Daniele Genta, s276922
Machine Learning and Deep Learning
Homework 3 – Report
Academic Year: 2019-2020

0 - Introduction

The goal of this homework is to implement DANN (Domain Adaptation Neural Network), a domain adaptation algorithm, on the PACS dataset, by using a customized AlexNet architecture.
Domain Adaptation is required when there is no access, during the training phase, to the domain (i.e. distribution) where the test pictures are taken from.
What happens is that when a model is tested on a dataset different from the training one is that its performances could drop dramatically.
In order to address the issue, a domain adaptation step is required. This methodology allows both the classification of input images in the source domain and the ability to transfer the classification effectiveness to the target domain, by properly modifying the architecture of the Neural Network.

Network architecture

In order to achieve the goal of the homework, an ad-hoc version of the AlexNet architecture will be used.
To be more specific, as it is shown from the following **figure 1**, after the standard AlexNet's Convolutional Layers, there are two parallel branches of Fully Connected layers (instead of a single path). The goal of this split is to identify if the currently analyzed image comes from the source or the target domain beside being able to classify properly the input.
The network is trained jointly on a labeled task, i.e. Photo classification, and an unsupervised task, i.e. learning to discriminate between Photo and Art Painting. Finally, the model is tested on Art painting.
Since PACS dataset is made up of 7 different classes, the output of the architecture will be a 7-dimensional vector concerning the main classifier branch, whereas the output of the domain classifier branch will be binary and will serve the scope of reporting whether the input image comes from the source or the target domain.

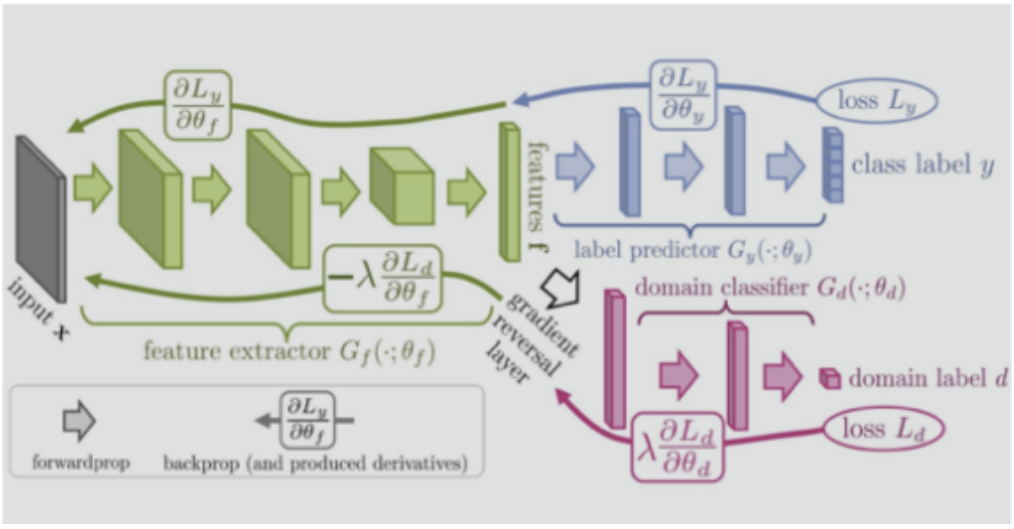

***Fig. 1**: AlexNet modified architecture*

1 - Dataset

The provided dataset is PACS, which contains a set of approximately 10'000 pictures distributed among 7 classes and 4 domains, which are: **P**hoto, **A**rt painting, **C**artoon, **S**ketch.
The goal of this homework, using Domain Adaptation, is to train the ad-hoc network on images of the Photo domain and then being able to correctly perform classification over instances from the Art domain at test phase.

Fig. 2: *Examples of the content of PACS, same class different domain (Photo and Art)*

The distribution of the domains is reported in the bar plot below (each bar takes into account images both from the train and the test portion of the dataset). As it can be seen from the following figure, the number of elements among different domains is unbalanced.

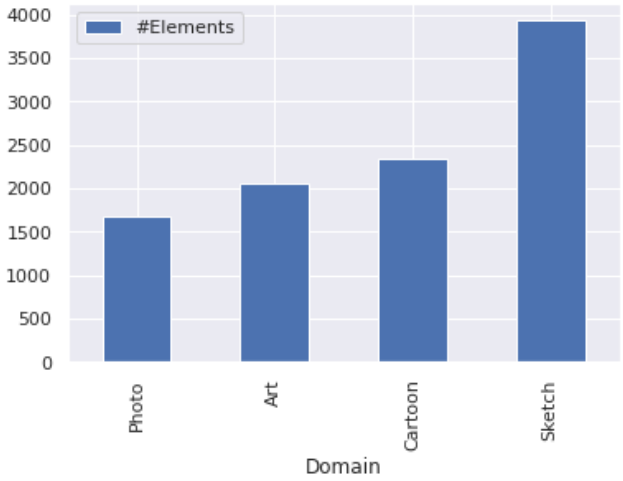

Fig. 3: *Distribution of the domains in PACS*

2 – Implementing the model

The goal of this section of the homework is to explain how to build the DANN, starting from the AlexNet default implementation.

As explained in the previous introductory section, what it is wanted is to build an architecture as shown in **figure 1**, starting from AlexNet.
Operatively, a separate branch that resembles the already existing Fully Connected layers should be added to AlexNet and, more specifically, a new densely connected layer with 2 output neurons must

be added. The implementation consists of 2 output neurons since this additive branch will serve as a binary classifier of the domain of the input images.
In order to control if the data along the network is passed to the main or the domain classifier, an ad-hoc flag, represented by the *Alpha* parameter, is used in the forward function and if the data is actually passed to the domain classifier then the gradient has to be reverted with the Gradient Reversal layer, the reversal is actually multiplied by the parameter alpha that controls the weight of the reversed backpropagation and should be optimized.
Throughout the entire homework AlexNet pretrained on ImageNet is used that, as explained in **homework 2**, exploits a set of weights learned on the ImageNet dataset.

3 – Domain Adaptation

Network setup

As for the network setup, its basic implementation, which will serve as baseline for all the following trials, involves a training over 30 epochs with an initial learning rate (LR) of 1e-3 and a loss function represented by the Cross-Entropy Loss function.
The gradients are updated, during the steps, via Stochastic Gradient Descent (SGD) with momentum (fixed at 0.9).
The learning rate is decaying by a factor of ten (GAMMA) after every 20 epochs (STEP-SIZE).
Furthermore, each step in executed on batches of 256 items, so each epoch involves 8 steps (which is the result of: training set size/batch size).

These parameters have been imported form the Homework 2 task and slightly modified, a further and more comprehensive study might perform a grid-search in order to optimize them on this homework.

A - Training without domain adaptation

The first step of the training phase was to use the Photo dataset as training set and the Art dataset as test set. There was no validation set. In this setting domain adaptation wasn't used, so there was no target set either.
By training the pretrained DANN without domain adaptation, the result was an **accuracy of 47**% and a loss trend that is reported in the following figure (**fig. 3**). The reported loss is not satisfying since is not very smooth and neither it is monotoning decreasing, further processing as DANN and hyper-parameter tuning will be discussed to address this issue (see **following sections**).

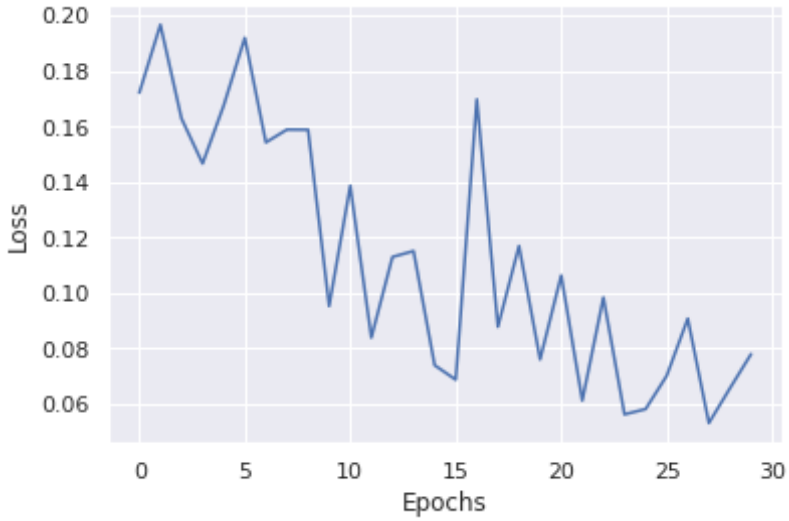

Fig. 4: *Training without domain adaptation, epochs against loss*

Table 1: *Training without domain adaptation experiment*

Architecture	LR	#Epochs	Step-size	Accuracy result (test set)
AlexNet	1e-3	30	29	0.47

B - Training with domain adaptation

The next trial of the training phase involves training with domain adaptation. This means that the network must be trained jointly on the labeled task (Photo) and the unsupervised task (discriminating between Photo and Art painting). In order to do that the entire scheme proposed in **fig.1** must be exploited and more specifically each training operation is divided into 3 steps.
Firstly, the network is trained on source labels by forwarding the source data (Photo images) to the first branch of the DANN, i.e. Gy.
Secondly, the discriminator is trained by forwarding the source data to the second branch of the DANN, i.e. Gd.
Finally, the discriminator is trained again by forwarding the target data (Art paintings images) to the second branch of the DANN.
Note that the last two steps are implemented in order to train a binary classifier able to distinguish between Photos and Art paintings.
In this section of the homework the entire Photo dataset is used as training set, in a supervised learning setting and the entire Art paintings dataset is used both as training set, in an unsupervised learning setting and as test set (Art is both the target and test set).
The first experiment is carried out by using the same hyper parameters of the previous paragraph plus a default alpha = 0.5. Alpha is a coefficient that multiplied to the inversion of the standard learning gradient gives the discriminator loss.
The number of epochs has been decreased 15 in order to speed up the computations and reduce overfitting, this is confirmed by the loss trend in **fig. 4**.
The result of this experiment is reported below (**Table 2, Fig. 4)** and in particular each kind of loss against the number of epochs.
Hyperparameters optimization is not performed in this setting since the parameters search will be implemented in the following section (**section 4 – Extra).**

Table 2: *Experiments training with domain adaptation*

N	Alpha	Batch size	LR	#Epochs	Step-size	Accuracy result (test set)
1	0.5	256	1e-3	30	20	0.49

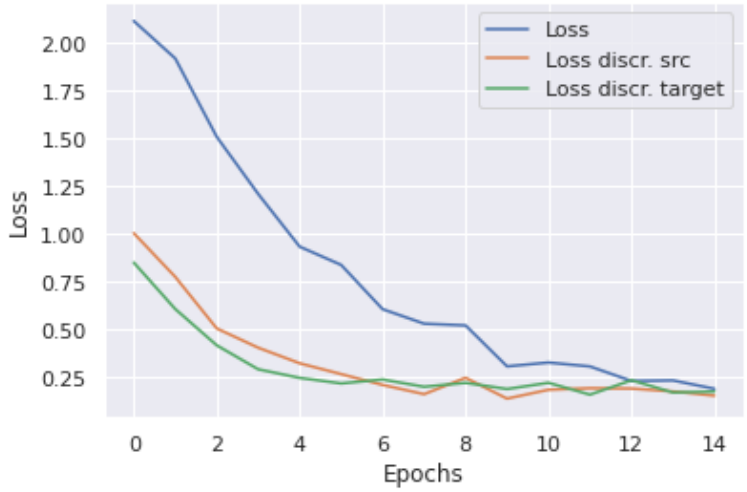

Fig. 5: *Training without domain adaptation, epochs against loss*

As reported in the above data, by performing Domain Adaptation the accuracy result on the test set improves by a small percentage, also the loss trend appears to be smoother. Considering the task and the dataset, this improvement can be considered as a success and the goal is to try to improve it further in the last section of this homework (**Extra**) by tuning the hyper-parameters through cross domain validation.
Note that the results are not fully deterministic, yet they can change by running the task multiple times (+- 1% test accuracy).

4 – Extra (Cross domain validation)

In the previous sections (**section 2, section 3**) validation was not performed, this represent an open research problem in domain adaptation.
Since looking at the test set results in order to tune the hyper-parameters is considered as cheating and not realistic, another methodology to validate Photo to Art should be pursued, without using the Art painting labels.
What is done is to use the other two domains (i.e. Cartoon and Sketch) as target sets.
Operatively, what is done is to tune the hyper-parameters on Photo to Cartoon and Photo to Sketch with and without domain adaptation, finally the result is averaged for each set of hyper-parameters and the previous points (**section 3**) repeated by testing the best scoring model on the Art dataset.

Grid Search

In order to perform the model selection, i.e. find the model with the best hyper-parameters, a manual grid search methodology has been adopted.
To be more specific: different combinations of the parameters LR, ALPHA, WEIGHT_DECAY and STEP_SIZE have been examined, whereas the other parameters have been kept fixed.
While ALPHA has not been tuned in the setting without domain adaptation (since it is not used), STEP_SIZE has not been tuned in the Domain Adaptation setting, due to lighten the workload of the network (and overcome Colab limitations) and it has been kept fixed at 10.
The choice of performing the grid search on three parameters only is mainly due to Colab limitations and the training time required.
The parameters and values considered are:

- LR = [1e-2, 0.005, 1e-3]
- ALPHA = [0.1, 0.5, 0.8]
- STEP_SIZE = [10,14]
- WEIGHT_DECAY = [5e-4, 5e-5]
- NUM_EPOCHS = 15
- BATCH_SIZE = 256
- MOMENTUM = 0.9
- GAMMA = 0.1

So, twelve and eighteen different hyper-parameters' combinations have been considered respectively in the with and without domain validation setting.

The analysis has been performed on training with and without domain adaptation, in both cases the network has been trained on Photo and validated on Sketch and Cartoon (in case of DANN the network has been trained on these two as well). Then, the results of the two trainings, in terms of accuracy, have been averaged and used to select the best scoring model. This model's and its related hyperparameters have been used to repeat point **3A** and **3B** that involved a training on Photo and a test on Art. The result of this final testing is the score of the model on the test set.

A - Cross domain validation without domain adaptation

As explained in the previous paragraph, a grid search is run on 12 different combinations of hyper-parameters and the results, in terms of validation accuracies and losses are registered (see **Appendix**). At the end of the grid search, the best scoring model and its set of hyper-parameters is chosen in terms of the maximum mean of the validation accuracies (cartoon and sketch accuracy).
We know want to use the parameters found in the previous paragraph in order to perform again the first experiment of **section 3** (i.e. training without domain adaptation).
The best configuration of hyper-parameters and the results achieved by using it are reported below (**table 3**, **fig. 5**).

Table 3: *Cross domain validation, without DANN*

LR	WEIGHT-DECAY	STEP-SIZE	Cartoon Accuracy (validation 1)	Sketch Accuracy (validation 2)	Art Accuracy (test)	Art Loss (test)
0.005	5e-5	13	0.29	0.32	0.49	2.55

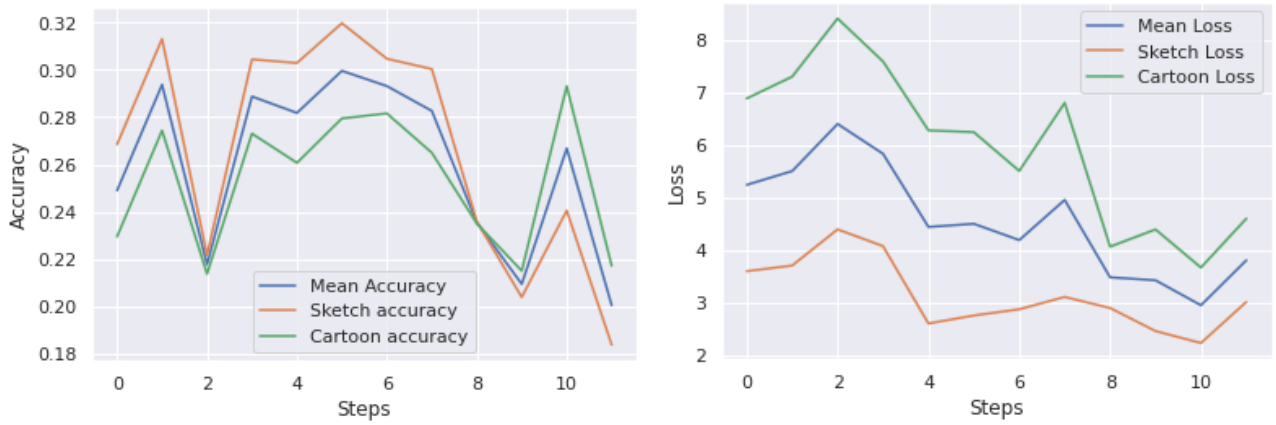

Fig. 6: *Trend of the losses and the accuracies on the validation sets in each step of the grid search*

As it can be seen from the previous plots, the accuracies on the validation sets is always smaller than what is has been achieved in the previous sections. This is because our model is trained with images coming from a specific domain (Photo) and validated on domains with very different features representation (Cartoon and Sketch). Furthermore, it might be assumed that the Cartoon and Sketch datasets have a different distribution and features from the Art dataset.
This is somehow confirmed from the plots in **fig. 5** that show that the loss is very high in and the accuracy is low with all the parameters' configurations.
On the other hand, the test accuracy achieved on the Art domain is comp to what is has been obtained in the previous sections.
Performing a grid search, without implementing the DANN, achieves a slightly better result with respect to the experiment carried out in section 3A, this could be interpreted as the consequence of having chosen better preforming parameters.

B - Cross domain validation with domain adaptation

Similarly, to before a grid search in run on 3 different hyperparameters simultaneously (LR, WEIGHT_DECAY and ALPHA in this case) and a total of 18 different combinations are trained and the results, in terms of validation accuracies and losses are registered.
At the end of the grid search, the best scoring model and its set of hyper-parameters is chosen in terms of the maximum mean of the validation accuracies (cartoon and sketch accuracy). What is different is that, using a DANN setting, we have to train two different nets for each set of hyper-parameters, this result in a longer training and heavier computations.
We now want to use the parameters found in the previous paragraph to perform again the second experiment of **section 3** (i.e. training with domain adaptation).
The best configuration of hyper-parameters and the results achieved by using it are reported below (**table 3**, **fig. 5**).

Table 4: *Cross domain validation, without DANN*

LR	WEIGHT-DECAY	ALPHA	Cartoon Accuracy (validation 1)	Sketch Accuracy (validation 2)	Art Accuracy (test)	Art Loss (test)
1e-3	5e-4	0.5	0.51	0.39	0.5	3.5

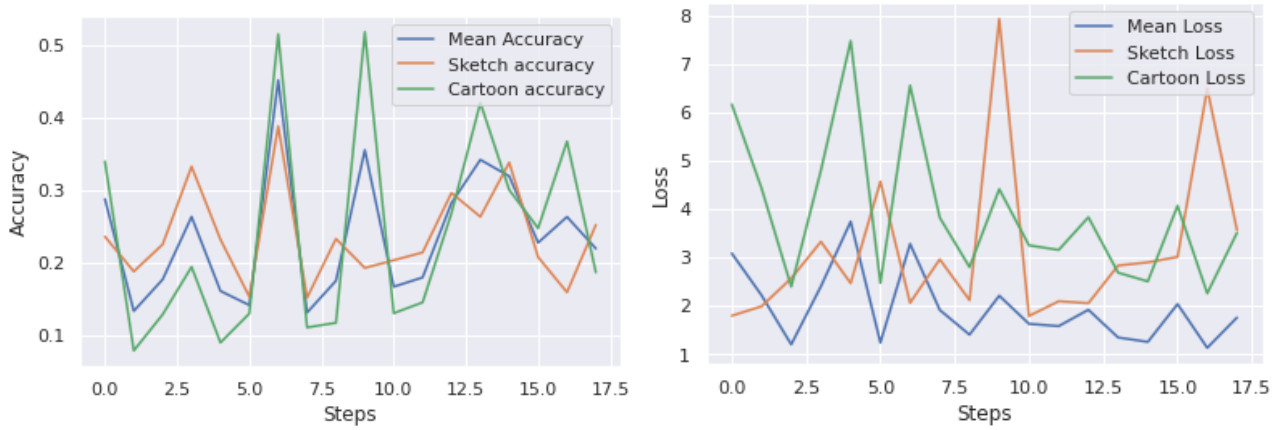

Fig. 7: *Trend of the losses and the accuracies on the validation sets in each step of the grid search (DANN setting)*

The final insights that can be extracted from these experiments are that cross domain validation is a good choice in order to validate the hyperparameters of a model, in a DANN setting.
In fact, in these kinds of problems even a small increasing in the test performances is substantial. However, the fact that the results are not deterministic, yet they present a certain variance, could imply that without performing a complete statistical analysis, the results could be somehow misleading.
The limitations of the Colab environment and the training time required made it unfeasible to perform further studies such as a complete grid search over all the hyper-parameters, by considering more values of them. This could be a subject of further studies carried out with a proper equipment.

Appendix

Complete results 4A - Accuracy

n	LR	WEIGHT_DECAY	STEP_SIZE	Cartoon Acc	Sketch Acc	Mean Acc
13	1E-02	5E-05	10	0.2295221843003413	0.26851616187325017	0.24901917308679572
14	1E-02	5E-05	14	0.2743174061433447	0.3130567574446424	0.29368708179399355
15	1E-02	5E-04	10	0.2137372013651877	0.22168490710104352	0.2177110542331156
16	1E-02	5E-04	14	0.27303754266211605	0.30440315601934337	0.2887203493407297
5	0.005	5E-05	10	0.2606655290102389	0.302876049885467	0.28177078944785294
6	0.005	5E-05	14	0.2794368600682594	0.3196742173581064	0.2995555387131829
7	0.005	5E-04	10	0.2815699658703072	0.30465767370832275	0.29311381978931494
8	0.005	5E-04	14	0.26493174061433444	0.3003308729956732	0.2826313068050038
9	1E-03	5E-05	10	0.23506825938566553	0.23593789768388904	0.23550307853477728
10	1E-03	5E-05	14	0.2150170648464164	0.20386866887248664	0.20944286685945152
11	1E-03	5E-04	10	0.2930887372013652	0.24051921608551793	0.2668039766434416
12	1E-03	5E-04	14	0.21715017064846417	0.18376177144311528	0.20045597104578972

Complete results 4B - Accuracy

n	LR	WEIGHT_DECAY	ALPHA	Cartoon Acc	Sketch Acc	Mean Acc
1	1E-02	5E-05	0.1	0.3391638225255973	0.23542886230593027	0.2872963424157638
2	1E-02	5E-05	0.5	0.0780716723549488	0.18732501908882668	0.13269834572188774
3	1E-02	5E-05	0.8	0.12798634812286688	0.22473911936879612	0.1763627337458315
4	1E-02	5E-04	0.1	0.19368600682593856	0.3321455841180962	0.2629157954720174
5	1E-02	5E-04	0.5	0.08916382252559726	0.2316110969712395	0.1603874597484184
6	1E-02	5E-04	0.8	0.12926621160409557	0.15245609569865107	0.1408611536513733
7	0.005	5E-05	0.1	0.5149317406143344	0.38788495800458134	0.4514083493094579
8	0.005	5E-05	0.5	0.11006825938566553	0.15118350725375412	0.1306258833197098
9	0.005	5E-05	0.8	0.11646757679180887	0.23237465003817764	0.17442111341499325
10	0.005	5E-04	0.1	0.5179180887372014	0.19216085517943496	0.35503947195831814
11	0.005	5E-04	0.5	0.1296928327645051	0.2028505981165691	0.1662717154405371
12	0.005	5E-04	0.8	0.14462457337883958	0.21354034105370323	0.17908245721627142
13	1E-03	5E-05	0.1	0.26791808873720135	0.2957495545940443	0.2818338216656228
14	1E-03	5E-05	0.5	0.420221843003413	0.26291677271570374	0.34156930785955836
15	1E-03	5E-05	0.8	0.2999146757679181	0.3377449732756427	0.3188298245217804
16	1E-03	5E-04	0.1	0.24701365187713312	0.20717739882921862	0.22709552535317587
17	1E-03	5E-04	0.5	0.36689419795221845	0.15856452023415626	0.26272935909318734
18	1E-03	5E-04	0.8	0.18600682593856654	0.2517179944006108	0.21886241016958868