1.Introduction

The focus of this assignment is to deal with domain adaptation problem scenario, consisting in a ordinary classification problem of images but belonging to different domains. In particular, a model trained with images of a specific distribution is evaluated in the context of a different (but related) target distribution, and, if the test data differ significantly from the training one, it may fall into bad performances. In general the level of relatedness between source (training data) and target (test data) domains determines how successful the adaptation will be. In other word, during the training phase is not possible to access the domain of the target and it can lead to bad performances of the net. So, the purpose of this assignment is to dive into this kind of problem.

In particular, for this task PACS dataset is used: it consists of *Art Painting, Cartoon, Photo and Sketch* domains, each containing the same seven classes (for a total of 9991 images). The original size of each image is 227x227 with 3 channels (i.e, RGB) and the distribution of the classes across the domains are the following:

	Dog (0)	Elephant (1)	Giraffe(2)	Guitar(3)	Horse(4)	House(5)	Person(6)	Tot
Sketch	772	740	753	608	816	80	160	3929
Art Painting	379	255	285	184	201	295	449	2048
Cartoon	389	457	346	135	324	288	405	2344
Photo	189	202	182	186	199	280	432	1670

Table 1: number of images in each domain

It is noticeable how the dataset is quite unbalanced not only across different domains, but also across different internal classes, and It makes some of them more recognizable than others. In particular, for this assignment the source domain is ‘Photo’ and the target is ‘Art painting’. The source domain dataset counts less samples than the target (in general it is the smallest one, according to the table above), and this is a further point that may affect the performances.

The domains differences of the pictures are, instead, shown with the images below, by extracting an image of the same class from each different set:

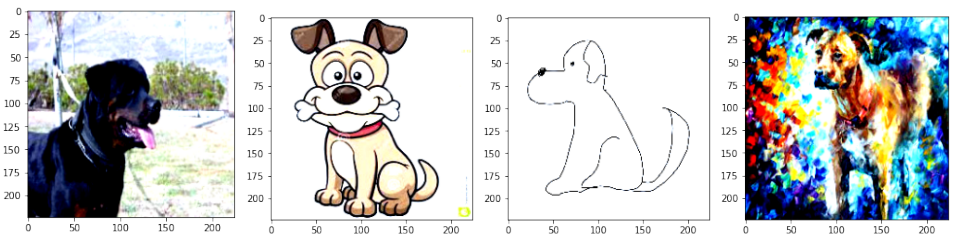


Figure 1: images from PACS dataset, one for each domain. Respectively: Photo, Cartoon, Sketch and Art Painting

2. Classification with no adaptation technique

As first step, a simple classification task is performed by training a traditional AlexNet with Photo dataset and then testing it on Art Painting one, to have a general idea about the performances without using any adaptation techniques. Before feeding the net, the datasets are preprocessed: pictures are center-cropped for a final 224x224 square image, to fit the expected input size of AlexNet, and the channels of each images are normalized to have mean equal to 0 and standard deviation to 1, in order to achieve better performances: in fact, convolutional neural networks in general are less robust and perform worse with non centered datasets.

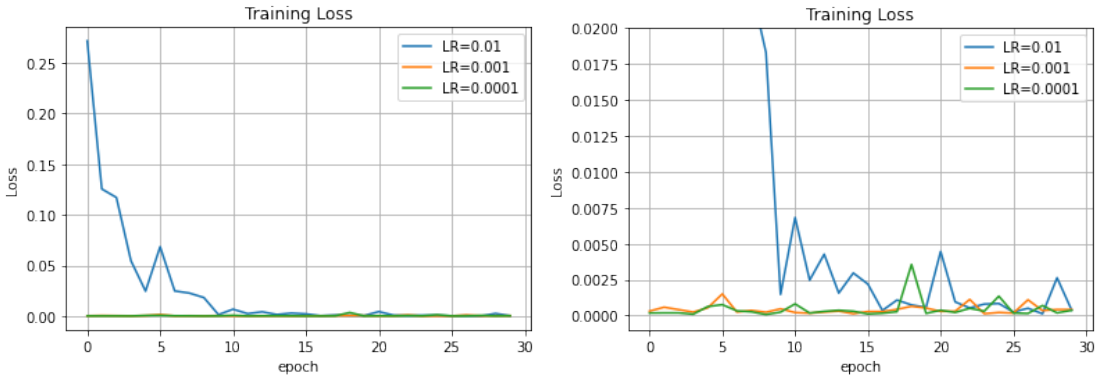
In addition, the last fully connected layer of the net is reformed to fit the output classes of the datasets used, seven.

Before proceeding with the first step, it may be interesting to check the behavior of the net with datasets belonging to the same distribution: in order to do this the net is trained with a larger part of photo dataset and tested on the remaining smaller part(167 samples). This split is executed in a random stratified fashion in order to preserve enough training samples for each domains and sufficient test samples for each label. The results are collected in the following table:

Learning Rate	Accuracy
0.01	97.3
0.001	96.1
0.0001	96.4

Table 2: Test accuracy for each learning rate

The training losses trend,instead, is shown in the following charts (the same figure is reported in two different charts with different scales in order to check the trend around zero):



Only learning rates were changed in order to achieve these performances: weight decay,epochs, the step size and the batch size were left fixed respectively to 5e-5, 30, 20 and 258, because the purpose of this step has been to give a general idea of the range of values got by evaluating the model on a test set with the same domain of the

training data, and not to find the combination of hyperparameters performing best. According to the previous representation, in fact, it is noticeable the high performances the model reached. The accuracies are over 95% and the losses trend really close to zero (no possible overfitting scenarios were taken into account here).

Then, the model is finally trained with the entire Photo dataset and tested on Art Painting one: different hyperparameters are tuned in order to find the configuration giving as good results as possible. In the following table the performances of each set of hyperparameters used are shown:

Learning Rate	Epochs (Step Size: 20)	Optimizer	Batch size	Weight-Decay	Test Accuracy
1e-2	30	SGD+Mom	256	5e-5	50.19% ± (2.4%)
1e-2	30	SGD+Mom	128	5e-5	51% ± (1.7%)
1e-3	30	SGD+Mom	256	5e-5	49.33 % ± (0.1%)
1e-3	30	SGD+Mom	128	5e-5	50.04% ± (0.2%)
1e-4	30	SGD+Mom	256	5e-5	44.68 % ± (3.2%)
1e-4	30	SGD+Mom	128	5e-5	43.18% ± (0.7%)

Table 3: average accuracy of three runs for each set of hyperparameters

Even in this case, the aim has been to show a solid range of resulting performances (the reason why not a really deep research of best hyperparameters has been done): it is under the light how switching the domain of the test set the performances fall down from over 95% to around 50%, because the source and target domains are not so related to guarantee good performances in an ordinary classification task (the source domain is not representative for target domain)

3. Domain adaptation (DANN) and new network setup

To try to overcome this issue a domain adaptation technique comes in handy. In general it consists on learning a discriminative classifier in the presence of a shift between training and test distributions. In this assignment, adversarial-domain neural network (DANN) approach is used: it consists on a net which still uses layers and loss functions and it can still be trained by using backpropagation algorithms based on stochastic gradient descent, but differs from the others in its work flow: it builds mappings between the source and the target domains, so that the classifier learned for the source domain can also be applied to the target one. This is achieved by jointly optimizing the underlying features as well as two discriminative classifiers operating

on the features themselves: one predicts class labels as usually, while the other one, the domain classifier, discriminates between the source and the target domain during the training phase. Both the classifiers received as input the same features extracted through a series of convolutional layers: these are trained simultaneously to confuse the domain discriminator and at the same time to guarantee good performance in labels classification task, and for this purpose two different losses are used. So, in the learning stage, the DANN aims to minimize the label prediction loss on the source labeled data: thus, the parameters of the features extractor and the class predictor are both optimized to minimize the empirical loss on the source domain (this guarantees the features to be discriminative on the source domain). At the same time, DANN attempts to make the features invariant across the domain, which is equivalent to make the distributions of the source and the target similar. In order to do this, the parameters of the feature extractor are also optimized to maximize the classification loss of the domain class (domain confusion loss): this is due to a gradient reversal layer that leaves the input unchanged during forward propagation and reverses the gradient by multiplying it by a negative scalar (Alpha) during the backpropagation. [1] [2]

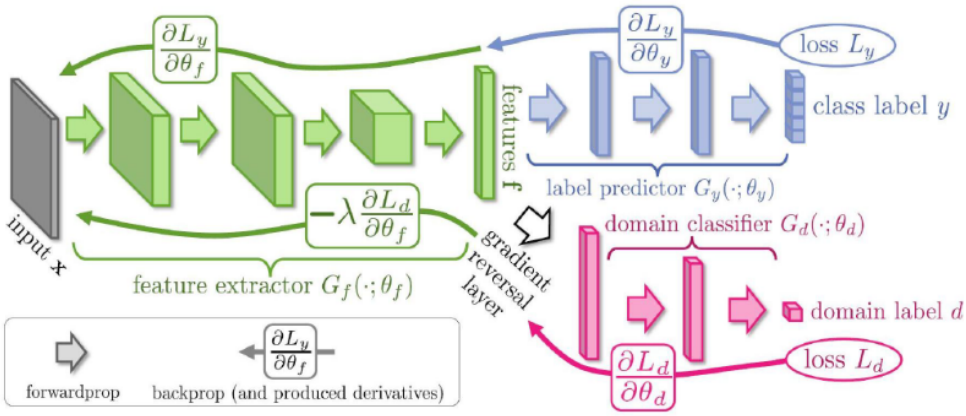


Figure 2: DANN neural network architecture [1]

According to this, a new class AlexNet_Dann class is implemented as well as the one already provided by Torch: the discriminator domains classifier and the gradient reversal layer are added to the default implementation of the net, and the value of Alpha is taken into account to address the feature to the classifiers: where it is equal to None, the features are given as input to the standard class label classifier, where instead it is equal to a float number(to be tuned), the features are driven to the domain discriminator classifier and Alpha works as described above.

Learning Rate	Epochs (Step Size: 20)	Optimizer	Batch size	Weight-Decay	Test Accuracy
1e-02	30	SGD+Mom	128	5e-5	53.21 % ± (2.1%)
1e-02	30	SGD+Mom	256	5e-5	50.34 % ± (2.9%)
1e-03	30	SGD+Mom	128	5e-5	51.4 % ± (1.1%)
1e-03	30	SGD+Mom	256	5e-5	51.10 % ± (0.8%)
1e-04	30	SGD+Mom	128	5e-5	48% ± (0.1%)
1e-04	30	SGD+Mom	256	5e-5	46% ± (0.1%)

Table 4: Average test accuracy of three runs with DANN adaptation

It's important to stress how the training phase has been performed by making the so-called 'cheating', that, according to machine learning language, is used to indicate when information of test set is used during the training phase: in fact, Art painting dataset has been also used in the domain discriminator branch.

Now these results are compared to those obtained without domain adaptation:

Hyperparameters	Test Accuracy without DANN	Test Accuracy with DANN	Increment
LR= 1e-02 BATCH SIZE: 128	51% ± (1.7%)	53.21 % ± (2.1%)	2.21%
LR= 1e-02 BATCH SIZE: 256	50.19% ± (2.4%)	50.34 % ± (2.9%)	0.25%
LR= 1e-03 BATCH SIZE: 128	50.04% ± (0.2%)	51.4 % ± (1.1%)	1.4%
LR= 1e-03 BATCH SIZE: 256	49.33 % ±(0.1%)	51.10 % ± (0.8%)	1.77%
LR= 1e-04 BATCH SIZE: 128	43.18% ± (0.7%)	47.8% ± (0.8%)	4.62%
LR= 1e-04 BATCH SIZE: 256	44.68 % ± (3.2%)	45.78% ± (0.8%)	1.1%

Table 5: Accracies obtained with and without domain adaptation

As expected, by performing training with domain adaptation there are increments meaningful almost for each set of hyperparameters.

4. Cross Domain Valdation

4.1 Without DANN

In this step, the two other domains are taken into account during the training phase and they are used as validation sets to avoid keeping on cheating as previously described: in details, the model is trained with images of the source domain (i.e. Photo) and evaluated by using first Sketch dataset and the Cartoon one, by considering the average accuracy output; then with the best hyperparameters found, the model is trained again and tested on Art Paintings target domain.

Firstly, this is performed without DANN adaptation, so without giving any informations to the network about the domains difference. The model is simply trained with images from Photo dataset and evaluated with Sketch and Cartoon one. Batch size and learning rate are tuned for this task in order to try to achieve the best configuration possible. The other hyperparameters for this first step are kept fixed as:

- Epochs = 30.
- Step Size = 20
- Weight Decay = 5e-5

The performances obtained on both the validation datasets are reported below:

	Sketch		**Cartoon**	
	Batch size=128	**Batch size = 256**	**Batch size = 128**	**Batch size = 256**
LR = 1e-2	33.45%	27.41%	27.8%	28.4%
LR = 1e-3	29.51%	24.54%	25.9%	24.54%
LR = 1e-4	21.3%	16%	19.3%	27.5%

Tables 6: Accuracy on Cartoon and Sketch datasets without DANN

	Average Accuracy per hyperparameters set	
	Batch size = 128	**Batch size = 256**
LR = 1e-2	30.45%	27.41%
LR = 1e-3	29.51%	24.54%

LR = 1e-4	20.3%	21.7%

Table 7: Average accuracies between Cartoon and Sketch

The best average accuracy is given by the combination of Learning rate = 1e-2 with a Batch size of 128.

It is interesting to note how the single accuracy both on Cartoon and Sketch is significantly lower than the one obtained previously on Art Painting, indicating that the first two datasets are worse related with the source one than Art Painting. Anyway, finally the network is trained with the best hyperparameters configuration and tested on Art Painting, obtaining an accuracy of 51%.

4.2 Without DANN

As second step, the same procedure is applied but using also the domain adaptation technique. In particular, the classifier of the model is trained with images from the source target (always Photo), while the discriminator with images from both source and target, which will be once Sketch set, and then Cartoon one. In this way the model have no information about the domain of the set on which it will be then tested.

The hyperparameters to be tuned are the same as before plus Alpha, which controls the backpropagation of the discriminator layer, giving it more or less importance according to its value: the higher it is, the more impactful the discriminator loss will be in the training phase, encouraging the model to focus mainly on confusing the domains. For a first analysis, the alfa values taken into account are: [0.01, 0.1, 0.25].

In the following table are reported directly the average accuracies between cartoon and sketch domains for a more compact representation:

	Alpha = 0.01		**Alpha=0.1**		**Alpha=0.25**	
	BS = 128	**BS=256**	**BS=128**	**BS=256**	**BS=128**	**BS=256**
LR=1e-2	26.34%	28.45%	26.9%	25.2%	17%	17.3%
LR=1e-3	28%	25.3%	26.4%	27%	19.2%	18%
LR=1e-4	23.13%	21.34%	24.56%	22.12%	30.1%	27.3%

Table 8: Average accuracies between Cartoon and Sketch datasets of each set of hyperparameters

From the table it is possible to note how the more alpha increses, the more a low learning rate is needed in order to obtain reasonable performances. In fact, the worst average accuracy (17%) is achieved with a learning rate of 1e-2 and a value of alpha equal to 0.25, in fact the losses diverge after few epochs.

Consistent with this fact, the best average accuracy (30.1%) is given with a low learning rate, 1e-4, even combined with an Alpha of 0.25. According to this, kept fixed the learning rate higher values of alpha are tested in order to try to achieve even better performances. In particular, alpha=0.5 and alpha=1 are used.

The best average performances are obtaining with:

- Learning rate: 1e-4
- Batch size: 128
- Alpha: 0.5

Which lead to an accuracy of 33.4% on Cartoon domain and of 28.1% on Sketch, for an average accuracy of 30.75%. The losses converge right away to zero, as shown in the chart below:

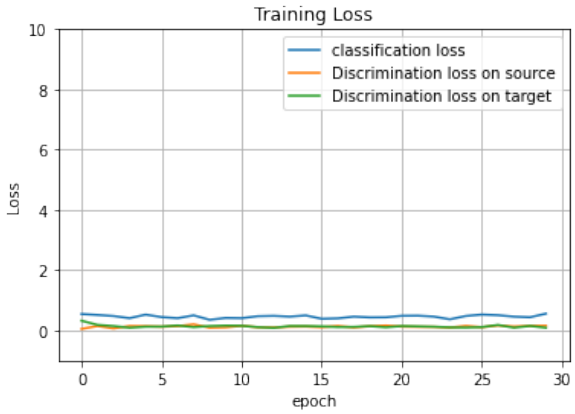


Figure 3: last batch (for epochs) losses trend

The net is trained again with these hyperparameters and it is testet on Art Painting dataset, providing an accuracy of 51%: this is slighlty lower than the one obtained without domain adaptation, but this hyperparameters provide a more robust model against domains shift because they are the ones best performing on different domains, among those tested. At the same time, the accuracy obtained without cross validation is higher because those hyperparameters were tuned according only to Art Painting images distribution, so they perform well on that dataset and other pretty similar, but they are that less robust to domain shift, leading to bad performances when used on another domain classification.

REFERENCES

[1] https://towardsdatascience.com/deep-domain-adaptation-in-computer-vision-8da398d3167f

[2] https://arxiv.org/pdf/1802.03601.pdf