POLITECNICO DI TORINO

Homework 3

Student name: *Erich Malan*
s267475

Course: *Machine Learning and Deep Learning* – Professor: *Barbara Caputo*
Due date: *June, 2020*

Domain Adaptation

Introduction

The aim of this assignment is the image classification facing the problem of different domains. In general it means that in a context of equal targets, the domain of data used to train the model is different from the one with is tested or then adopted. The purpose is to generate a model that is based on the most general features to be as steady as possible towards different circumstances. In this paper will be treated the use of a simple AlexNet and its derivative DANN version that will in the end finetuned.

Data

0.1. Data exploration. The dataset proposed is the PACS dataset (PACS is the acronym of Photo Art Cartoon Sketch), which is composed of 9991 images subdivided into 4 domains, each presenting 7 categories. Each folder's name corresponds to a different domain and contains the folder labeled as the target containing the images of such category. The images' shape is the same of size [224,224,3], in fact are squared pictures of 224x224 RGB encoded, and so they have 3 channels. The 4 domains presented are 'art_painting', 'cartoon','photo','sketch'. Only 'art_painting' and 'photo' will be used, respectively for the testing and the training part, as target domain and source domain. The labels whereas are integers encoded but the categories correspond to: human, horse, guitar, dog, house, giraffe and elephant. It must be said that the two datasets present the same distribution as it can be seen in fig. 2 and in fig.3 but they have differents sizes, the source dataset is composed of 1670 images while the target one with 2048. As it is easy to notice the paintings look more colorful and they are obviously less natural.

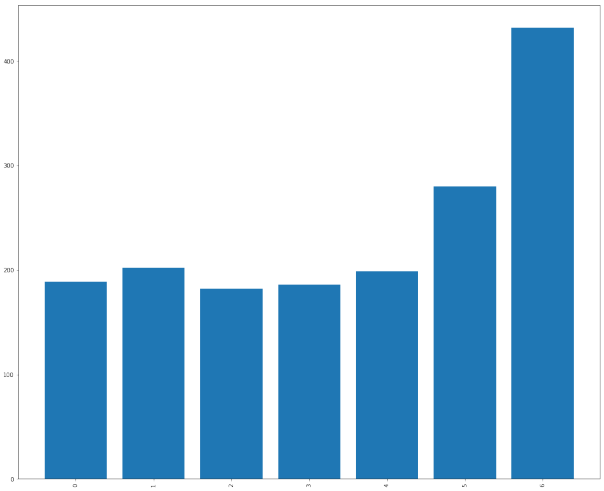

Figure 1: *Distribution of source classes*

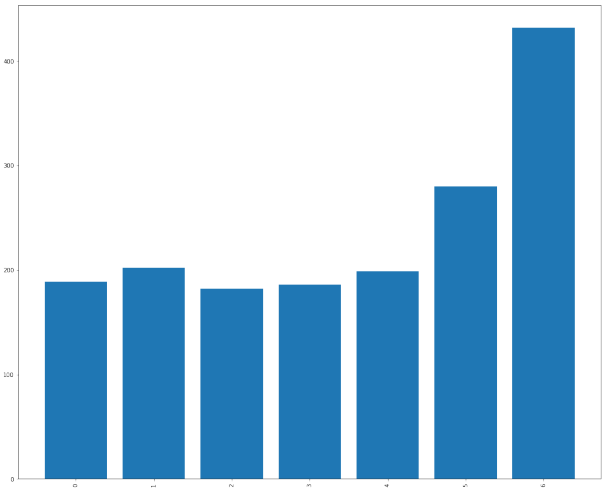

Figure 2: *Distribution of target classes*

In the figures below are presented the content of the two datasets.

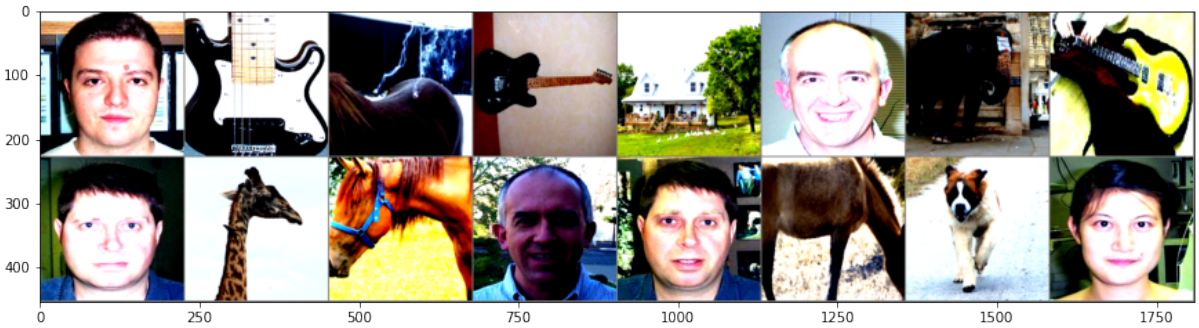

Figure 3: *Some random images of the source dataset*

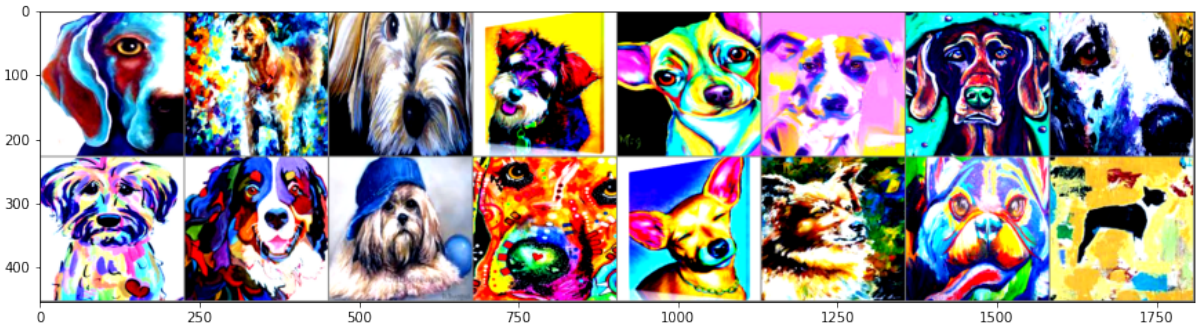

Figure 4: *Some images of the target dataset*

Data preparation. Accordingly to the assignment requirements data must be loaded by using ImageFolder dataloader. Two differents sets are loaded into the source dataset (which the net must be trained on) and the target dataset that is used to test the model which correspond to 'Photo' and 'Art_painting' respectively. As requirement no validation is provided. Each image is loaded into the torchvision imagefolder dataset, that handles the image processing with the following pipeline transformation:
1) Resize(256)
2) CenterCrop(224)
3) ToTensor()
4) Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

CenterCrop and ToTensor are particularly important to match AlexNet inputs.

Domain Adaptation

Alexnet. The first part is a basic attempt to perform a classification, practically it results in training an alexnet (previously pretrained on ImageNet) with the 'Photo' dataset to test it then on the 'art_painting' dataset. The hyperparameters are given and are resumed along the table 1. The last classification layer of the net is modified to fit the 7 categories that must be predicted, every layer get optimized along the training.

BATCH SIZE	LR	MOMENTUM	WEIGHT DECAY	NUM EPOCHS	STEP SIZE	GAMMA
256	1e-2	0.9	5e-5	30	10	0.1

The net starts with an high accuracy over the source set, at the very first step it starts with an accuracy of 0.70, but even nicer is the second step where the accuracy hops to 88%. It is a considerably nice result flanked by a low loss that starts from 0.65 and falls down to 0.12 at the very next step. In the end the loss of the training set decreases to 0.0062 while the accuracy reaches the 92%. But by checking the results over the target sets the situation is completely different. Even if not requested I was curious about the loss and the accuracy over the test set so a validation part has been performed at the end of each epoch, and while the accuracy remains pretty stable or slightly increases the loss rapidly increases and doubles the start point in few epochs. The real test presents an accuracy of 47% much lower (almost the half) of the training found. If we consider that 1/7 = 0,14, then the 14% would mean a random guess score, but it obtains 3 times more so is not that bad result, but it must be also said that is achieved for the most part by the pretraining on ImageNet. (Other personal experiments without transfer learning shown 0.14-0,22 accuracy which are almost the random guess threshold in case of a uniform distribution of targets).

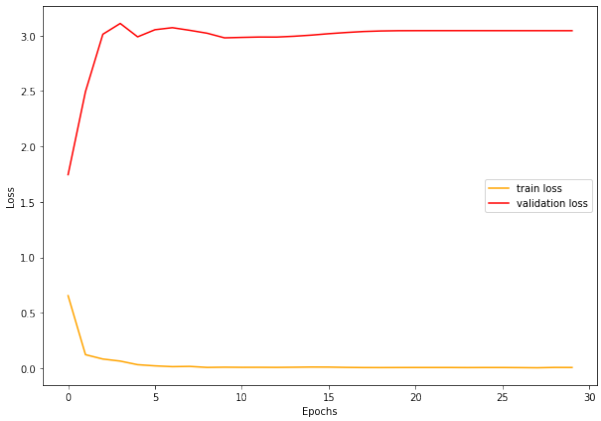

Figure 5: *Alexnet (default) Loss lineplot*

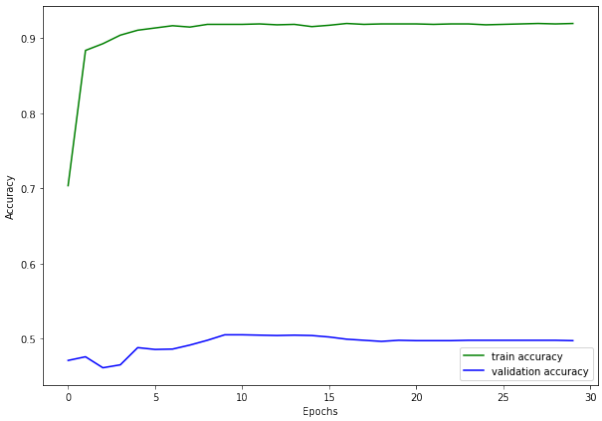

Figure 6: *Alexnet (default) Accuracy lineplot*

DANN. The DANN in this case is a modified version of AlexNet, it is composed by the same Convolutional layers, but for what concerns the classificator there are two different streams or branches. The first one is the same as before, it gets trained to predict the correct labels, while the second one share the same structure, initial weights and biases, but it ends with only 2 nodes that represent the 2 possible domains. Moreover the backpropagation of the net in case of domain classification is slightly different, it need an addictional hyperparameter ALPHA which is multiplied to the negative gradient, this is intended to move the SGD away from extracting dissimilar features across domains. ALPHA in practical is intended to maximise the domain loss. The forward method of DANN is defined as follows: if the ALPHA parameter is given it trains the

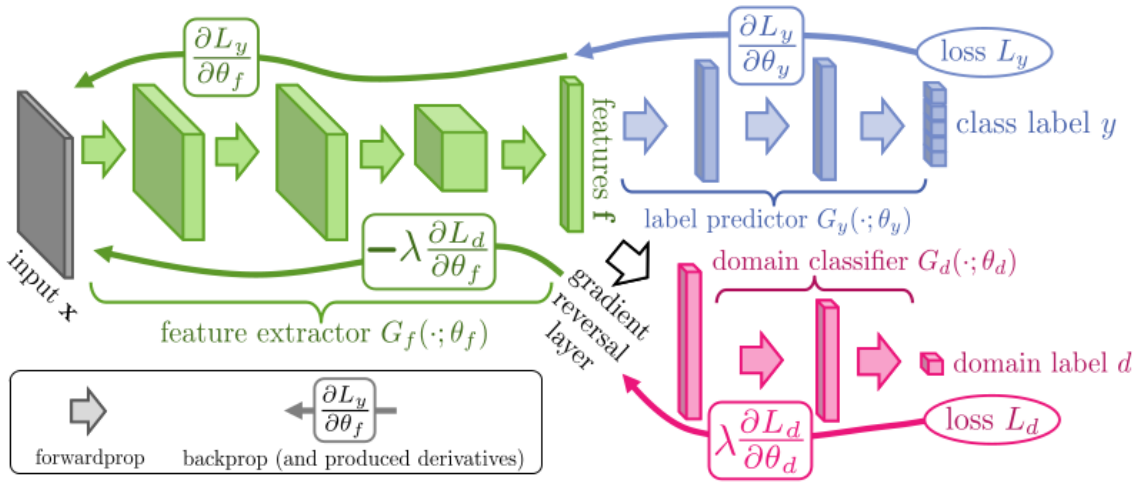

Figure 7: *DANN alexnet structure*

domain classifier along to the feature extractor and it computes the gradient reversal layer, otherwise it performs the label classification. The transfer learning undergo a little modification because of the new structure: in the domain classification branch the same classification branch weights and biases are loaded.

The train pipeline is organized as follows: the net is trained to predict the photos' labels and the first loss is computed, then it forwards the photos but this time the labels are corresponding to the domain, furthermore it is trained with the 'art_paintings' across the labels of the corresponding domain (in this case 0/1 are used because of the binary distinction across photo and art sets). Finally the 3 losses are summed up and backpropagation is performed. The training loss is substantially low in the order of $3 * 10^{-3}$, and the various accuracies even reach the 100% as is depicted in fig. 9 and in fig. 10. This means that the model is not working as expected because the domain loss should be maximized balancing the common loss, further description are explained after the finetuning.

Even if the validation loss is similar to the training one, the resulting test accuracy is worst than the normal AlexNet's one: 38% of accuracy over the 'art_paintings' test, which is not that high especially because by definition the net is trained also with the test set, which is a kind of cheat. In the two lineplot, differently from the alexnet it can be seen that the loss decreases until epoch 10, both target and domain accuracy during training reach high accuracies this could mean that ALPHA need to be increased.

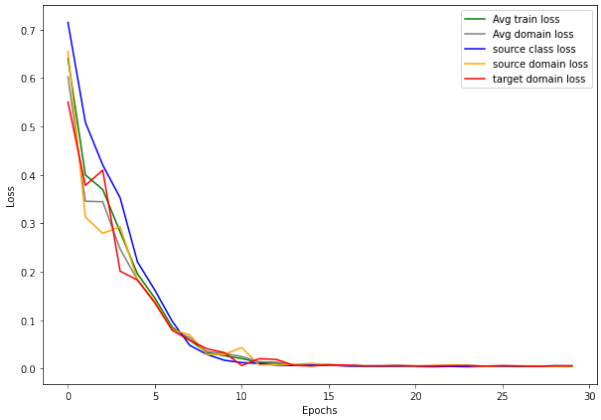

Figure 8: *DANN (default) Loss lineplot*

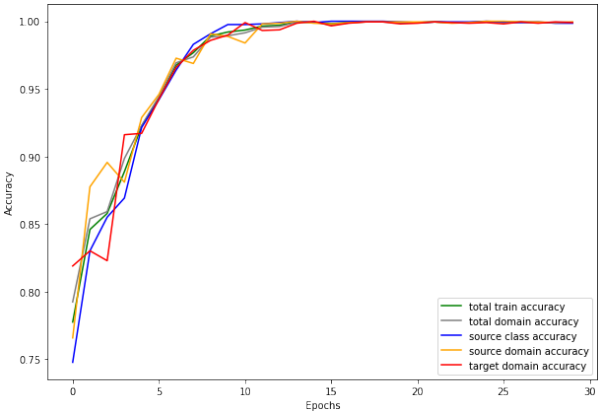

Figure 9: *DANN (default) Accuracy lineplot*

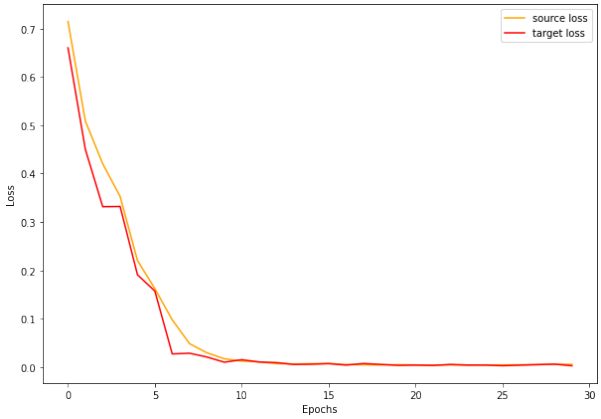

Figure 10: *DANN (default) source and target Loss lineplot*

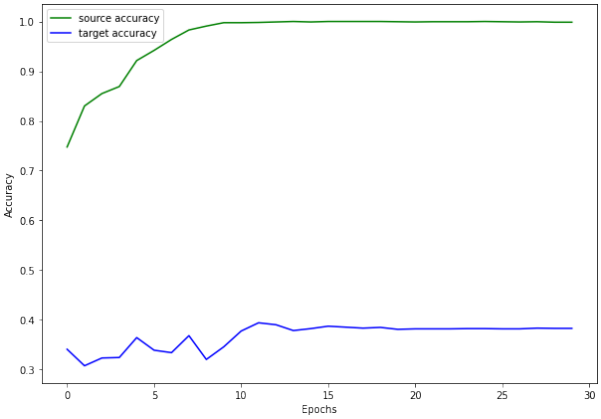

Figure 11: *DANN (default) source and target Accuracy lineplot*

	Train loss	Validation loss	Train Accuracy	Test Accuracy
AlexNet	0.0062	3.042	0.92	0.47
DANN	0.0052	0.0050	1	0.38

In conclusion the loss of the DANN is better because it as an higher training (it iterates 2 times the source, and the biggest difference comes with the validation loss which in the first case was not trained over the target dataset too) but without hyperparameters optimizations the simple alexnet results as more efficient.

Hyperparameters optimization

To finetune the models I provided two differents search paths. An extensive grid search was almost unfeasible because considering 4 hyperparams with 3 possibilities would lead to 3^4 models to be trained and tested. The two approaches used are:
1) keep the best search the next
2) random search

Their approaches to find max accuracy are completely different, the second one take advantage of random hyperparameter picked from reasonable intervals (for example the Learning Rate value is picked randomly from 10^{-3} to 10^{-5}) to cover different points over the 4-dimensional space. While the first approach search for the local minima of each hyperparameters then fixes it and search for another one. It is a little bit stiff and by pruning the space exclude the possibility of other minimum points.

Alexnet optimization.

Incremental accuracy. Best accuracy obtained 49%
Params:

'LR'	0.01
'WEIGHT_DECAY'	5e-5
'STEP_SIZE'	10
'GAMMA'	0.1

Learning Rate	Weight Decay	Step Size	Gamma	Test Accuracy
1e-1	5e-5	10	0.1	0.42
1e-2	5e-5	10	0.1	0.47
1e-3	5e-5	10	0.1	0.46
1e-2	5e-4	10	0.1	0.47
1e-2	5e-5	10	0.1	0.47
1e-2	5e-6	10	0.1	0.46
1e-2	5e-5	15	0.1	0.47
1e-2	5e-5	10	0.1	0.49
1e-2	5e-5	5	0.1	0.48
1e-2	5e-5	10	0.05	0.48
1e-2	5e-5	10	0.3	0.46
1e-2	5e-5	10	0.5	0.47

Random search. Best accuracy obtained 50%

'LR'	0.001
'WEIGHT_DECAY'	0.0005
'STEP_SIZE'	10
'GAMMA'	0.135

The best accuracy is achieved throught the random search of 50% while the incremental approach leads the model to 49%, but it must be said that it shares the same params of attempt 2 and 5 that get only 47%.

DANN optimization.

Incremental accuracy. Best accuracy obtained 50%
Params:

'LR'	0.01
'WEIGHT_DECAY'	5e-4
'STEP_SIZE'	10
'GAMMA'	0.05
ALPHA	0.5

Learning Rate	Weight Decay	Step Size	Gamma	ALPHA	Test Accuracy
1e-1	5e-5	10	0.1	0.1	0.41
1e-2	5e-5	10	0.1	0.1	0.47
1e-3	5e-5	10	0.1	0.1	0.24
1e-2	5e-4	10	0.1	0.1	0.491
1e-2	5e-5	10	0.1	0.1	0.47
1e-2	5e-6	10	0.1	0.1	0.487
1e-2	5e-4	15	0.1	0.1	0.47
1e-2	5e-4	10	0.1	0.1	0.49
1e-2	5e-4	5	0.1	0.1	0.48
1e-2	5e-4	10	0.05	0.1	0.495
1e-2	5e-4	10	0.3	0.1	0.48
1e-2	5e-4	10	0.5	0.1	0.49
1e-2	5e-4	10	0.05	0.05	0.49
1e-2	5e-4	10	0.05	0.1	0.49
1e-2	5e-4	10	0.05	0.5	0.50

Random search. Best accuracy obtained 44%
Params:

'LR'	0.0001
'WEIGHT_DECAY'	5e-5
'STEP_SIZE'	10
'GAMMA'	0.125
ALPHA	0.45

Conclusion

In this case the incremental search revealed to be the most performing, while the random search unfortunately reached very low accuracies (avg 30% with a best value of 44%). By the way both model obtained 50% of accuracy over the target dataset, the DANN showed to be much more sensible to the learning rate, it presented an accuracy drop for a Learning Rate of 10^{-3} (the model was learning too slowly). Another aspect to consider is the weight decay, the incremental search revealed that an higher cost for larger weights was better. It turned out by looking at the loss plots that after that only

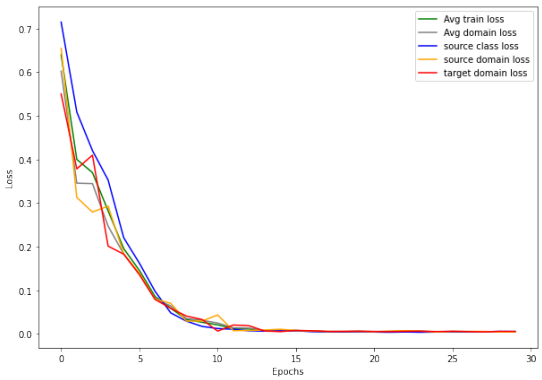

Figure 12: *DANN (default) loss plots*

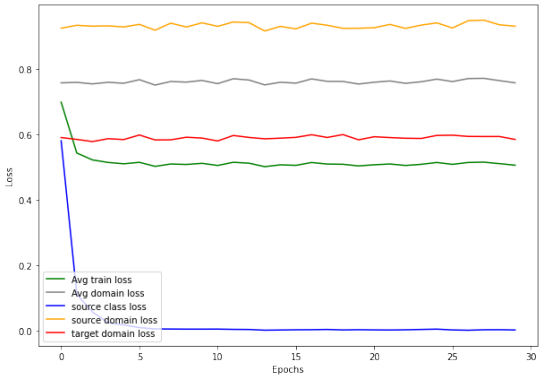

Figure 13: *DANN (default) loss plots after weight decay increment*

the label loss was being minimized while the others presented a low variance sawtooth function, this practically means that the DANN was not anymore trying to maximize the domain loss by extracting only on the features that distinguish the labels. The domain loss is not increasing but is neither decreasing so the DANN started more or less following what has been designed for, especially with higher ALPHA values which balance more the target and domain losses.

By concluding the DANN implementation did not appeared to be a valuable option because it takes more time to train but is also true that the results searching with incremental accuracy search method appear to be more stable. Moreover as enounced previously this domain adaptation was not properly pure because of the use of the domain classification the same set of testing part, with different sets it could be even worst because the finetuning could be slightly overfitting. Probably by implementing different evaluations on the remaining domain datasets will lead to more steady and general results across domains.

net	LR	WEIGHT DECAY	STEP SIZE	GAMMA	ALPHA	Test Accuracy
AlexNet	0.001	5e-4	10	0.135		50%
DANN	0.01	5e-4	10	0.05	0.5	50%