POLITECNICO DI TORINO

MACHINE LEARNING AND DEEP LEARNING

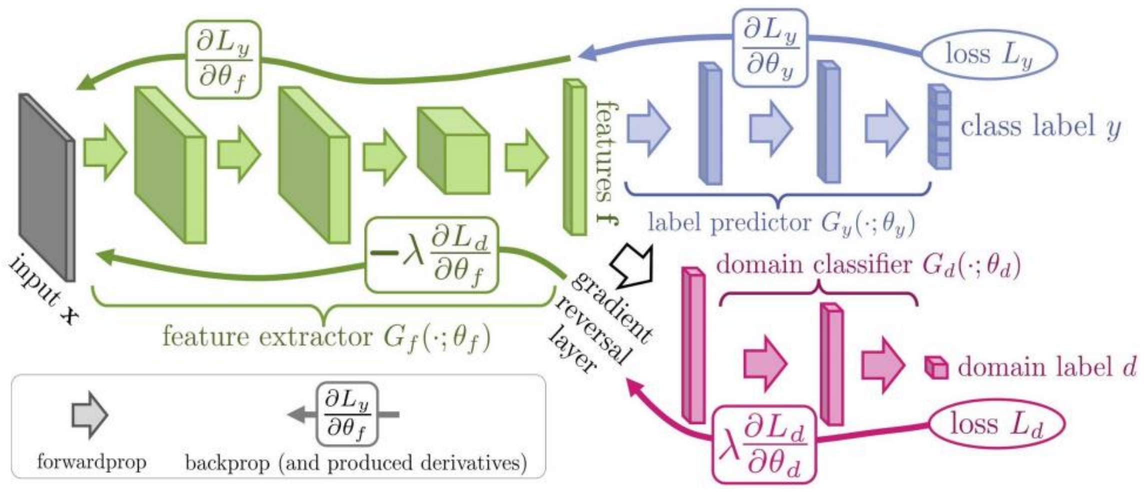
ALBERTO MARIA FALLETTA - S277971

# HOMEWORK 3 – REPORT

# DEEP DOMAIN ADAPTATION

## Introduction

A huge amount of labeled data is needed in order to have a high-performance network but unfortunately this is not always possible. Domain Adaptation allows the use of available labeled data of similar nature but different domain in order to enlarge the training dataset avoiding data distribution shift (domain adaptation is in fact the act of learning in presence of a data distribution shift between training and test datasets). In this assignment we will focus on deep domain adaptation that means combining domain adaptation and deep feature learning within one training process therefore looking for features able to discriminate independently from the domain.


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

In this homework we will implement DANN, a Domain Adaptation algorithm, on the PACS dataset using AlexNet (not the network presented above which is a small network proposed in the original paper for digit recognition).

Caveat: due to the small dimension of the data involved in the assignment, the numerical results are highly volatiles, the one reported are relative to an average run.

## Dataset

PACS is a small domain generalization dataset consisting of 9991 images divided in four domains (Photo with 1670 images, Art painting with 2048 images, Cartoon with 2344 images, Sketch with 3929 images).

The pictures below show how images are divided between the four domains. We can immediately notice how the dataset is unbalanced by looking not only inside one specific domain (where, for example, horse class in Sketch domain has more than 10x the images of house class in the same domain) but also across domains (where, for example, person class in Art painting domain has 2.8x the images of the same class in Sketch domain).

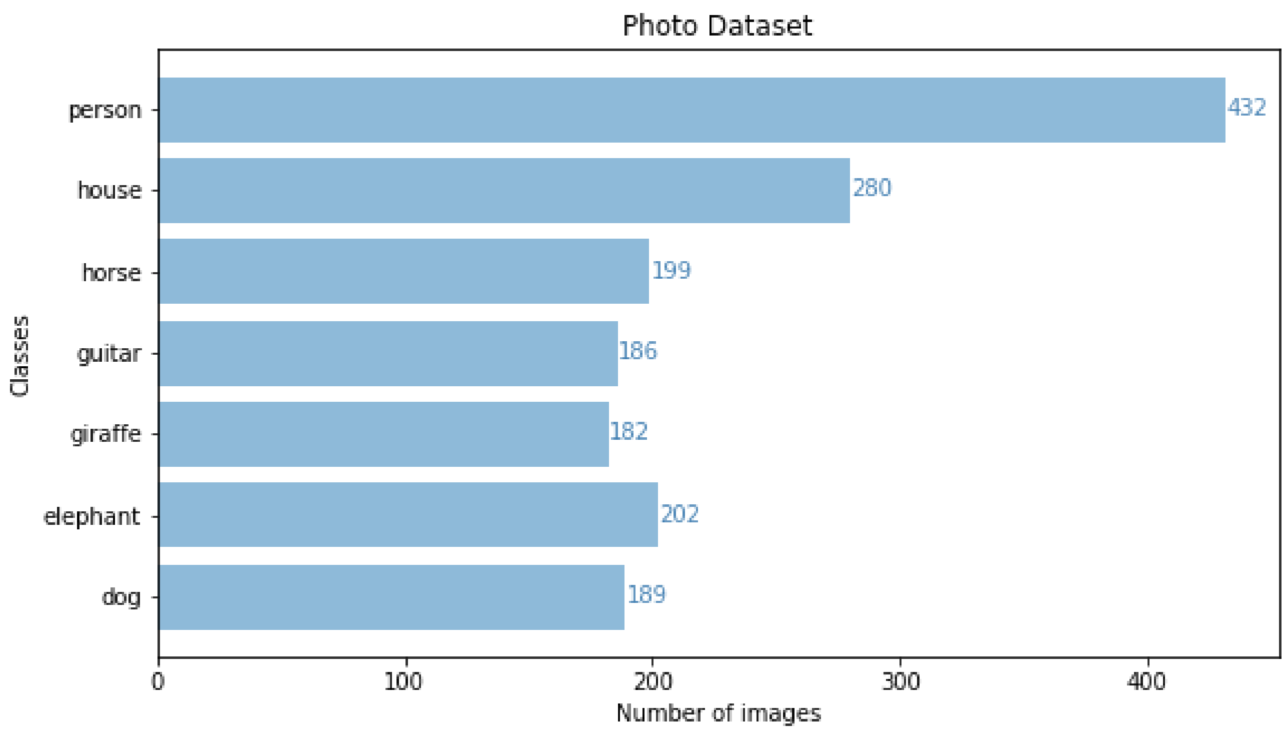


*Figure 1 Class distribution of the images in Photo Dataset*

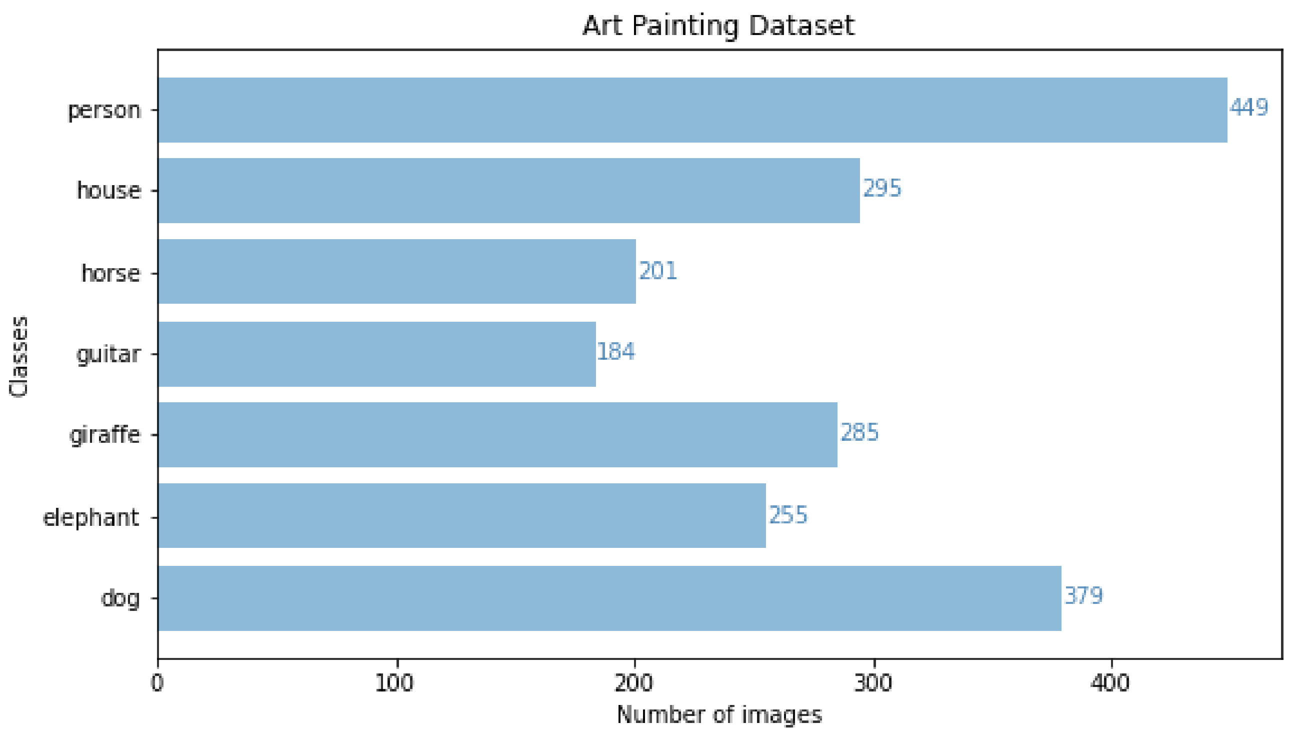


*Figure 2 Class distribution of the images in Art Dataset*

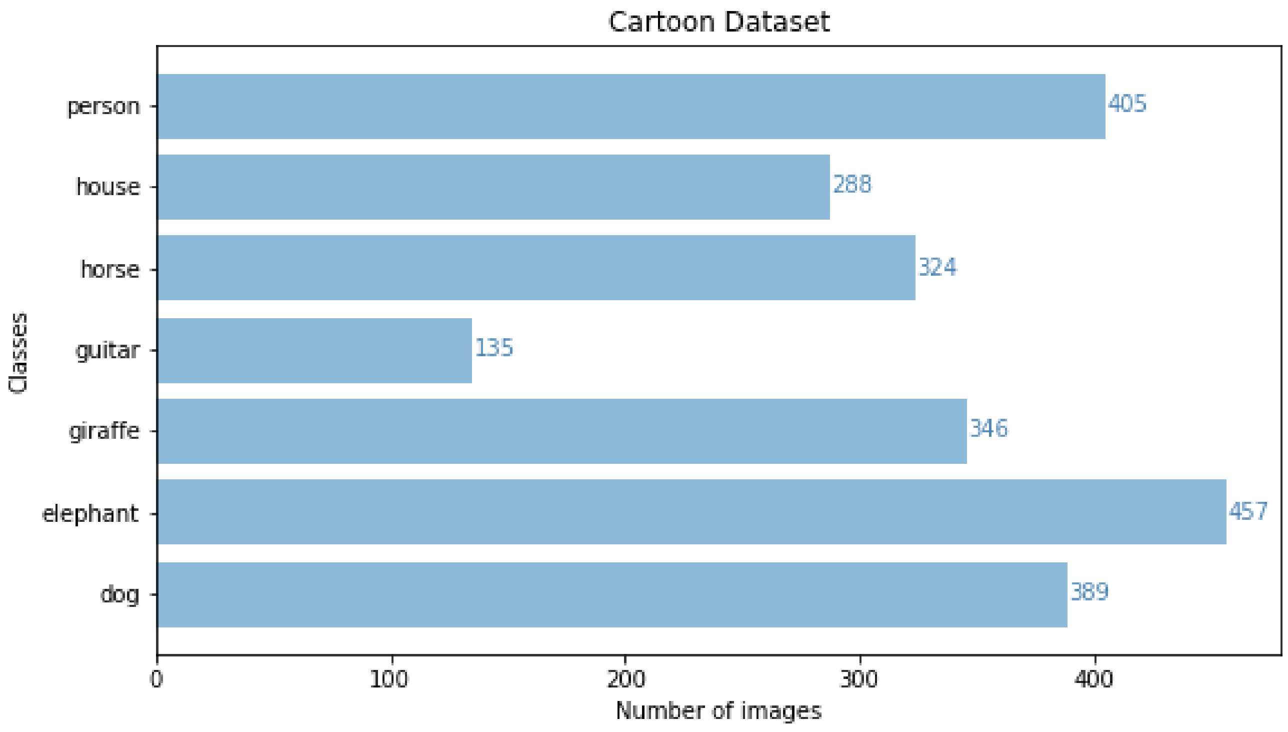


*Figure 3 Class distribution of the images in Cartoon Dataset*

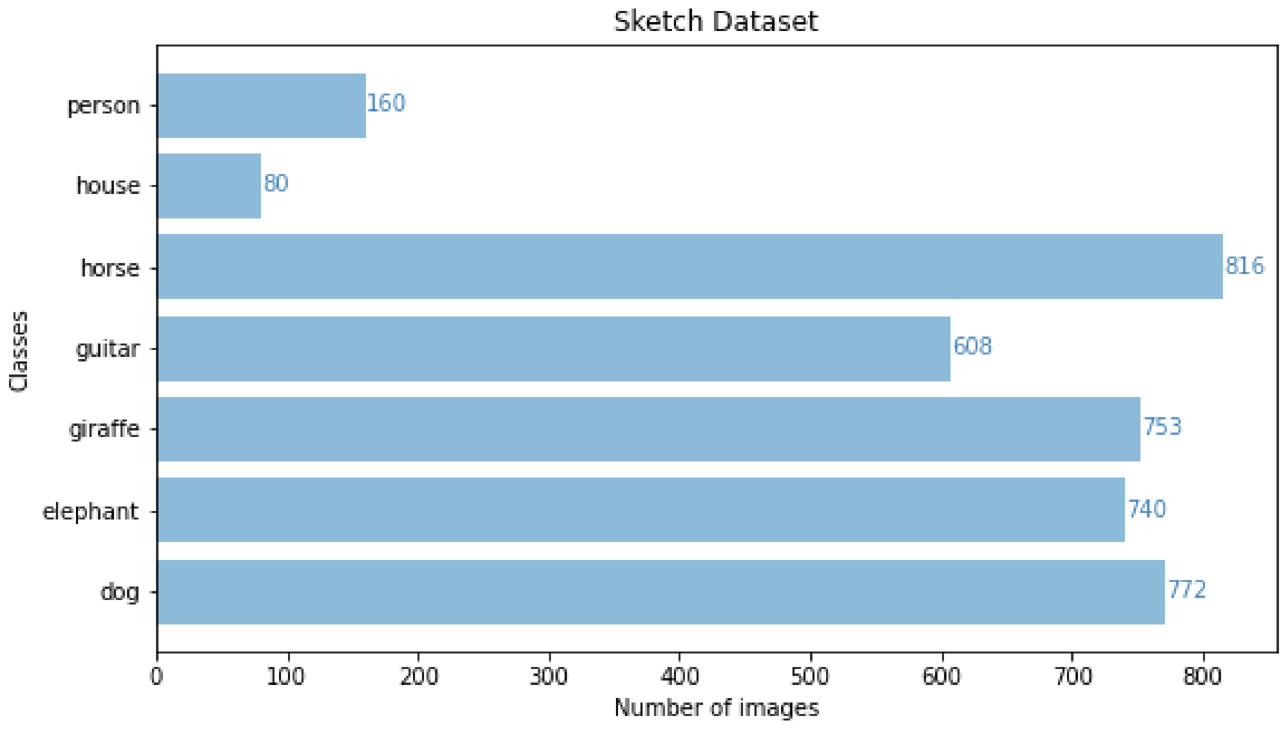


*Figure 4 Class distribution of the images in Sketch Dataset*

## Domain Adaptation

Differently from the plots of the previous assignment, where the one on the right was the validation accuracy over the epochs, in this assignment instead, the plots on the right show the accuracy on the test dataset changing over the epochs. This is because validation is an open problem in Domain Adaptation, therefore, in the first part of the homework I will train the model with the template's default parameters since, as we have seen in the previous homework, it achieves good accuracy scores with pretrained weights, while in the second part I will focus on validation and parameters search.

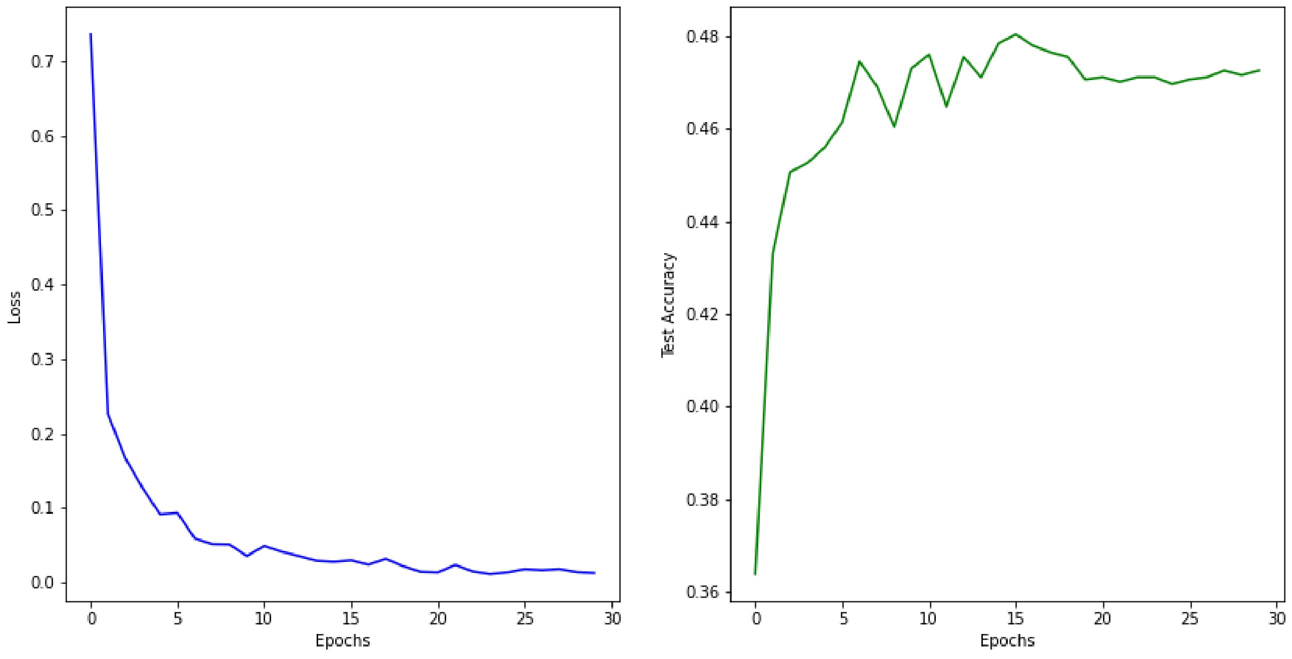


*Figure 5 Loss and test accuracy of AlexNet without Domain Adaptation (template's default parameters)*

Training the model without Domain Adaptation (i.e. training a normal AlexNet) on Photo and testing on Art painting leads to accuracy of around 0.47 by the 30th epoch.

I then implemented the training with Domain Adaptation by jointly focusing on the label predictor branch of the modified AlexNet (called "classifier" in the code) and the domain discriminator branch (called "domain_classifier" in the code) with the same previous "out-of-the-box" parameters and a static (constant throughout the whole training) adaptation factor alpha with value 0.2 (value bigger than which the loss would start considerably increasing after a certain number of epochs).

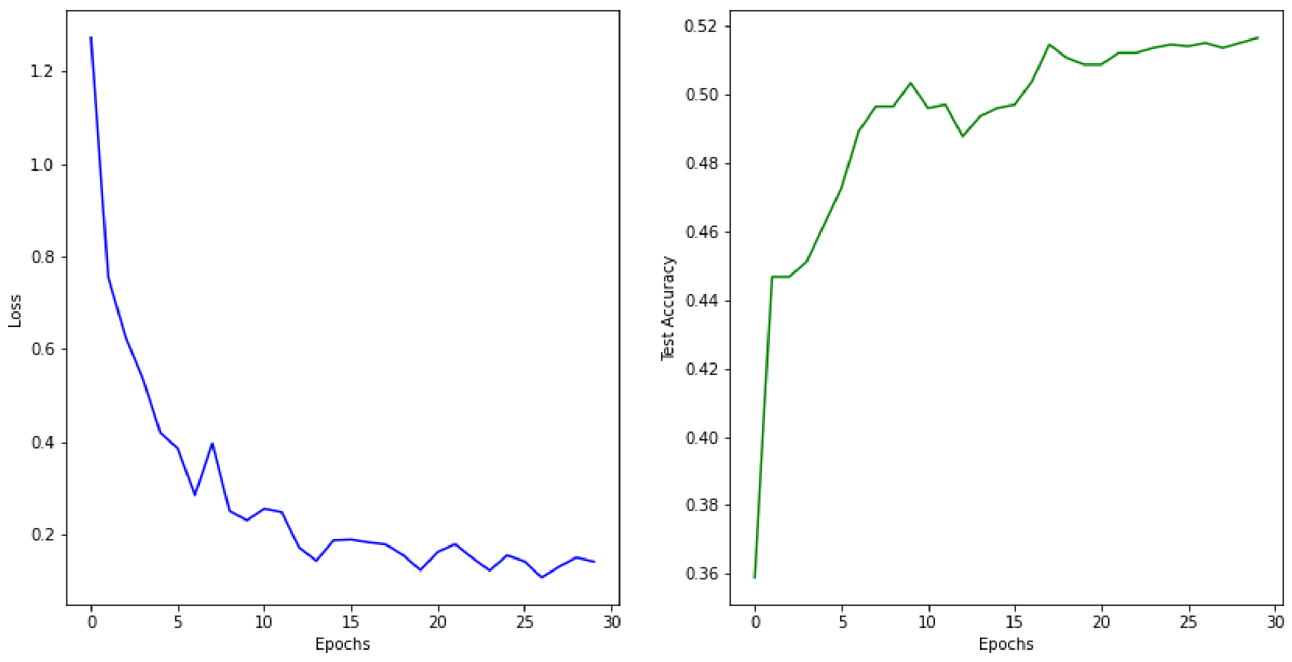


*Figure 6 Loss and test accuracy of AlexNet with Domain Adaptation (template's default parameters)*

Training on Photo and testing on Art painting with this implementation of the Domain Adaptation led to an improvement of the previous accuracy of 0.47, achieved without adaptation, of around four percentage points, achieving 0.51 by the 30th epoch.

## Cross Domain Validation

Performing validation directly on Art painting set would be "cheating" since we would be tuning hyperparameters according to the test set that, as the name suggests, is not the validation set.

There are many possible parameters to tune such as batch size, learning rate, number of epochs, step size, gamma and alpha (for domain adaptation part).

I focus my parameter search on learning rate, step size, and batch size, keeping constant the number of epochs at an intermediate value and the value of alpha which I later searched[1]:

| Learning Rate = 1e-3<br>Number of Epochs = 35 | | Batch Size | |
|---|---|---|---|
| | | 256 | 512 |
| Step Size | 20 | μ = 0.232, σ = 0.006 | μ = 0.190, σ = 0.006 |
| | 30 | μ = 0.256, σ = 0.010 | μ = 0.244, σ = 0.011 |

| Learning Rate = 1e-2<br>Number of Epochs = 35 | | Batch Size | |
|---|---|---|---|
| | | 256 | 512 |
| Step Size | 20 | μ = 0.253, σ = 0.012 | μ = 0.294, σ = 0.018 |
| | 30 | μ = 0.313, σ = 0.006 | μ = 0.310, σ = 0.004 |

Running a grid search on Photo without Domain Adaptation and averaging the accuracy achieved on Cartoon and the one achieved on Sketch make us find the best accuracy score achieving hyperparameters ('batch_size': 256, 'lr': 0.01, 'num_epochs': 35, 'step': 30) that achieve an averaged accuracy of 0.313.

Using these hyperparameters to train the model without adaptation on Photo and testing it on Art painting make us achieve an accuracy of around 0.49 which is an improvement of 2 percentage points with respect to the run with "out-of-the-box" hyperparameters.

[1] Test Accuracy is the result of two runs. It is therefore expressed by expected value and standard deviation.

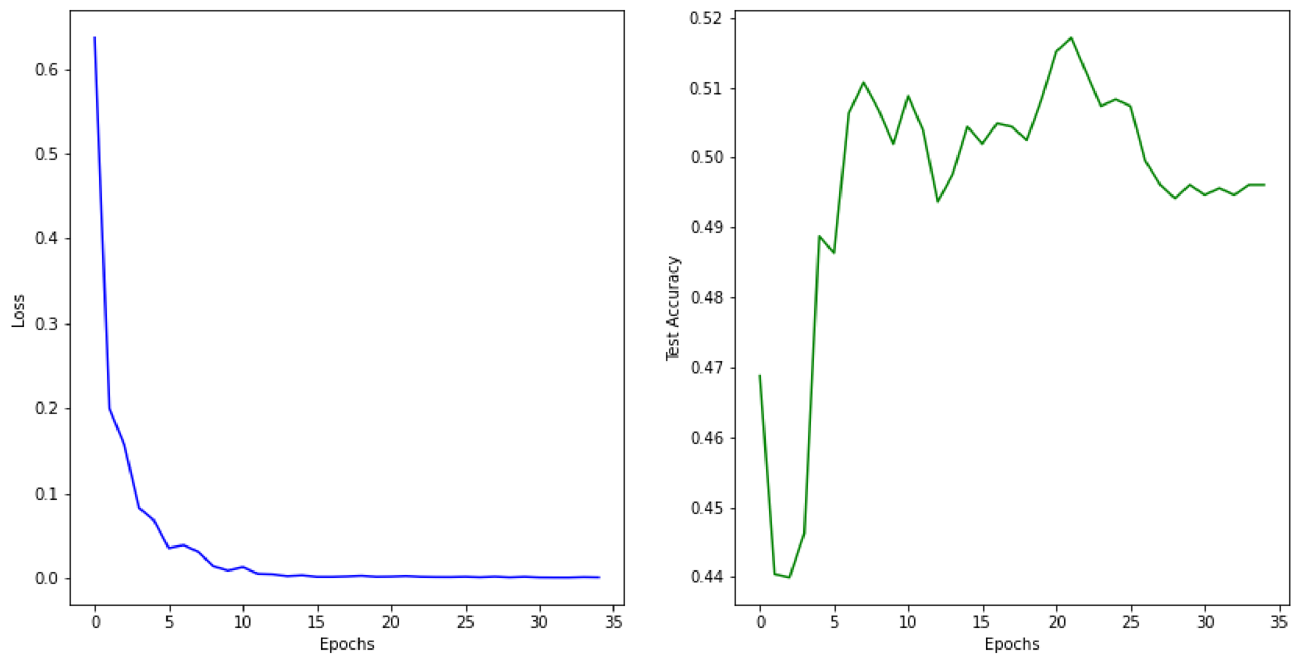


*Figure 7 Loss and test accuracy of AlexNet without Domain Adaptation (best parameters from search)*

I then performed a parameter grid search on Photo with Domain Adaptation, averaging the accuracy achieved on Cartoon and the one achieved on Sketch, keeping alpha constant at 0.2:

| Learning Rate = 1e-3<br>Number of Epochs = 35 | | Batch Size | |
|---|---|---|---|
| | | 128 | 256 |
| Step Size | 20 | μ = 0.583, σ = 0.277 | μ = 0.370, σ = 0.004 |
| | 30 | μ = 0.728, σ = 0.146 | μ = 0.355, σ = 0.002 |

| Learning Rate = 1e-2<br>Number of Epochs = 35 | | Batch Size | |
|---|---|---|---|
| | | 128 | 256 |
| Step Size | 20 | μ = 0.283, σ = 0.000 | μ = 0.283, σ = 0.000 |
| | 30 | μ = 0.283, σ = 0.000 | μ = 0.283, σ = 0.000 |

So I found the best accuracy score achieving hyperparameters for the search being ('batch_size': 128, 'lr': 0.001, 'num_epochs': 35, 'step': 30) that achieve a surprising averaged accuracy of 0.728.

In this search I did not use 512 as possible batch size parameter since it makes the gpu go out of memory.

Using these hyperparameters to train the model with adaptation on Photo and testing it on Art painting make us achieve an accuracy of around 0.52 which is an improvement of 1 percentage point with respect to the run with “out-of-the-box” hyperparameters.

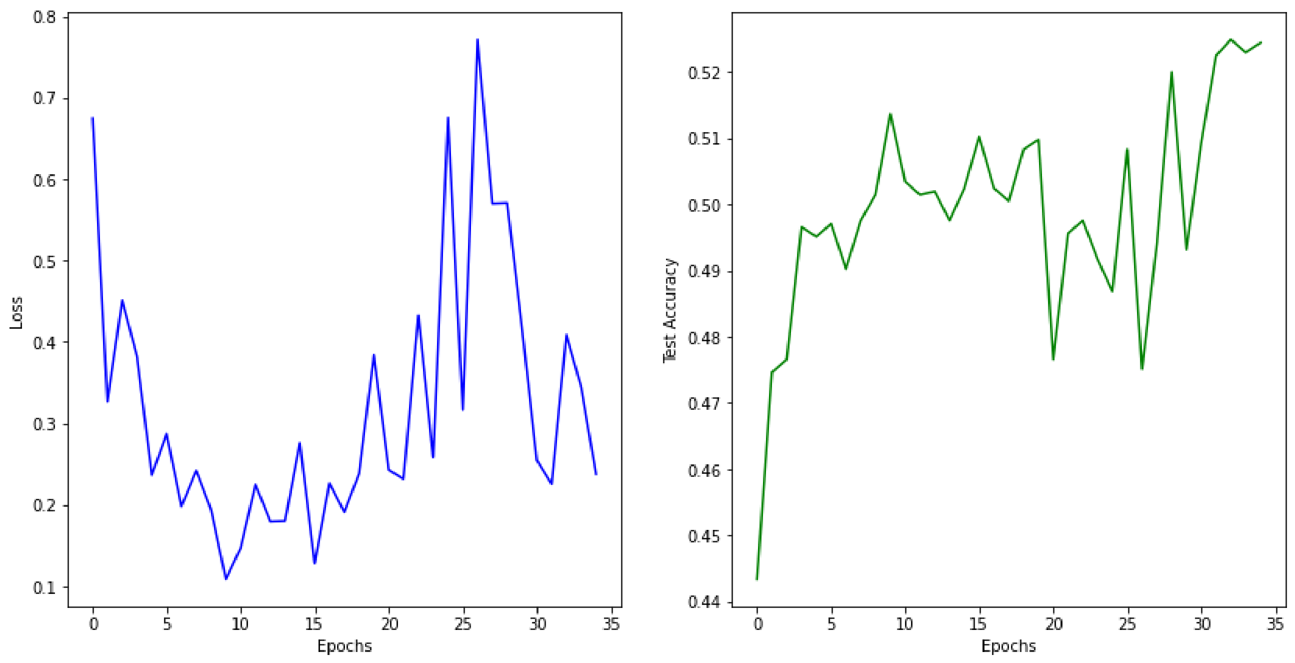


*Figure 8 Loss and test accuracy of AlexNet with Domain Adaptation (best parameters from search)*

At this point I performed a manual search for alpha, experimenting with multiple values both statics and dynamics, training on Photo and testing on Art painting with the best test accuracy achieving parameters:

| Static Alpha | Test Accuracy |
|---|---|
| 0.10 | $\mu = 0.492$, $\sigma = 0.014$ |
| 0.15 | $\mu = 0.507$, $\sigma = 0.009$ |
| 0.20 | $\mu = 0.515$, $\sigma = 0.010$ |
| 0.30 | $\mu = 0.495$, $\sigma = 0.007$ |
| 0.40 | $\mu = 0.332$, $\sigma = 0.010$ |

| Dynamic Alpha | Test Accuracy |
|---|---|
| Exponentially increasing[2] | $\mu = 0.501$, $\sigma = 0.033$ |
| Linearly increasing from 0 to 1.0[3] | Loss: NaN |
| Linearly increasing from 0 to 0.5[4] | Loss: NaN |

[2] In the code called “Dynamic adaptation factor # 1”
[3] In the code called “Dynamic adaptation factor # 2”
[4] In the code called “Dynamic adaptation factor # 3”

Using the exponential adaptation factor proposed in the paper in order to suppress noisy signal from the domain classifier at the early stages of the training procedure, as well as implementing a custom linear increasing exponential adaptation factor unfortunately did not lead me to better results, or the accuracy would never be above 0.52 or passed a certain threshold the loss would always skyrocket degrading significantly the model's performance.

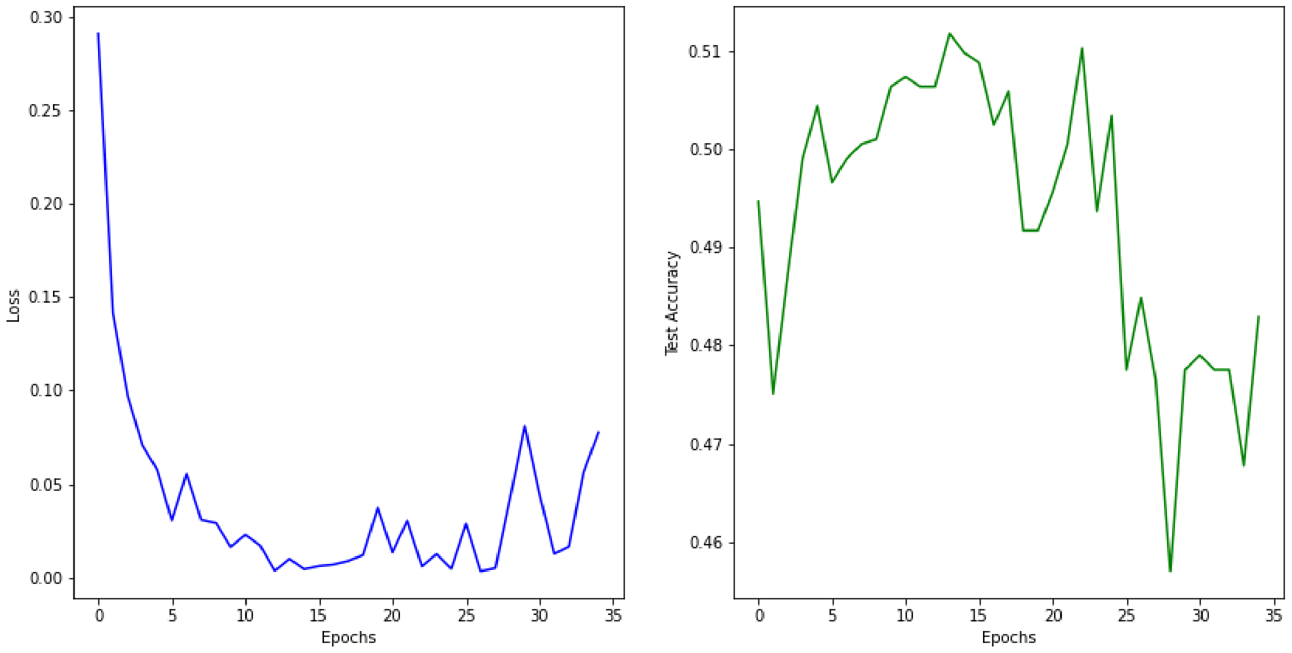


*Figure 9 Loss and test accuracy of AlexNet with Domain Adaptation (best parameters and exponential adaptation factor)*

The final plot shows how the losses change during the last training phase (best hyperparameters and domain adaptation). More in details it is possible to see how the blue one (Source classifier loss) is in a downward trend, since the optimizer's task is to minimize it, while the red and orange ones (respectively target domain loss and source domain loss) are in a slight upward trend, since the optimizer tries to maximize those quantities due to the reversal layer. The upward trend is subtle because alpha is still 0.2 which is a small value.

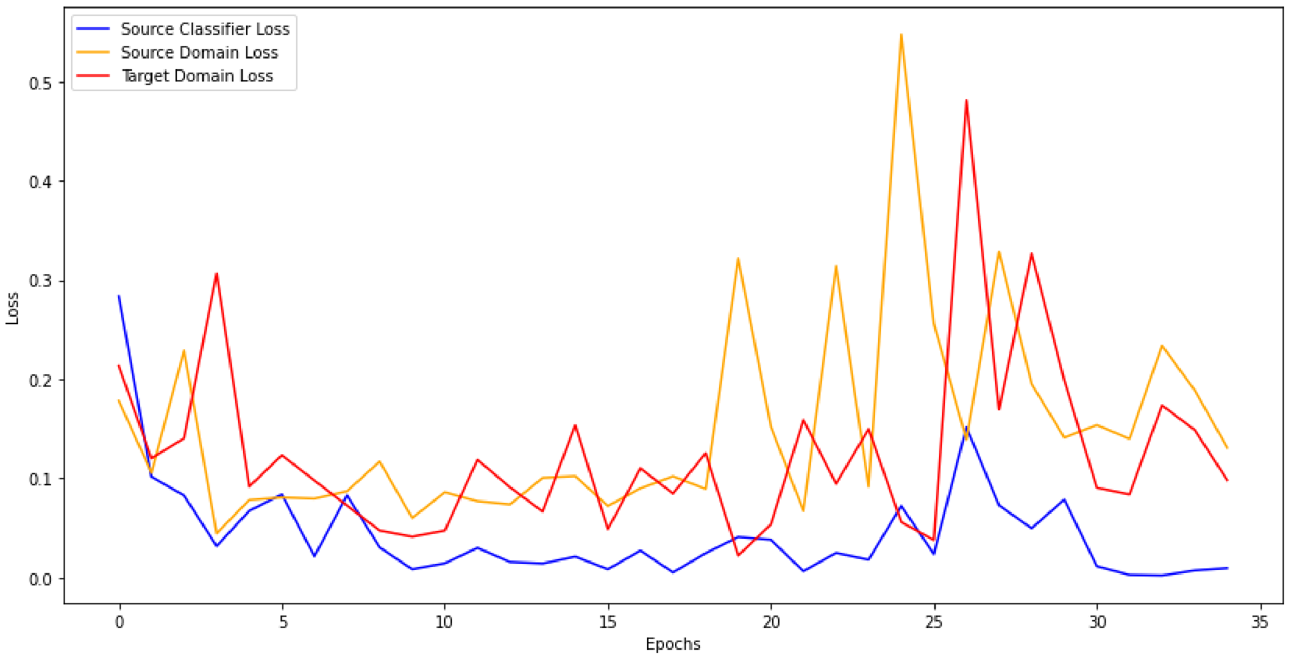


*Figure 10 The trends of the three losses*