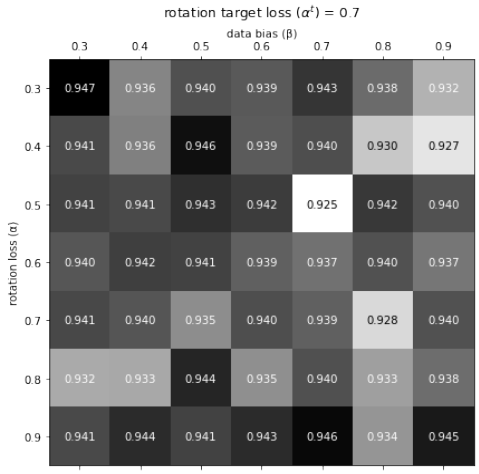

Figure 12: Grid search for Rotation Domain Adaptation tuning.

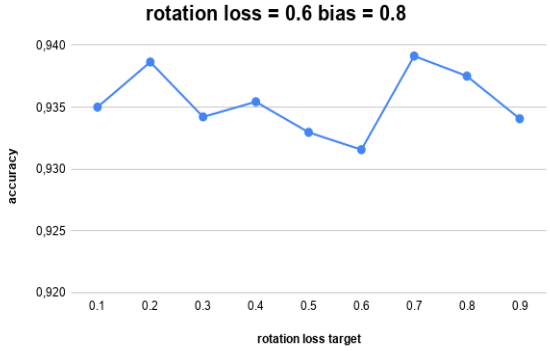

Figure 13: Best validation for different values of α_t in Rotation Domain Adaptation.

AdaIN	Art_paint.	Cartoon	Sketch	Photo	Avg.
		AlexNet			
Deep All	64.79	70.64	60.50	**89.26**	71.30
JiGen	67.17	**71.72**	63.99	88.84	72.93
JiGen-AdaIN	**69.61**	71.31	**69.38**	89.12	**74.85**
		ResNet-18			
Deep All	79.83	75.30	67.81	95.86	79.70
JiGen	**85.28**	**81.72**	78.14	**98.48**	85.91
JiGen-AdaIN	85.25	79.60	**86.57**	96.42	**86,96**
JiGen-AdaIN*	83.69	79.44	85.13	95.86	86.03

Table 6: Multi-source Domain Generalization and Domain Adaptation results on PACS obtained as the average over three repetitions for each run comparing performance using JiGen and JiGen+AdaIN. For DG we used the couple (0.6,0.6) as hyperparameters, while for DA we used (0.6,0.8,0.7). * means that the AdaIN model is trained on source domains only.

For Domain Generalization, instead, we worked with four different models as the number of domains of our dataset. Each model is trained on all domains except for the target one since in DG setting only source images

Figure 14: These graphs shows the results from the ablation on α_s, β and α_t in Rotation Domain Adaptation. The red line is the Deep All accuracy while the blue line is the global average accuracy over all target domains obtained varying only one of the hyperparameters at a time.

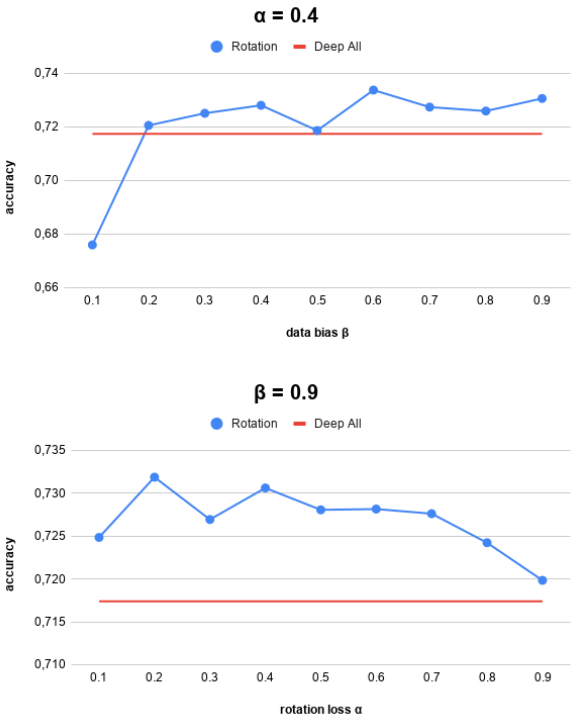

Figure 15: These graphs shows the results from the ablation on α and β in rotation Domain Generalization. The red line is the Deep all accuracy while the blue line is the global average accuracy over all target domains obtained varying only one of the hyperparameters at a time.

are used during training. To better understand how the performances of the network would change with different models we also decided to report the performances of our model in DA setting using AdaIN models of DG setting.

Results In both cases (DA and DG) the greatest improvement in performance was obtained for the sketch domain, probably due to the fact that AdaIN changes the colour and

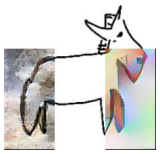

Figure 16: Example of an image of a sketched horse created using AdaIN

the style of the image, leaving only the borders untouched, as seen in *Figure*16. This implicitly improves sketch performances since borders are the most relevant feature used in the classification task. As expected, when the AdaIN network is trained only on source domains the resulting DA model is not as performing as the one in which AdaIN was trained on target as well.

6. Conclusion

In this paper, starting from the existing JiGen network we explored different variations to the jigsaw puzzle task to see how the deep model behaved. As we showed in the experiments section above, we found out that the rotation task is a good alternative to the jigsaw. In AdaIN we followed a different approach: we didn't change the overall structure of the JiGen network but we added a different kind of data augmentation through the AdaIN model and we showed that it improves the overall performances of the network both in Domain Adaptation and Domain Generalization settings. Our work shows that image transformation techniques are useful in preventing the network from focusing on low level features only, therefore helping the model in the generalization process.

References

[1] F. M. Carlucci et al. Domain generalization by solving jigsaw puzzles. *CVPR*, 2019.

[2] R. Caruana. Multitask learning. *Machine learning*, 1997.

[3] C. Doersch et al. Unsupervised visual representation learning by context prediction. *ICCV*, 2015.

[4] Y. Ganin et al. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 2016.

[5] S. Gidaris et al. Unsupervised representation learning by predicting image rotations. *ICLR*, 2018.

[6] A. Gretton et al. A kernel method for the two-sample-problem. 2007.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[10] D. L and Y. Yang. Domain Generalization (PACS). `https://domaingeneralization.github.io//`, 2017.

[11] G. Larsson et al. Learning representations for automatic colorization. *ECCV*, 2016.

[12] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *ICCV 2017*, 2017.

[13] M. Noroozi et al. Boosting self-supervised learning via knowledge transfer. In *CVPR*, 2018.

[14] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, 2016.

[15] D. Pathak et al. Context encoders: feature learning by inpainting. *CVPR*, 2016.

[16] M. Simon, E. Rodner, and J. Denzler. Imagenet pre-trained models with batch normalization, 2016.