Homework 3
Marco Di Nepi 277959

The dataset

PACS is a dataset composed by 7 classes and 4 domains:

- Photo (1670 images)
- Art painting (2048 images)
- Cartoon (2344 images)
- Sketch (3929 draws)

Exploiting the ImageFolder class the data can be easily read:

```
P = ImageFolder(DATA_DIR+"/photo", transform=train_transform)
A = ImageFolder(DATA_DIR+"/art_painting", transform=eval_transform)
C = ImageFolder(DATA_DIR+"/cartoon", transform=train_transform)
S = ImageFolder(DATA_DIR+"/sketch", transform=train_transform)
```

The goal of this homework is to train the network using the images contained in Photo to predict the labels of the images in Art and Painting. The results obtained with a standard AlexNet will be compared to the ones obtained using a Domain Adaptation neural network. For this task no data augmentation is applied.

Implementing the Model

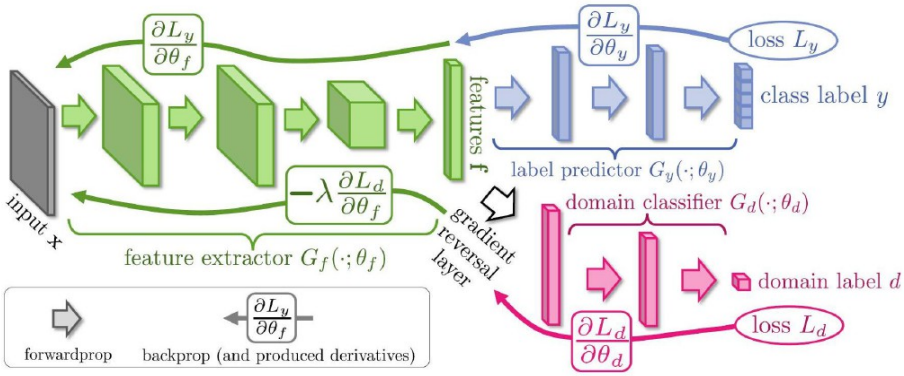
In order to implement the second model we can start from the code of AlexNet and add a second classifier,called gd,in the init function. This classifier has the same architecture of the fully connected layers, the only difference is the last layer, since there are only two classes to be predicted. Gd will be trained in order to distinguish between the source domain and the target domain.

The next step is to copy the weights of the first classifier trained on ImageNet to gd with:

```
model.gd_classifier[1].weight.data = model.classifier[1].weight.data
model.gd_classifier[1].bias.data = model.classifier[1].bias.data
```

Finally, the forward function is changed. If it receives the parameter alpha, data goes to gd and gradient reversal is applied, otherwise data goes to the standard classifier and the forward function is the same as AlexNet's.
Alpha will be optimized in one of next steps.

Our goal is to maximize the error of Gd exploiting gradient reversal generating domain invariant features.



The parameter update formula is

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda\frac{\partial L_d^i}{\partial \theta_f}\right)$$

Domain Adaptation

Firstly, I used the pretrained AlexNet without adaptation.
With the default hyperparameters (LR = 1e-3, 30 Epochs, Batch_size = 256) it has been obtained a test accuracy on Art and Painting around 48%.

After that, I implemented the training function for DANN. The training has been divided in 3 steps and has been used a third dataloader for Art and Painting to be used during the training phase.

```
#Step 1
outputs = net(s_images)
loss1 = criterion(outputs,s_labels)
loss1.backward()

#Step 2
outputs = net(s_images,ALPHA)
loss2 = criterion(outputs,zeros)
loss2.backward()

#Step 3
outputs = net(t_images,ALPHA)
loss3 = criterion(outputs,ones)
loss3.backward()
```

Where s_images and t_images are from the source and the target respectively.
Doing this, we are trying to confuse the extracted features, so that notwithstanding the images come from different domains, they should be classified indipendently.

The test accuracy with dann, using the same hyperparameters and alpha = 0.04 reaches 49.8%.
I tried also with smaller values of alpha and the results were slightly worse than this one, on the other hand with very high values (for example 0.5) the losses go to nan and the accuracy dramatically decreases. In fact, Alpha and learning rate should have a sort of balance to avoid si phenomena, so a large lr should stay with a big alpha and viceversa.
This means that with a good value of alpha there is small improvement with respect to the AlexNet without DANN but the optimization of this hyperparameter is a crucial point.
For this phase, validation has not been done to not use again the Art painting set.

Cross Domain Validation

To validate Hyperparameters without using the labels of Art and painting, we can use the two remaining domains, Cartoon and Sketch. For each set of hyperparameters, I did the following steps

Without adaptation:

- Train the network on Photo
- Test the results on Cartoon
- Test the results on Sketch
- Take the average between the 2 accuracies and if it is better than the previous one, save the hyperparameters in a list.

With adaptation

- Train one model with source = Photo and target = Cartoon
- Test on cartoon
- Train a second model with Source = Photo and target = Sketch
- Test on Sketch
- Take the average between the 2 accuracies and if it is better than the previous one, save the hyperparameters in a list.

Once the best hyperparameters are found, a new network is trained on Photo and finally tested on Art.

Since the process is very slow and colab automatically stops if it takes too much time, I decided to reduce the number of epochs to 10 and manually validate the hyperparameters with for loops. Every combination between
LR = [0.001,0.01,0.02,0.1], Batch Size = [256,512], Alpha = [0.01,0,03,0.05,0.1] have been tested. Step_size and gamma have not been taken into account for the grid search in order to avoid too many iterations.

For the standard AlexNet the obtained set of hyperparameters are LR = 0.01 and BATCH_SIZE = 512. The final test accuracy is 51%.

In the DANN case the best found value of alpha is 0.03 while the learning rate and the batch size are 0,01 and 256 respectively. With these hyperparameters the

accuracy on Art and Painting is more than 52%.

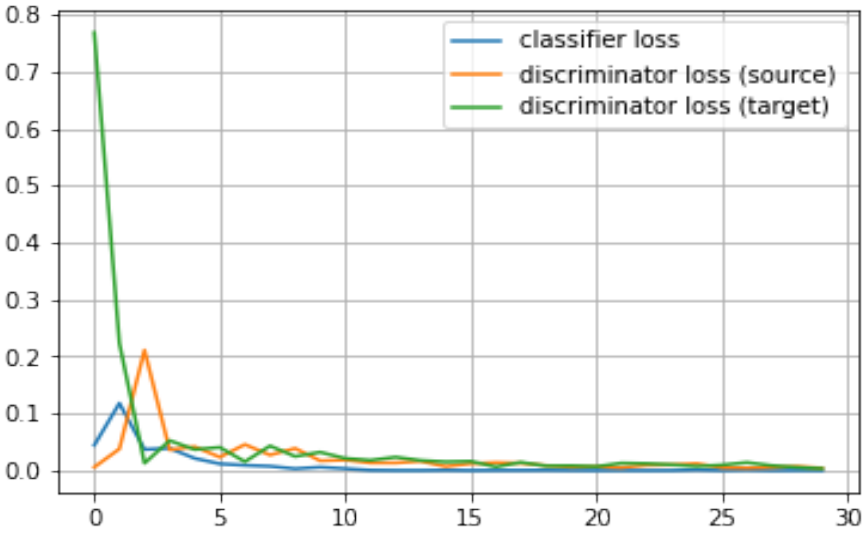
LR / Batch Size	**256**	**512**
1	0,27 | 0,25	0,21 | 0,18
0,01	0,28 | 0,31	0,21 | 0,21
0,02	0,29 | 0,28	0,34 | 0,35
0,1	0,16 | 0,19	0,16 | 0,19

Tabella 1: Accuracies of Cartoon and Sketch without Dann

It can be noticed that the validation on different domains is not easy and don't always lead to good results. For example, in case of sketch, the loss goes to nan with alpha = 0,1 and the accuracy is generally lower. We can conclude that the hyperparameters found can have a bias towards certain domains and taking the mean accuracy is not the best choice.

Both the results are better than the ones without validation and even this time using DANN there has been an improvement of one percentage point.

The Graph below shows the trend of the losses during the training phase for the latter model. In this case the discriminator loss (high in the firt epochs) has not a positive trend and becomes very small, this happens because the DANN classifier is still trying to distinguish as much as possible the domains and since in the PACS dataset the domains are very different it is not an hard task.



In conclusion domain adaptation is not trivial: in our case the improvements obtained are very limited and in order to generate domain independent features

what we have done is not enough. On the other hand, also the process of validation and the choiche of the other domains is very important to achieve better results.