Gabriele Tiboni
S276241

Report of Homework #3

MACHINE LEARNING AND DEEP LEARNING COURSE

1. Introduction

The scope of this work will be to understand how to implement the DANN (Domain-adversarial neural network) algorithm on AlexNet architecture in order to address domain adaptation in an image recognition task. Performances both with domain adaptation and not will be compared and analysed on the PACS dataset.

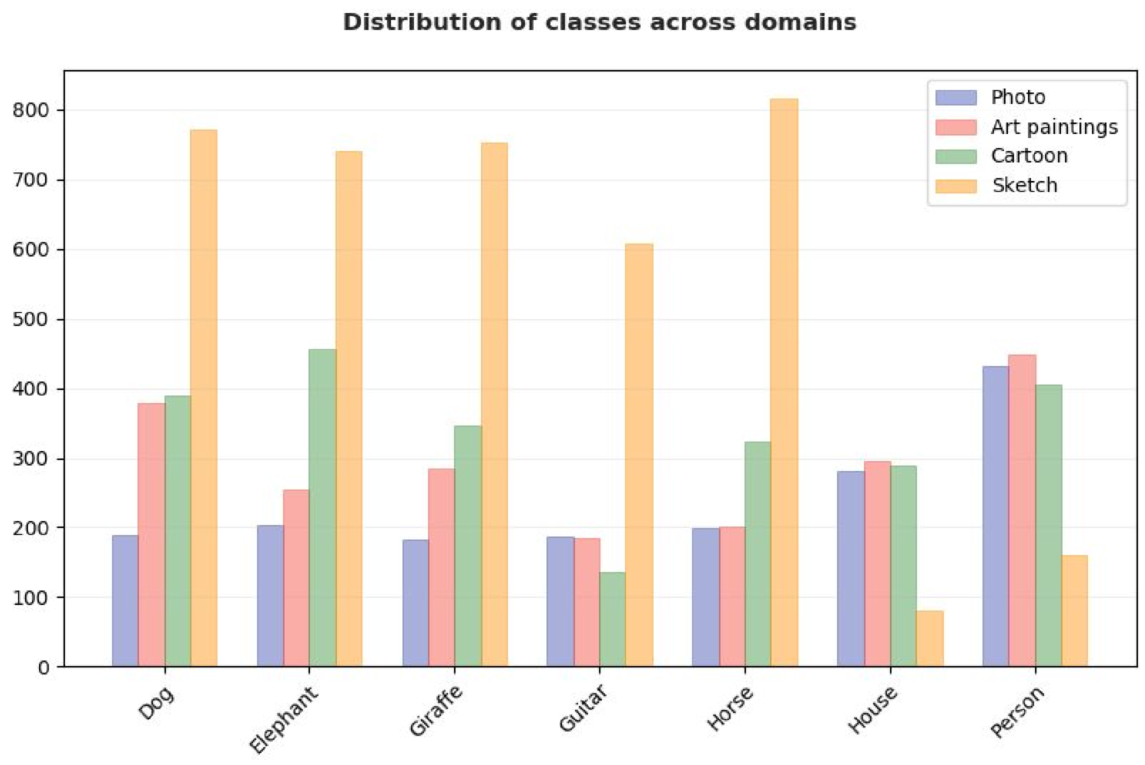


Figure 1 *class distributions across domains on the PACS dataset.*

The given dataset is composed of a total of 9991 squared images of size 227x227, coming from four different domains: *Photo, Art paintings, Cartoon* and *Sketch.* Each domain has the same seven labels stated in fig. 1.

The precise dataset distribution across domains is as follows:

- *Photo*: 1670
- *Art Paintings*: 2048
- *Cartoon*: 2344
- *Sketch*: 3929

See [3] for all details on PACS dataset.

The domain adaptation task consists in performing a classification analysis under the assumption that test data (a.k.a. *target* data*)* follows a slightly different distribution w.r.t. training data (a.k.a. *source* data). For example, the *Photo* dataset contains real photos of dogs and elephants, whereas the *Sketch* dataset only contains handwritten sketches of dogs and elephants. May a model learn to recognize sketches or art paintings when trained on photos only?

Source *Target*

Our goal is to build a robust model that is able to classify correctly regardless of the domain shift between source images and target images. In particular, this approach is referred as "*unsupervised domain adaptation"* as no target labels are needed during the training phase (only the actual knowledge of what domain each image belongs to). Still, source images are clearly labeled and will be used to train the classifier.

All the code, implementation details and results of the following work are available at https://github.com/gabrieletiboni/DANN-on-PACS-dataset

2. Model implementation

The DANN architecture (Domain-adversarial neural network, see [1]) will be used for addressing the domain adaptation task. This approach is directly inspired by the theory of domain adaptation which suggests that, in order to achieve meaningful results in domain transfer, predictions must be made upon features that cannot discriminate between source and target data.

Thus, the DANN method proposes to embed the domain knowledge into the learning process, by slightly modifying the existing network structure. In particular, a new MLP classifier referred as *domain classifier* is added after the convolutional layers of a CNN (feature extractor) and is used during training as a binary classifier to distinguish between source and target data.

The magic will then happen during the training process, where for each minibatch composed of both source and target data the following steps are executed:

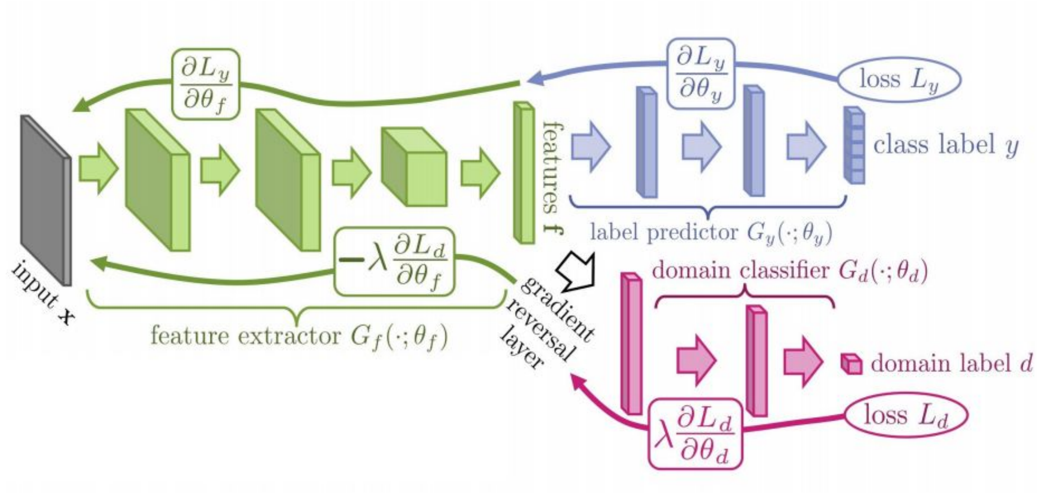
1. Source data is forwarded both through the standard classifier (*label predictor*) and through the *domain classifier* ;
2. Target data is forwarded only through the *domain classifier* ;
3. Derivatives w.r.t. weights on the feature extractor are computed in order to **minimize** the loss on the label predictor and **maximize** the loss on the *domain classifier* (the remaining weights are updated in the standard way);

Point 3 above is implemented by reversing the gradient that backpropagates from the domain classifier (which is also multiplied by a new hyperparameter *alpha*), so that updates follow the derivative direction (gradient ascent).

> This way, images during test time are mapped to a new feature space that is encouraged to discriminate between labels while making hard to distinguish between domains.

In fact, the approach aims to learn a model that can generalize well from one domain to another (on the same classification task), ensuring that the internal representation of the neural network contains no discriminative information about the origin of the input, while still preserving a low average loss on source data.

The following figure sums up the DANN architecture and its training process:



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input x
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda\frac{\partial L_d}{\partial \theta_d}$

Figure 2 *DANN implementation on AlexNet*

Regarding the actual network implementation, the AlexNet network has been used with pretrained weights on ImageNet, and the domain classifier has been created as a copy of the FC part of AlexNet (with only two output neurons) attached at the end of the feature extractor (after the average pooling). The domain classifier has also been initialized with the same weights as the AlexNet Fully Connected part, except for the output layer.

For easiness, in this work no data augmentation has been performed on the training dataset, since the main focus has been put on the relative difference in performance between the domain adaptation approach and the vanilla one. Though, input data is still being transformed by standardizing each channel according to mean and standard deviations of PyTorch models pretrained on ImageNet.

3. Model training and evaluation

For the current analysis, it is required to build a model with photos only and make it able to recognize art paintings (on the same task) as best as possible. Hence, the *photo dataset* will be used as the source set while the *art painting dataset* will be used as the target set. The *cartoon* and *sketch* datasets will be later used for hyperparameter tuning (cross-domain validation).

Before actually testing the architecture on the domain adaptation task, a first run has been performed on the source dataset only, to check whether the network could efficiently converge if no domain shift was present in between training and test images.
With 70% of photos used for training and the remaining 30% for the model evaluation, the results obtained were the following:

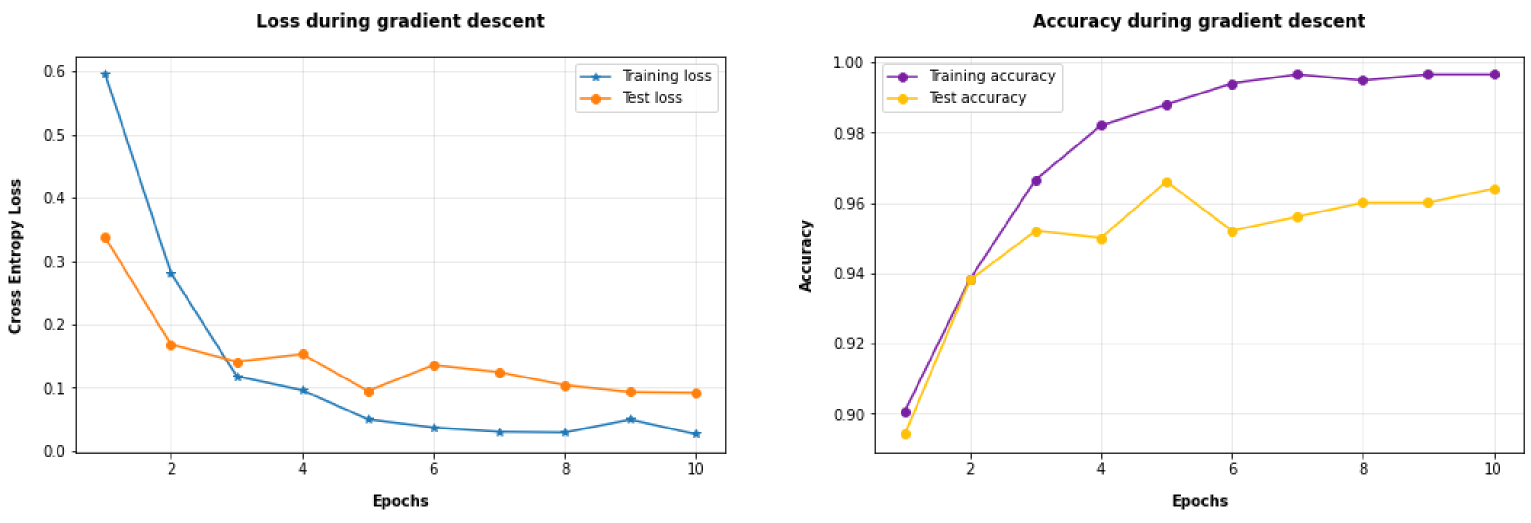


Figure 3 *Model evaluation when training on photos and testing on photos*

The model showed to converge fairly well, being able to learn classes with a **96,3% ± 1,3%** accuracy when no domain shift is added at test time. An upper bound for our task has then been found, it was now time to check how well the model could perform when test data would be slightly different than training data.

As stated in the assignment, the whole *photo dataset* has been used for training the model (1670 images), and the whole *art painting dataset* has served for model evaluation (2048 images). No hyperparameter tuning has initially been performed and a set of reasonable starting hyperparameters have been chosen. Furthermore, comparisons between models throughout this report will take into account accuracies only (instead of losses), since it is the preferred approach followed by the DANN paper and it was so suggested by the course assistants for this task.

Firstly, the model was trained without domain adaptation (with the standard training process), and obtained the following results on the *art painting dataset*:

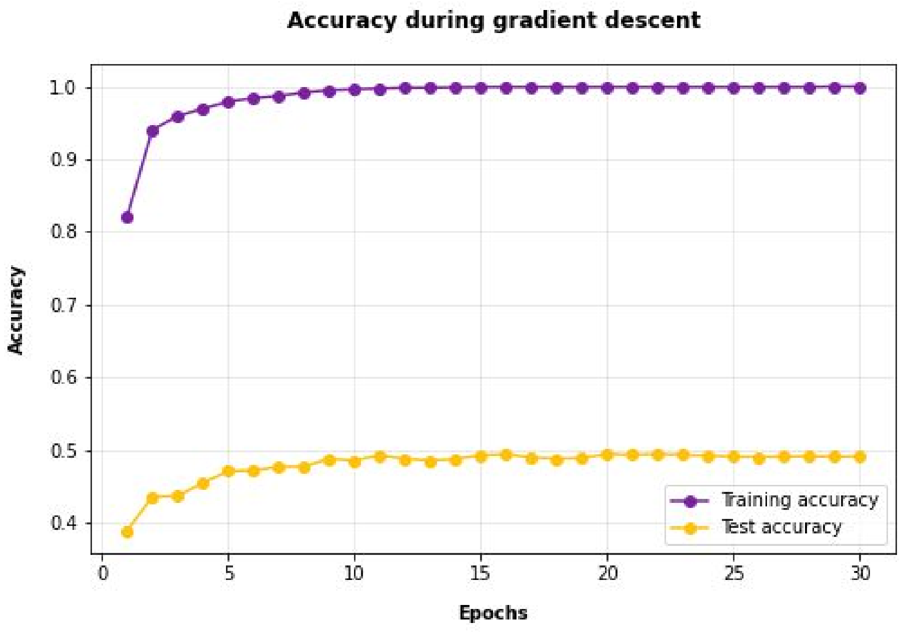


Figure 4 *No domain adaptation. LR=1e-3, Weight_decay=5e-5, step_size=20, gamma=0.1,Batch_size=256*

The model converged on the training set very quickly after only about 10 epochs, but it suffered a lot from the domain shift in images at test time with respect to the previous standard learning scenario. With the above hyperparameters in *fig. 4* the network achieved **49,43% ± 0,33%** accuracy on the art painting images, averaged over 3 runs.

Later the model has been retrained with the same hyperparameters as before but with domain adaptation, by using the DANN approach in the training process implemented in point 2 with alpha=0.1. The following results were obtained:

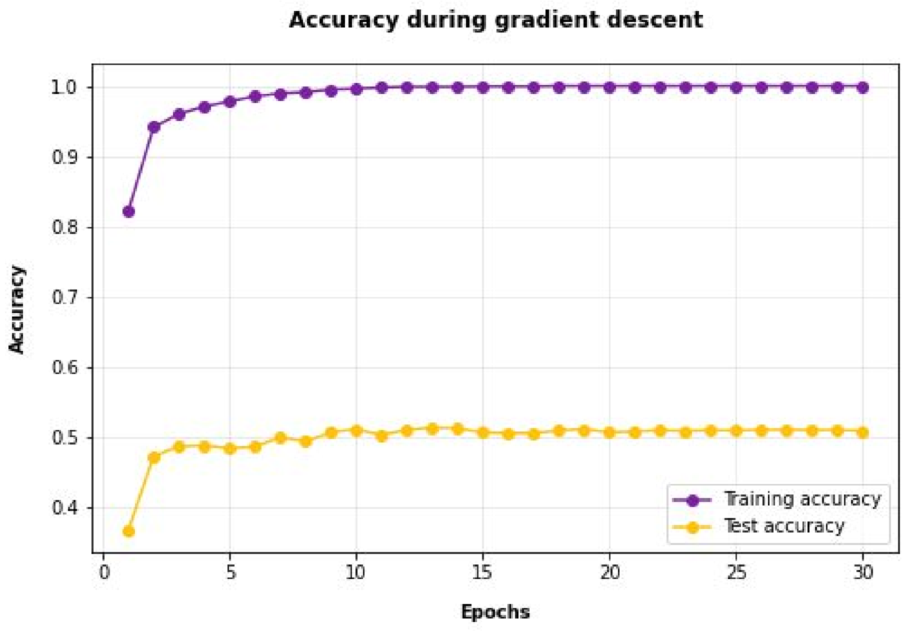


Figure 5 *With domain adaptation. LR=1e-3, Alpha=0.1, Weight_decay=5e-5, step_size=20, gamma=0.1, Batch_size=256*

The learning curve was similar, but the final test accuracy after 30 epochs has increased when the domain adaptation technique was used during training. In fact, the model achieved a higher **50.63% ± 0.18%** final accuracy on art paintings. Even though both models don't reach very high absolute performances, a consistent improvement has been observed when applying the DANN strategy during training, which proved the theoretical assumption of the approach to be correct. The feature extractor must have learned a slight different feature space representation that would be less discriminative in terms of domain while still maintaining separation between classes.

In particular, the two models have been compared side by side in the figure below:

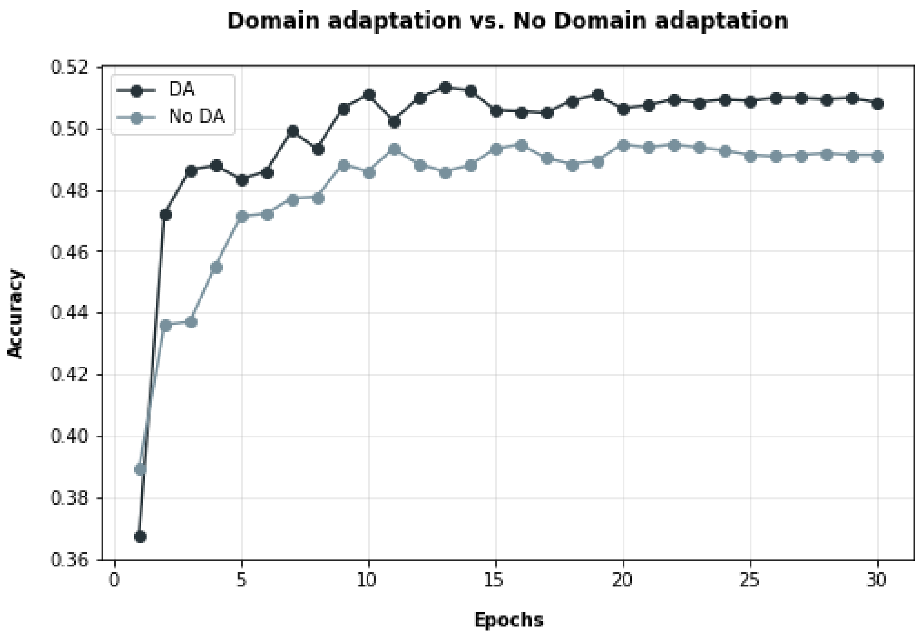


Figure 6

When directly compared, it is easier to see the improvements brought by DANN adaptation, ever since the very first epochs.

Finally, it is interesting to note that the loss on the domain classifier is not exactly increasing during training with domain adaptation. It is indeed decreasing as shown in figure 7:

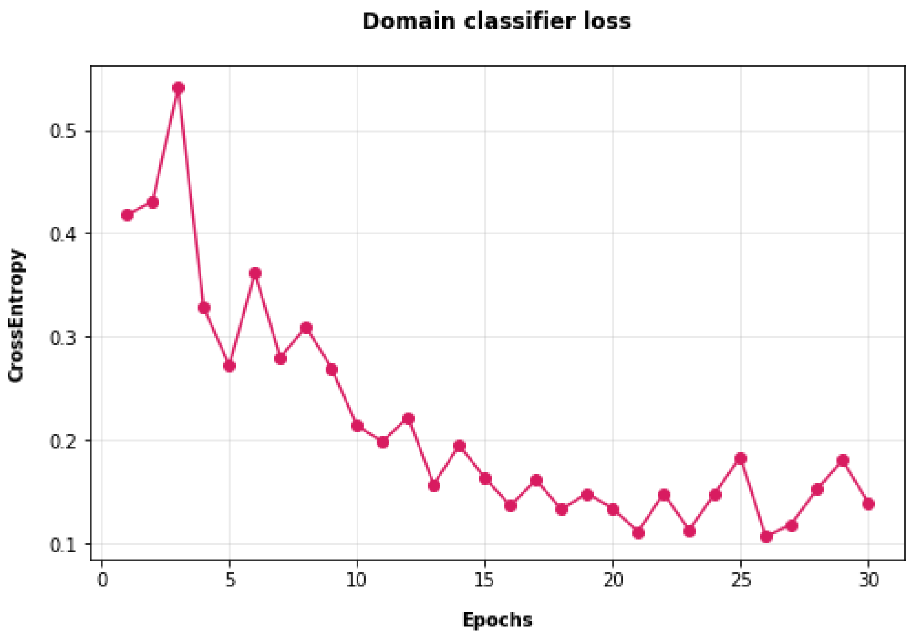


Figure 7

This phenomenon is in fact expected, as, while the feature extractor is trying to maximize the loss of the domain classifier, the fully connected layers are still being updated in order to minimize the final loss. Furthermore, since alpha has been set to 0.1 (multiplier of the reverse gradient) weight updates on the FC layers lead to a higher contribute in loss change, resulting in a net decrease over time.

4. Cross-domain validation

Like every machine learning task, hyperparameter tuning is a key step to find the best performing model for the given analysis. Unlike standard learning scenarios though, domain adaptation makes it a little harder to find the best set of hyperparameters, and validation is indeed still an open problem in domain adaptation settings.

Since the whole target dataset is used during training for the DANN procedure (adaptation set and test set coincide), one could think to tune the hyperparameters based on the test results. However, such approach would result in making decisions based on test set labels, which is formally incorrect given the initial assumptions that no target labels are needed for the task. Remind that target data is still needed during training time (unlike standard test data in other learning scenarios), it just does not contain any labels.

Thus, in order to make sure that no test labels are used for hypertuning, a different technique has been used known as *cross-domain validation*. This approach takes advantage of the other two domains in the PACS dataset (*cartoon* and *sketch*), by performing the same domain adaptation task on each of the two and averaging the obtained results for finding the best model.

It is important to note that in this case the true labels of cartoon and sketch samples are instead needed, since they will serve as validation datasets.

Two different grid-searches of hyperparameters have been performed on the cartoon and sketch domains, respectively with and without DANN adaptation during training.

Note that due to Colab limitations grid-searches have been done manually. Other techniques can be used if free access to GPU's is available, such as randomly select different sets of hyperparameters.

4.1 Cross-domain validation without DA

The first grid-search, performed by averaging the final accuracies on photo to cartoon transfer and photo to sketch transfer, without DANN adaptation, is fully reported in the following table:

	Batch size	128		256	
	StepSize-#Epochs	20-30	30-60	20-30	30-60
Learning rate	1e-2	***25,11***	***28,83***	***28,69***	***27,97***
	5e-3	***28,93***	***29,92***	***30,03***	***28,88***
	1e-3	***29,49***	***28,21***	***27,36***	***26,57***
	1e-4	***25,82***	***27,80***	***27,41***	***24,62***

Table 1 *Average accuracy on photo to cartoon transfer and photo to sketch transfer, with momentum SGD and step down factor of 0.1. Each value has been averaged over three runs to get a better estimation.*

As stated in Table 1, setting the learning rate equal to 0.005 consistently led to better results, even when considering a different step size or batch size. Overall, the following best set of hyperparameters has been found:

- *Learning rate:* **5e-3**
- *Batch size:* **256**
- *Step size:* **20**
- *Number of epochs:* **30**

The best model found has been further inspected during the training process, shown in fig. 8 below:

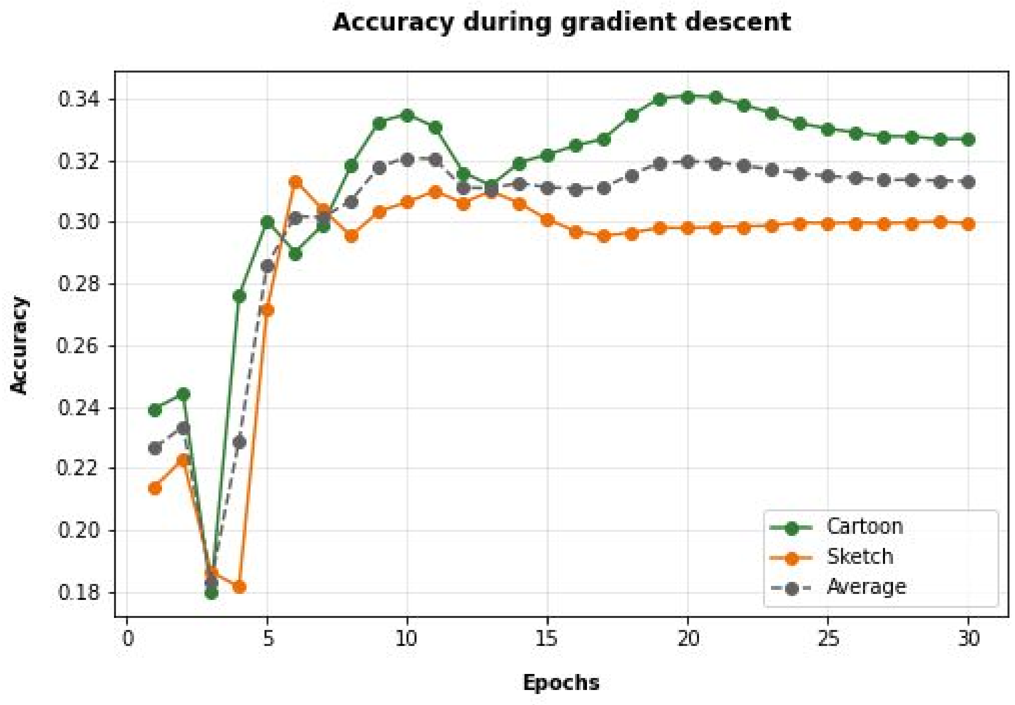


Figure 8

With respect to the art painting domain, the network struggled even more when tested on cartoon and sketch domains, whereas only a 30% average accuracy is reached. However, note that the accuracy obtained during validation does not give any indication on how the final model on art paintings will perform, since different adaptation and test data will be used. Average accuracies on cartoon and sketch domains are solely used for finding the best set of hyperparameters.

The current best model has finally been evaluated on photo to art paintings transfer using the best set of hyperparameters found, and achieved **52,02% ± 1.85% accuracy**, averaged over 5 runs, when trained without DANN adaptation.

4.2 Cross-domain validation WITH DA

The second grid-search has instead been performed with DANN adaptation during training. Once again, the entire *cartoon* and *sketch* datasets have been used for both adapting the model during training with adaptation and evaluating the model afterwards (transductive domain adaptation). Remind that a new hyperparameter *alpha* is added when using the DANN approach, which proportionally scales the backpropagating gradient coming from the domain classifier (See point 2.).

Table 2 below reports all the results obtained:

Learning rate	Batch Size	*Alpha*			
		0.01	0.1	0.5	1.0
1e-2	128	***28,42***	***18,12***	-	-
	256	***30,25***	***24,02***	-	-
5e-3	128	***29,98***	***36,35***	-	-
	256	***28,93***	***34,53***	-	-
1e-3	128	***28,25***	***28,40***	-	-
	256	***25,34***	***26,74***	-	-
1e-4	128	***24,35***	***25,01***	***27,60***	***38,39***
	256	***24,71***	***26,38***	***25,07***	***30,60***

🟨	*Diverged during training*
🟩	*Highest score*

Table 2 *Average accuracy on photo to cartoon transfer and photo to sketch transfer, with DANN adaptation during training, on 30 epochs and step-size=20. Each value has been averaged over 3 runs to get a better estimation.*

This time the best set of hyperparameters found, according to the highest average accuracy, was:

- *Learning rate:* **1e-4**
- *Batch size:* **128**
- *Alpha:* **1.0**

With an average accuracy over photo to cartoon and photo to sketch transfer of **38,39%** ± **2.35%**.

Note that all values refer to 30 epochs and step size equal to 20, in order to make the manually grid search feasible in terms of time and effort. Some higher step sizes have also been tried here and there, but often led to very similar or worse results.

It is important to highlight that due to the different learning setting with domain adaptation, higher learning rates or higher values for alpha caused divergence on the training losses. Indeed, the model was sometimes not able to perform the maximization-minimization trick on the two classifiers and ended up diverging quickly on one of the two validation domains.

The following graph shows how the accuracy remains unchanged on the sketch domain and the model is no longer able to learn when the loss diverges and its gradient cannot be computed:

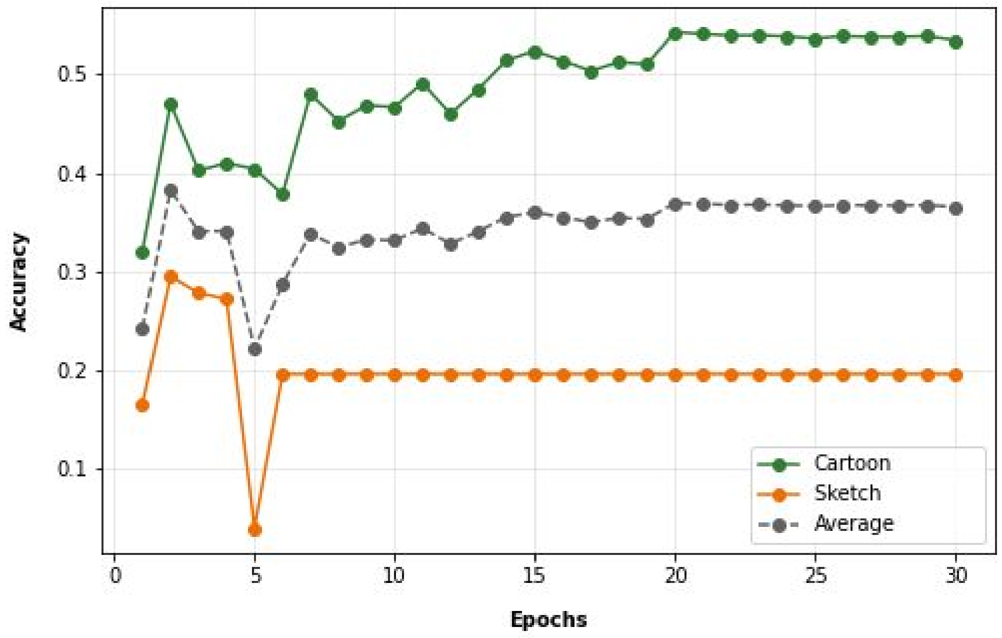


Figure 9 *Cross-domain validation with LR=5e-3, Batch size=128, Alpha=0.1*

When decreasing the learning rate, instead, the network could handle each learning step even at higher values of alpha.

The current best model has finally been evaluated with DANN adaptation on photo to art paintings transfer using the best set of hyperparameters found, and achieved **48,78% ± 1.70% accuracy**, averaged over 5 runs.

Later, the same model has been retrained **without domain adaptation**, in order to further study the relative difference in final performances of the two approaches. The new model obtained 45,28% ± 1.41% accuracy on average, and proved once again the consistent performance increase brought by the DANN approach.

A final comparison, on the best set of hyperparameters just found, is shown in fig 10 below:

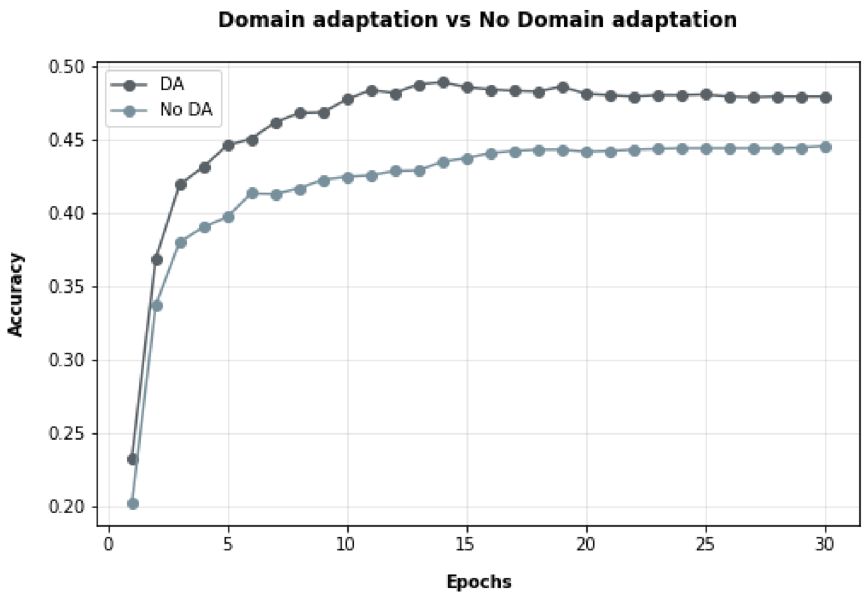


Figure 10 *LR=1e-4, Batch size=128, Step size=20, Alpha=1.0*

Note that the first grid-search performed without domain adaptation led to a final higher accuracy on the art paintings dataset (52,02%) w.r.t. the second grid-search (48,78%). However, in a real world scenario such comparison would not be possible since target domain labels are not accessible, hence it would be formally wrong to choose the best performing model upon test results. Furthermore, if losses are taken into account instead of accuracies, with the higher learning rate 5e-3 a stronger overfitting is observed and a much higher final test loss is recorded, making it hard to choose between the two models even if test labels were accessible.

Finally, in hope to reach even better performances, one could follow a different approach for setting alpha that has also been presented in [1] (referred as *lambda* in the paper), consisting in an exponential scheduling that makes alpha go from 0 to 1 over the course of training.

References

[1] Domain-Adversarial Training of Neural Networks (DANN), 2016
https://arxiv.org/abs/1505.07818

[2] PyTorch implementation of DANN
https://github.com/fungtion/DANN

[3] PACS dataset
https://github.com/MachineLearning2020/Homework3-PACS