Machine Learning and Deep Learning
Politecnico di Torino

Homework 3
Silvia Giammarinaro

Student ID: 269991

June 15, 2020

1 Introduction

In this homework, we analyze the Domain Adaptation algorithm using Alexnet and the PACS dataset. PACS stands for Photo, Art painting, Cartoon and Sketch, which are the different domains presented in the dataset. Each domain is evenly divided in 7 classes. Our task is to learn to distinguish classes taken from different domains. As we can see, the domains are highly imbalanced, so we need to take this into account when iterating over the batches.

Figure 1: House samples from PACS dataset

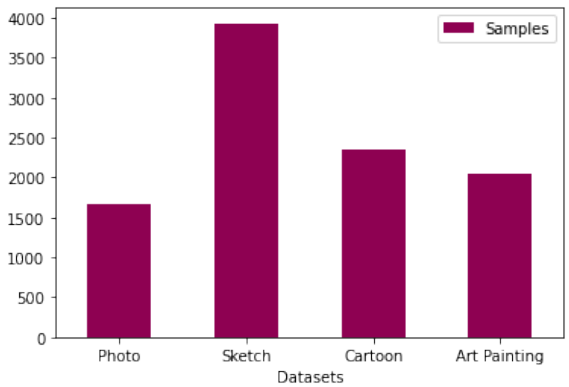

Figure 2: Dataset distribution

The domain adaptation is one of the unsupervised learning algorithms, given a sample distribution, you teach the network to predict samples taken from a different domain. Domain adaptation is an open problem in deep learning, as an example we can consider self driving cars. The smart car should be able to identify a pedestrian in all possible context (weather condition, viewpoint, landscape). Another example could be classifying objects having as training data the images taken for advertisement (i.e. professional photos posted on Amazon) and as test data images from everyday life, taken with a non-uniform background and low quality cameras. Domain adaptation tries to align the two or more different data distribution in order to be able to perform the task independently from the context, the domain. In section 3 we will report all the definitions needed to understand this algorithm.

Figure 3: Domain adaptation application in self driving cars

Figure 4: Office caltech dataset samples

2 Implementing the model

The network architecture is called DANN: domain adversial neural network. We can divide the architecture in four parts:

- the feature extractor G_f composed by the convolutional layers;
- the feature extractor G_y which predicts the source classes labels;

- the domain classifier G_d which determines if the input comes from the source domain or the target domain;
- the gradient reversal layer which inverts the gradients of the domain classifier multiplying them by a factor *alpha*.

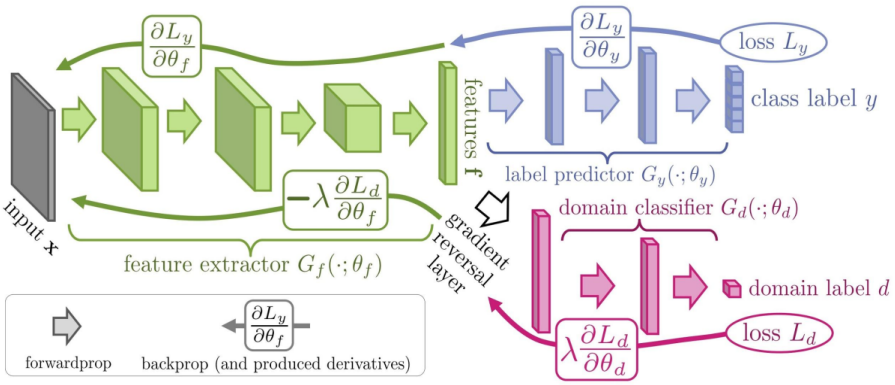

Figure 5: Net architecture

To define this architecture we start from the definition of the AlexNet model avalaible in Github. G_f is made of the convolutional layers, G_y and G_d fully connected layers. The only difference between G_y and G_d is the output shape: the branch G_y has to predict the class label (seven possible values) and the branch G_d has to predict the domain's label (four alternatives). Every branch is associated with a cross entropy loss, L_y and L_d. We report below the branches G_y and G_d.

```
(class_classifier): Sequential(
  (0): Dropout(p=0.5, inplace=False)
  (1): Linear(in_features=9216, out_features=4096, bias=True)
  (2): ReLU(inplace=True)
  (3): Dropout(p=0.5, inplace=False)
  (4): Linear(in_features=4096, out_features=4096, bias=True)
  (5): ReLU(inplace=True)
  (6): Linear(in_features=4096, out_features=7, bias=True)  )
(domain_classifier): Sequential(
  (0): Dropout(p=0.5, inplace=False)
  (1): Linear(in_features=9216, out_features=4096, bias=True)
  (2): ReLU(inplace=True)
  (3): Dropout(p=0.5, inplace=False)
  (4): Linear(in_features=4096, out_features=4096, bias=True)
  (5): ReLU(inplace=True)
  (6): Linear(in_features=4096, out_features=2, bias=True)  ) )
```

After inizializing our architecture, we preload the weights from the AlexNet model used for the Imagenet competition. As last step, we rewrite the forward function, which defines if the data has to go in the branch G_y or G_d and if the hyper-parameter *alpha* is being used. Here we report the implementation of the forward function.

```
def forward(self, x, alpha=None):
  x = self.features(x)
  x = self.avgpool(x)
  x = torch.flatten(x, 1)
  if alpha is not None:
    reverse_feature = ReverseLayerF.apply(x, alpha)
    discriminator_output = self.domain_classifier(reverse_feature).cuda()
    return discriminator_output
  else:
    class_outputs = self.class_classifier(x).cuda()
    return class_outputs
```

3 Domain Adaptation

Domain adaptation is a type of transfer learning. In a general supervised problem we have a sample vector $\mathcal{X}$ with the associated label vector $\mathcal{Y}$. The task is to predict the labels given the sample distribution $P(X)$, so we want to learn $P(Y|X)$. As in homework one, we divide the dataset in train, validation and test set and run the experiments looking for the best model to predict the label vector. In real world applications, we can generalize the general supervised problem when dealing with different sample distribution and different task.
A domain $\mathcal{D}$ is what defines the sample data, $\mathcal{D} = \{\mathcal{X}, P(X)\}$. A task $\mathcal{T}$ defines the link between the sample data and the labels, the output of the classification task, $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$. A transfer learning problem is defined in the following way: given a source domain $\mathcal{D}_S$ and the learning task $\mathcal{T}_S$ and a target domain $\mathcal{D}_T$ and the learning task $\mathcal{T}_T$, we want to learn the data distributions given $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$.
In this passage we are analysing the case in which $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$, we use Photo as source domain and Art Painting as target domain and the task is to predict the class label. Our problem can be indentified as homogeneus supervised, because we have the same classes in both domains and we are training on source and target domain (cheating).

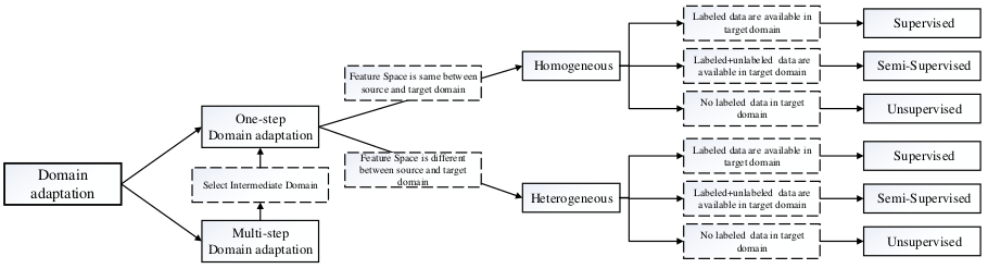

Figure 6: Different types of domain adaptation

The first two experiments consist of comparing the results obtained without and with adaptation. In the first one the branch G_d is not used, that's why alpha is equal to zero. With domain adaptation the training phase of a epoch is divided in three steps:

- forward the source data to the branch G_y to learn the domain labels;
- forward the source data to the branch G_d to learn the domain label;
- forward the target data to the branch G_d.

For every step we compute the loss and we backpropagate it. The loss associated with the domain classifier G_d is taken with the negative sign and it is multiplied by the *alpha* factor. We report here our hyperparameters choices and the losses and accuracies trends. We perform a random search to find the best configurations. Other hyperparameters used for all the experiments presented in this homework are batch size equals 200, SGD as optimizer with momentum equals to 0.9 and weight decay equals to $5*10^{-5}$ and StepLR as scheduler. To validate our models, we use the art panting samples. As a first approach, we choose to validate the model at every epoch with the test set to see how our model is performing. As mention in homework one, the test set should be used wisely and we should not overfit on it, this should be our "real life" experiment and looking for an higher accuracy on it could lead to more weaker performance when dealing with other samples.

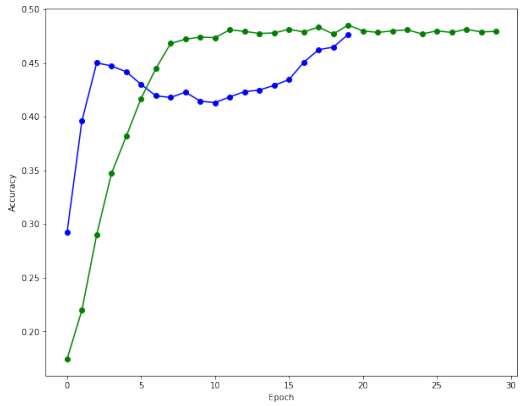

Figure 7: Accuracy per epoch

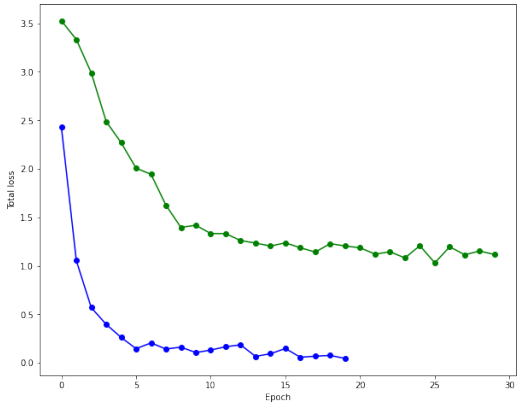

Figure 8: Total loss per epoch

LR	N. Epochs	Step Size	Gamma	Alpha	Test Accuracy	Color
$8 * 10^{-3}$	20	10	0.5	0	0.477	Blue
10^{-3}	30	8	0.25	0.08	0.479	Green

As we can see, when we are not using domain adaptation the training process is quite strange, we do not have a good trend for the accuracy. Instead when using domain adaptation we see a little improvement.

4 Cross Domain Validation

In the previous experiment we validated our model with the test set, so the validation set is equal to the test set. To improve the performances of the previous experiment, we use two different validation sets. We validate our model using the other two domains: cartoon and sketch. After the validation step, we compute the mean of the two validation accuracies and compute the test accuracy on Art painting. We first analyze the case without domain adaptation.

LR	N. Epochs	Step Size	Gamma	Test Acc.	Color
$8*10^{-3}$	25	20	0.5	0.446	Blue
10^{-3}	25	10	0.5	0.36	Green
$8*10^{-3}$	25	10	0.5	**0.46**	Red
$4*10^{-3}$	25	20	0.5	0.457	Magenta
$4*10^{-3}$	25	10	0.1	0.458	Yellow

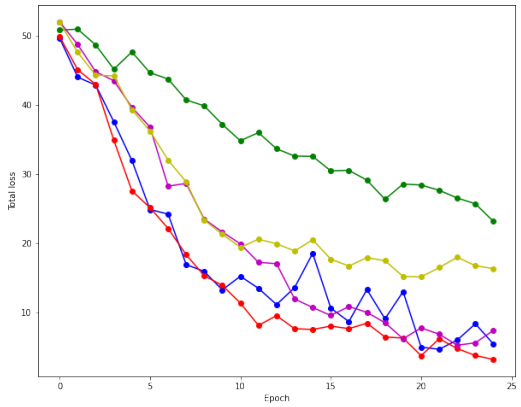

Figure 9: Loss per epoch

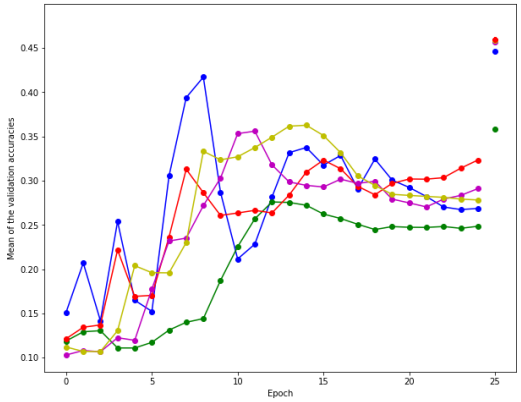

Figure 10: Mean validation accuracies per epoch and test accuracy on art painting

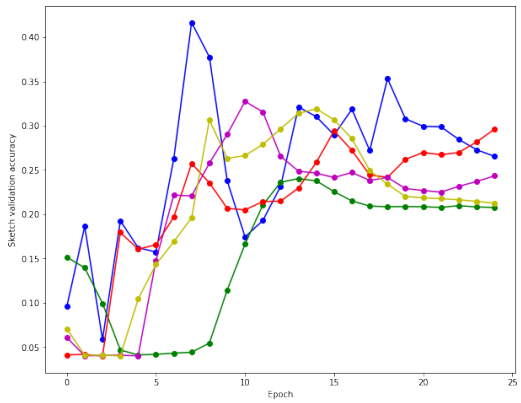

Figure 11: Sketch validation accuracy per epoch

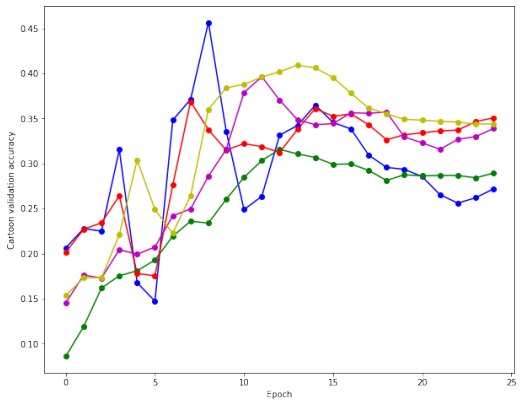

Figure 12: Cartoon validation accuracy per epoch

As we can see we obtain strange trends for both loss and accuracies, this means our learning process is not optimal. In trends reported in yellow, we have a lot of spikes in the plots, which is not what we are looking for. Then we can see the model with the higher mean accuracy gives us the best test accuracy (red model). The best test accuracy will be our baseline for the next experiment.
Moving on to the domain adaptation, we want to prove domain adaptation helps improve our classification task. In this case the domain adaptation is homogeneous unsupervised, we are looking at the target domain labels only for the test phase.

LR	N. Epochs	Step Size	Gamma	Alpha	Test Accuracy	Color
$1*10^{-4}$	20	10	0.5	0.2	0.22	Blue
$5*10^{-3}$	20	10	0.5	0.1	0.47	Green
$8*10^{-3}$	25	10	0.25	0.1	**0.49**	Red
$1*10^{-4}$	25	10	0.25	0.15	0.19	Magenta
$8*10^{-3}$	25	10	0.25	0.1	0.461	Yellow

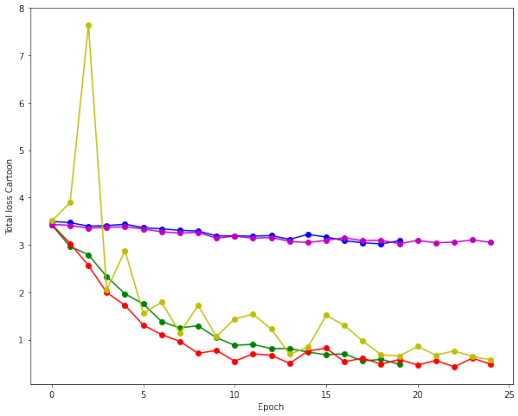

Figure 13: Loss per epoch

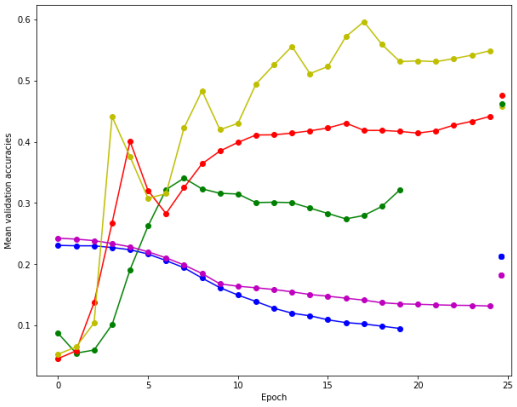

Figure 14: Mean validation accuracies per epoch and test accuracy on art painting

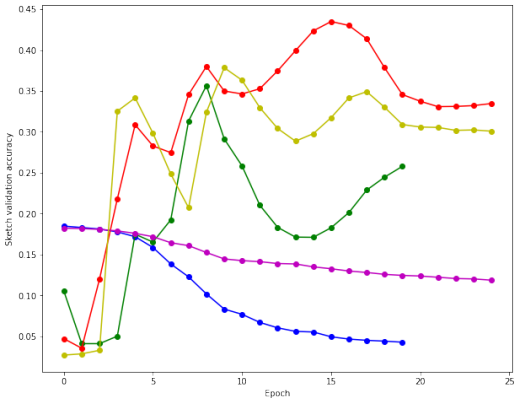

Figure 15: Sketch validation accuracy per epoch

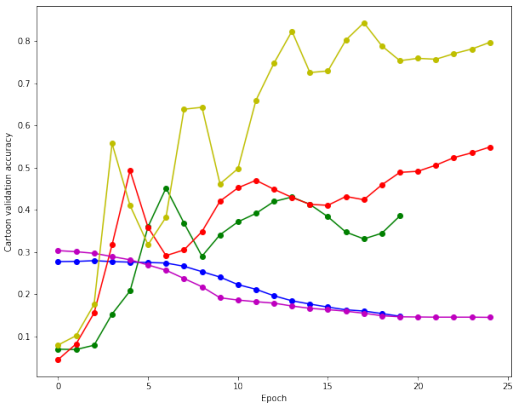

Figure 16: Cartoon validation accuracy per epoch

The best model obtained with domain adaptation overcomes the one obtained in the previous experiment. With domain adaptation, the CNN tries to extract domain-invariant features. Looking for the best model with the highest mean validation accuracy is a good way to find the best model for the art painting domain.

5 References

Mei Wang, Weihong Deng - Deep Visual Domain Adaptation: A Survey (2018)
Israel Istitute of Technology CS236605 course's notes
Deep semantic segmentation among different domains, master thesis's presentation taken from past thesis published on the teaching portal