POLITECNICO DI TORINO

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

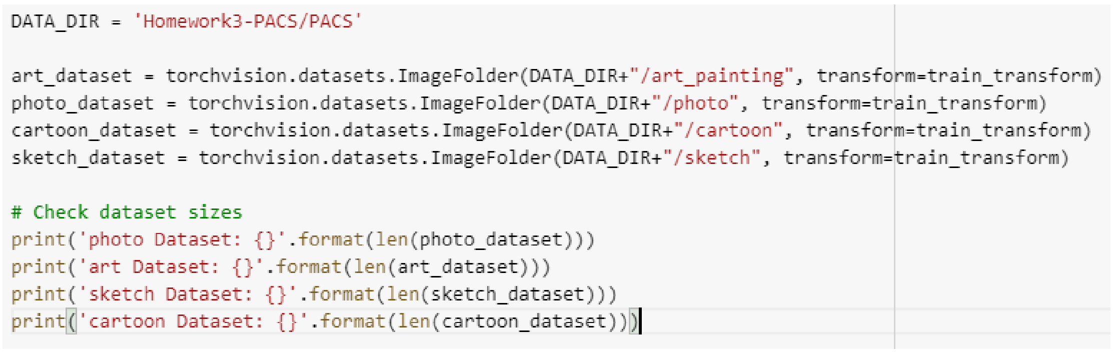
2019-2020

HOMEWORK3

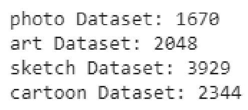
ANTONINO MUSMECI

The goal of this homework is to implement a domain adaptation algorithm. In this homework we are going to use PACS dataset. This dataset consisting of images from photo, art painting, cartoon, and sketch domains.

We use the pytorch ImageFolder class to read the dataset.

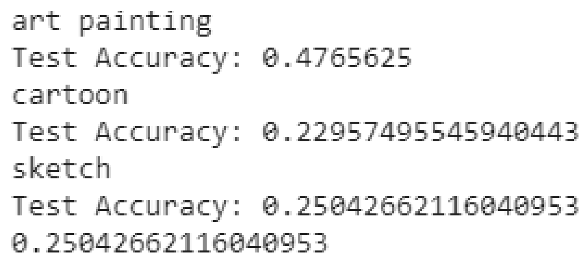


```
DATA_DIR = 'Homework3-PACS/PACS'

art_dataset = torchvision.datasets.ImageFolder(DATA_DIR+"/art_painting", transform=train_transform)
photo_dataset = torchvision.datasets.ImageFolder(DATA_DIR+"/photo", transform=train_transform)
cartoon_dataset = torchvision.datasets.ImageFolder(DATA_DIR+"/cartoon", transform=train_transform)
sketch_dataset = torchvision.datasets.ImageFolder(DATA_DIR+"/sketch", transform=train_transform)

# Check dataset sizes
print('photo Dataset: {}'.format(len(photo_dataset)))
print('art Dataset: {}'.format(len(art_dataset)))
print('sketch Dataset: {}'.format(len(sketch_dataset)))
print('cartoon Dataset: {}'.format(len(cartoon_dataset)))
```



```
photo Dataset: 1670
art Dataset: 2048
sketch Dataset: 3929
cartoon Dataset: 2344
```

We train our model (Alexnet) on one of this dataset and then we test it on another. as we can see, even if we use the pretrained model, if we train our model on the dataset of photo and then we test on the other dataset that came from different domain we get lower accuracy

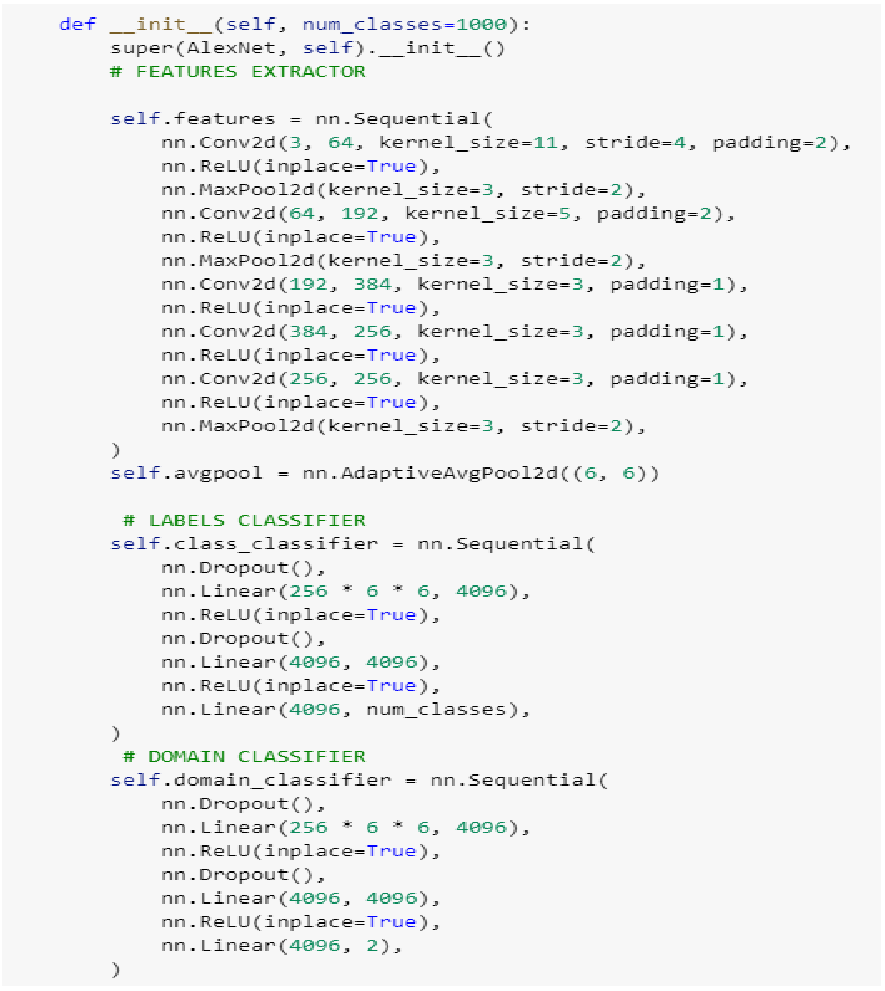


```
art painting
Test Accuracy: 0.4765625
cartoon
Test Accuracy: 0.22957495545940443
sketch
Test Accuracy: 0.25042662116040953
0.25042662116040953
```

In this homework we implement DANN, a Domain Adaptation algorithm, using AlexNet

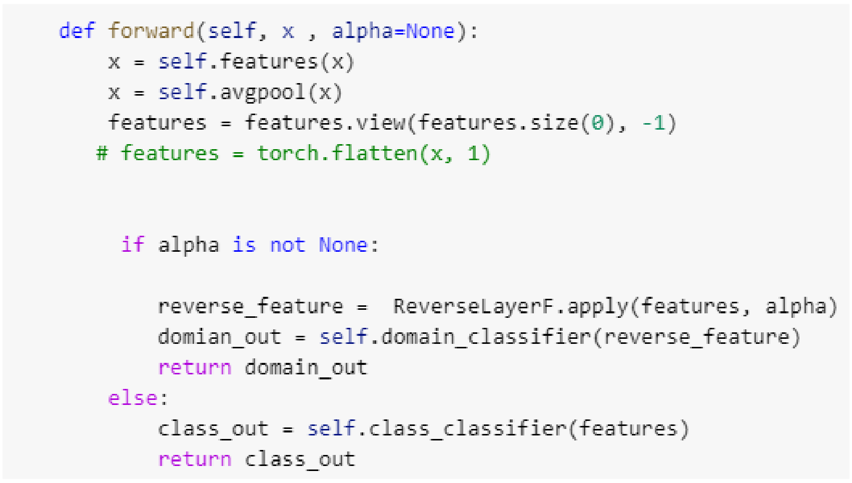
DANN Domain-Adversarial Neural Networks learning approach for domain adaptation, in which data at training and test time come from similar but different distributions. This adaptation behavior is achieved with a feed-forward model by augmenting it with few standard layers and a new Gradient Reversal Layer. We use the alexnet model for label classfier and we add a separate branch with the same architecture of AlexNet fully connected layers for the domain classifier.

To do this we add the new branch domain classifier in the init of alexnet



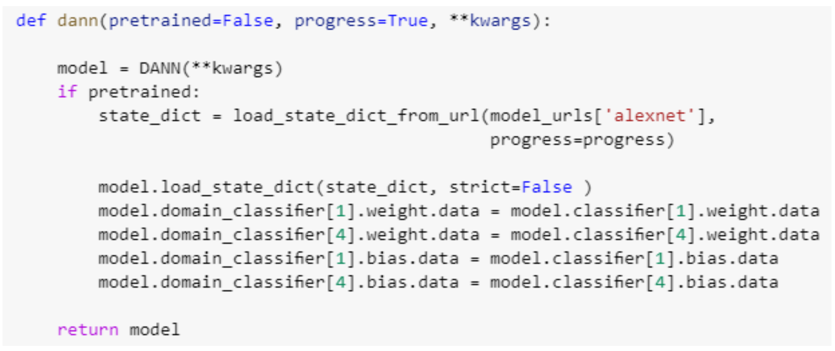
```
def __init__(self, num_classes=1000):
    super(AlexNet, self).__init__()
    # FEATURES EXTRACTOR

    self.features = nn.Sequential(
        nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(64, 192, kernel_size=5, padding=2),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Conv2d(192, 384, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(384, 256, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
        nn.Conv2d(256, 256, kernel_size=3, padding=1),
        nn.ReLU(inplace=True),
        nn.MaxPool2d(kernel_size=3, stride=2),
    )
    self.avgpool = nn.AdaptiveAvgPool2d((6, 6))

     # LABELS CLASSIFIER
    self.class_classifier = nn.Sequential(
        nn.Dropout(),
        nn.Linear(256 * 6 * 6, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(),
        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Linear(4096, num_classes),
    )
     # DOMAIN CLASSIFIER
    self.domain_classifier = nn.Sequential(
        nn.Dropout(),
        nn.Linear(256 * 6 * 6, 4096),
        nn.ReLU(inplace=True),
        nn.Dropout(),
        nn.Linear(4096, 4096),
        nn.ReLU(inplace=True),
        nn.Linear(4096, 2),
    )
```

Then we need to modify the forward function to revert gradient in the layer we have added before



```
def forward(self, x , alpha=None):
    x = self.features(x)
    x = self.avgpool(x)
    features = features.view(features.size(0), -1)
   # features = torch.flatten(x, 1)


     if alpha is not None:

        reverse_feature =  ReverseLayerF.apply(features, alpha)
        domian_out = self.domain_classifier(reverse_feature)
        return domian_out
    else:
        class_out = self.class_classifier(features)
        return class_out
```

We are going to use the model pretrained on ImageNet so we need to modify the weights also in the new branch. We use the same weights of the original classifier



```
def dann(pretrained=False, progress=True, **kwargs):

    model = DANN(**kwargs)
    if pretrained:
        state_dict = load_state_dict_from_url(model_urls['alexnet'],
                                              progress=progress)

        model.load_state_dict(state_dict, strict=False )
        model.domain_classifier[1].weight.data = model.classifier[1].weight.data
        model.domain_classifier[4].weight.data = model.classifier[4].weight.data
        model.domain_classifier[1].bias.data = model.classifier[1].bias.data
        model.domain_classifier[4].bias.data = model.classifier[4].bias.data

    return model
```

For this Homework, the source domain is Photo, while the target domain is Art painting.To train the model we try to optimize lr and alpha, we use this set of value

lr_list = [0.0001, 0.0005, 0.001,0.005,0.01]

alpha_list = [0.001,0.005,0.01,0.03,0.05,0.08,0.1]

we need to perform a grid search to choose the best hyperparameter. The problem with this aproach is the we are using the target label to validate the model. This is like cheating because during the training time usually we do not have access to the label of test set. Then we need to use another aproach that is call cross domain validation. Instead of using the target set we will use a set coming from a different but similar domain to do the validation. In this homework we will use the cartoon dataset and the sketch dataset to choose the best hyperparameter and then we will test them on the target dataset.

4.a) VALIDATION WITHOUT DOMAIN ADAPTATION

We use the pytorch implementation of alexnet and we train it on photo dataset. We select the best lr between **lr_list = [0.0001, 0.0005, 0.001,0.005,0.01]**. To perform this validation we use 10 epoch for each training and after each training phase we test the model on sketch and cartoon dataset,average results to select the best model. Then we test the model on our test set and we obtain an accuracy of **0.47**

4.b) VALIDATION WITH DOMAIN ADAPTATION

Now we do domain adaptation using DANN. To validate the model we do cross domain validation, we train first a network using photo dataset for source and sketch dataset for target. Then we will train again the network using again photo for source and cartoon dataset for target. For each test step we save the accuracy and then we avarage them to select the best and learning rate.

This validation step require lot of time due colab limitation we use only 5 epoch for the train

we observe significant deviation between experiments, we do multiple runs and average the result

Learning Rate	Alpha	Photo to Cartoon Run 1	Photo to sketch Run 1	Photo to cartoon run 2	Photo to sketch run 2	Photo to Cartoon run 3	Photo to sketch run 3	Photo to Cartoon run 4	Photo to sketch run 4	Average
0,0001	0,001	0,248	0,278	0,252	0,209	0,187	0,333	0,190	0,309	0,251
0,0001	0,005	0,344	0,163	0,260	0,180	0,247	0,229	0,260	0,358	0,255
0,0001	0,01	0,26	0,179	0,291	0,258	0,334	0,181	0,267	0,203	0,247
0,0001	0,03	0,293	0,375	0,262	0,205	0,259	0,172	0,256	0,157	0,247
0,0001	0,05	0,272	0,272	0,232	0,271	0,207	0,229	0,224	0,213	0,240
0,0001	0,08	0,233	0,299	0,191	0,293	0,206	0,138	0,227	0,287	0,234
0,0001	0,1	0,293	0,158	0,257	0,164	0,241	0,212	0,214	0,239	0,222
0,0005	0,001	0,269	0,194	0,206	0,199	0,214	0,303	0,205	0,172	0,220
0,0005	0,005	0,201	0,273	0,243	0,173	0,227	0,207	0,199	0,172	0,212
0,0005	0,01	0,208	0,266	0,212	0,282	0,255	0,299	0,172	0,217	0,239
0,0005	0,03	0,177	0,249	0,225	0,266	0,260	0,221	0,197	0,176	0,221
0,0005	0,05	0,233	0,312	0,238	0,198	0,243	0,316	0,201	0,179	0,240
0,0005	0,08	0,27	0,203	0,206	0,178	0,245	0,250	0,205	0,287	0,231
0,0005	0,1	0,238	0,174	0,215	0,235	0,257	0,254	0,227	0,235	0,229
0,01	0,001	0,191	0,179	0,230	0,222	0,237	0,173	0,196	0,281	0,214
0,01	0,005	0,2	0,221	0,206	0,270	0,214	0,178	0,194	0,192	0,209
0,01	0,01	0,214	0,178	0,218	0,188	0,236	0,206	0,237	0,220	0,212
0,01	0,03	0,191	0,166	0,173	0,186	0,195	0,166	0,197	0,290	0,196
0,01	0,05	0,24	0,223	0,213	0,274	0,247	0,175	0,251	0,222	0,231
0,01	0,08	0,258	0,183	0,200	0,212	0,196	0,173	0,223	0,168	0,202
0,01	0,1	0,195	0,190	0,223	0,190	0,221	0,171	0,224	0,167	0,198
0,005	0,001	0,275	0,242	0,267	0,307	0,236	0,278	0,272	0,306	0,273
0,005	0,005	0,313	0,266	0,301	0,251	0,271	0,304	0,309	0,302	0,290
0,005	0,01	0,229	0,312	0,199	0,308	0,250	0,299	0,277	0,250	0,266
0,005	0,03	0,362	0,290	0,258	0,315	0,230	0,289	0,286	0,318	0,294
0,005	0,05	0,365	0,194	0,330	0,278	0,381	0,282	0,268	0,200	0,287
0,005	**0,08**	**0,376**	**0,303**	**0,337**	**0,238**	**0,363**	**0,339**	**0,453**	**0,307**	**0,340**
0,005	0,1	0,186	0,239	0,485	0,232	0,331	0,435	0,166	0,306	0,298
0,01	0,001	0,37	0,311	0,309	0,338	0,270	0,283	0,314	0,312	0,313
0,01	0,005	0,395	0,303	0,293	0,258	0,348	0,297	0,250	0,292	0,304
0,01	0,01	0,289	0,299	0,212	0,332	0,336	0,247	0,339	0,289	0,293
0,01	0,03	0,381	0,336	0,326	0,253	0,422	0,297	0,297	0,344	0,332
0,01	0,05	0,438	0,323	0,321	0,351	0,442	0,226	0,305	0,294	0,337
0,01	0,08	0,165	0,196	0,166	0,196	0,166	0,196	0,467	0,205	0,220
0,01	0,1	0,165	0,196	0,166	0,258	0,166	0,319	0,166	0,299	0,217

As we can see training our network 4 times and average the result we found that the best hyperparameter are **lr = 0.005** and **alpha = 0.08**

We train again the model with this hyperparameter using photo for source dataset and art for target dataset.

test accuracy run 1	test accuracy run 2	test accuracy run 3	test accuracy run 4	average
0,510	0,486	0,501	0,518	0,504

With this hyperparameter we get **0.504 ± 0.010** accuracy on the target dataset. This means that we increase the result of 3% with DANN. This is significateve for PACS which has a larger cross domain gap that result in a more challenging domain shift