Machine Learning and Deep Learning - [redacted] report

Hadi Nejabat s246601

1. Introduction

the field of Computer vision has made significant progress thanks to effectiveness of Convolutional Neural Networks (CNNs). As an example, in a classification task, you would often use one of the standard network architectures out there (AlexNet, ResNet, etc.) and train it using your dataset. This would likely lead to very good performance. Moreover, using Transfer Learning and training on a network that is pretrained on a large dataset and tune some of its upper layers using your own small annotated dataset has a huge impact on the results. Both of these approaches assume that your training data is representative of the underlying distribution. However, if the inputs at test time differ significantly from the training data, the model might not perform very well.

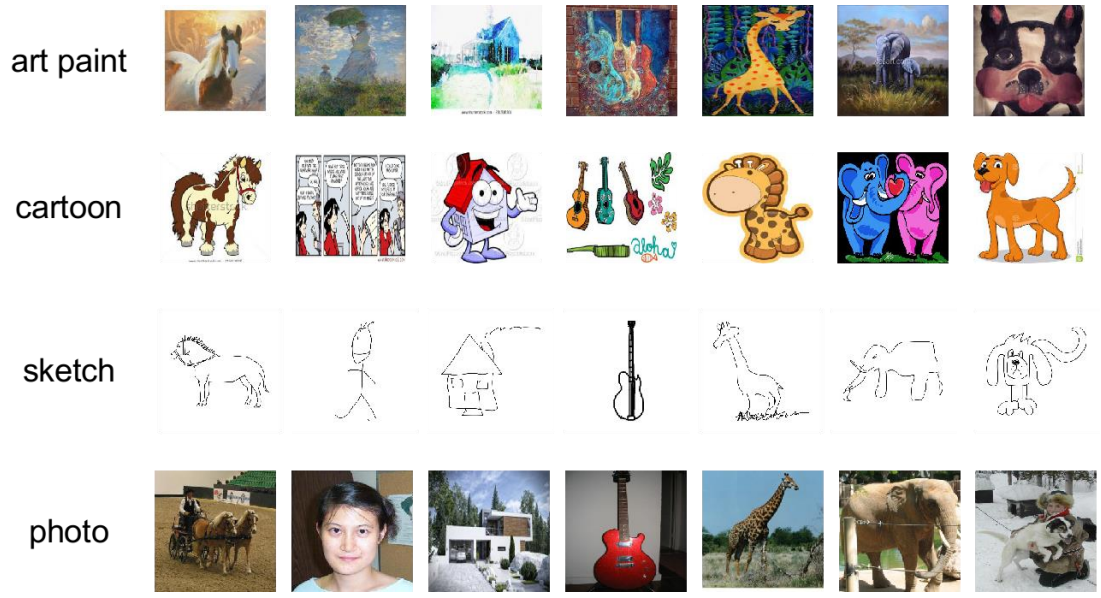
The reason the model did not perform very well in these scenarios is that the problem domain changed. In this particular case, the domain of the input data changed while the task domain (the labels) remained the same. On other occasions, you may want to use data from the same domain to accomplish a new task. Similarly, the input and task domains could differ at the same time. In these cases, domain adaptation comes to aid. Domain adaptation is a sub-discipline of machine learning which deals with scenarios in which a model trained on a source distribution is used in the context of a different target distribution. In general, domain adaptation uses labeled data in one or more source domains to solve new tasks in a target domain.

Our method is to use Adversarial Domain Adaptation. This technique tries to achieve domain adaptation by using adversarial training. if we use the domain confusion loss in addition to the classification loss used for the current task. The domain confusion loss is similar to the discriminator in GANs in that it tries to match the distributions of the source and target domains in order to “confuse” the high-level classification layers. Perhaps the most famous example of such a network is the Domain-Adversarial Neural Network (DANN). This network consists of two

losses, the classification loss and the domain confusion loss. It contains a gradient reversal layer to match the feature distributions. By minimizing the classification loss for the source samples and the domain confusion loss for all samples, this makes sure that the samples are mutually indistinguishable for the classifiers.

2. Martial and Methods

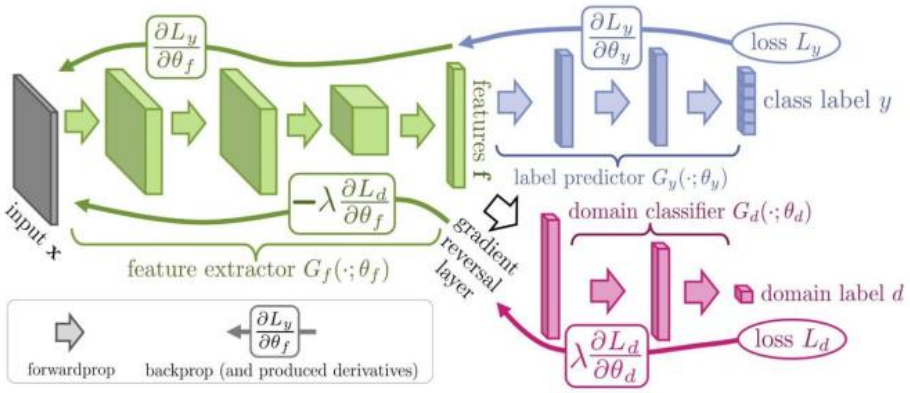
2.1. Dataset



For our task we will use the PACS dataset, which contains overall 9991 images, split unevenly between 7 classes and 4 domains: Photo, Art painting, Cartoon and Sketch. As can be clearly seen from the above Figure there are significant differences between domains. The original size of each image is 227x227 pixels (with 3 channels, i.e. RGB). The Image distributions are as follows:

	Dog	Elephant	Giraffe	Guitar	Horse	House	Person
Photo	189	202	182	186	199	280	**432**
Art/Painting	379	255	285	184	201	295	**449**
Cartoon	389	255	285	184	201	295	**449**
Sketch	772	740	753	608	816	80	**160**
Sum	**1729**	**1654**	**1566**	**1113**	**1540**	**943**	**1446**

2.2. Approach:



In this project we focus on the harder unsupervised case, although the proposed approach can be generalized to the semi-supervised case. We thus focus on learning features that combine discriminativeness and domain-invariance. This is achieved by jointly optimizing the underlying features as well as two discriminative classifiers operating on these features: (i) the label predictor that predicts class labels and is used both during training and at test time and (ii) the domain classifier that discriminates between the source and the target domains during training. While the parameters of the classifiers are optimized in order to minimize their error on the training set, the parameters of the underlying deep feature mapping are optimized in order to minimize the loss of the label classifier and to maximize the loss of the domain classifier.

It is shown in above figure that the training processes can be implemented into deep feedforward network that uses standard layers and loss functions, and can be trained using standard backpropagation algorithms based on stochastic gradient descent.

For my network training I used Cross Entropy loss as my criterion and SGD as my optimizer.

2.3. Model (Reconstructed AlexNet Model):

our task is to forge our model from AlexNet by first pre-loading the pre-trained weights on ImageNet and modifying the "init function" of the module and add the Domain Classifier. Considering the structure of DANN, I created a custom module where it can be seen in the following figure, that I separated the layers of original AlexNet and first created my feature extractor which are the first 5 convolutional layers of the model. The importance of this step is that I need to pre-load the weights by using "**`state_dict`**".

In PyTorch, the learnable parameters (i.e. weights and biases) of an `torch.nn.Module` model are contained in the model's parameters (accessed with `model.parameters()`). A state_dict is simply a Python dictionary object that maps each layer to its parameter tensor. Note that only layers with learnable parameters (convolutional layers, linear layers, etc.) and registered buffers (batchnorm's running_mean) have entries in the model's state_dict. Optimizer objects (torch.optim) also have a state_dict, which contains information about the optimizer's state, as well as the hyperparameters used.

Following that we use the same Fully connected layer of the original model in parallel for both our Label Predictor and Domain Classifier. Now for introducing our DANN task we must modify the forward function of the model and apply the principles of Adversarial Domain Adaptation. Where we define the "**`ReverseLayerF`**" to apply the gradient reversal layer.

2.4. Related work:

The model is trained on 256-sized batches. Images are preprocessed by the mean subtraction. A half of each batch is populated by the samples from the source domain (with known labels), the

rest is comprised of the target domain (with unknown labels). In order to suppress noisy signal from the domain classifier at the early stages of the training procedure instead of fixing the adaptation factor λ, we gradually change it from 0 to 1 using the following schedule; where γ was set to 10 in all experiments (the schedule was not optimized).

$$\lambda_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1,$$

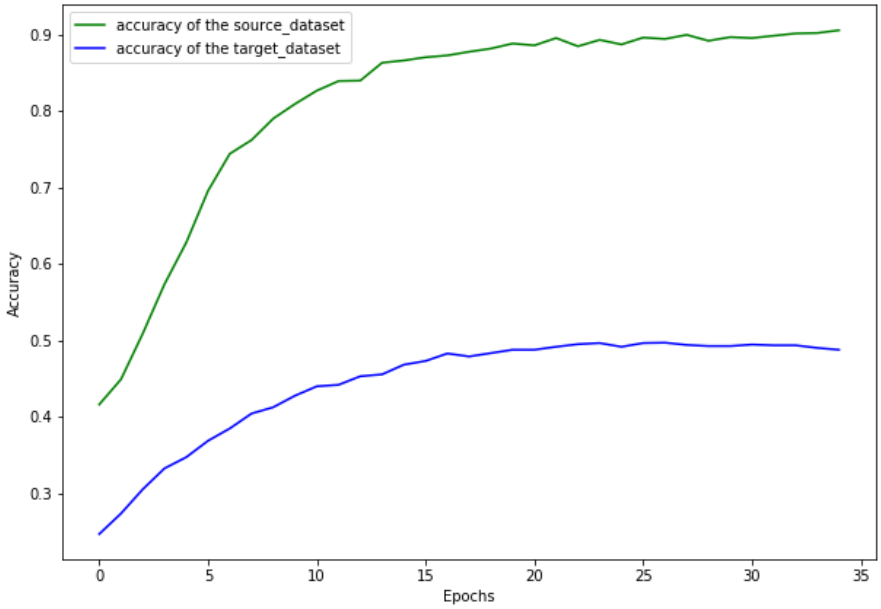
3. Implementation details:

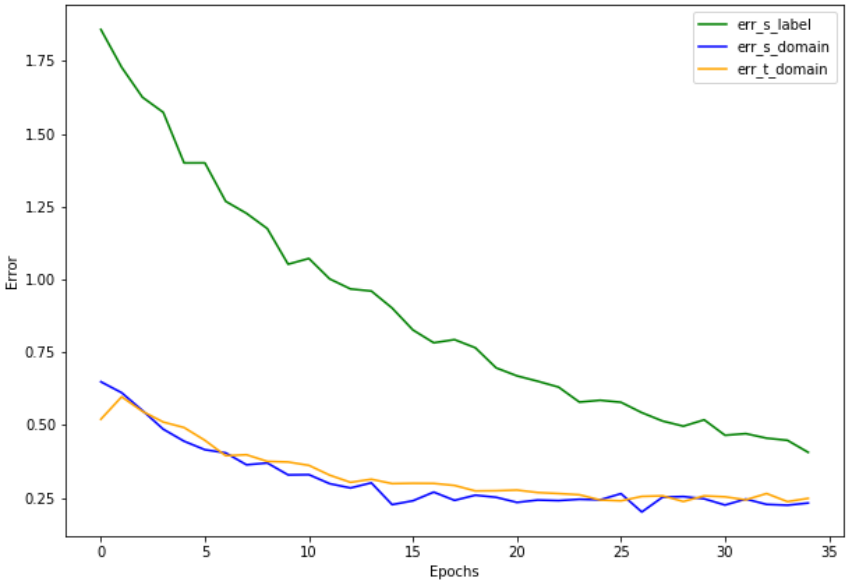
to begin, we train a Baseline with original AlexNet and study its performance to be applied in case of a domain adaptation problem. In this first part, runs were carried out without considering the cartoon and sketch domains as validation sets. Therefore, the network was trained on the photo domain and tested on art painting without domain adaptation. After recording the data, we starting the DA by feeding the DANN module our source and target data. The results regarding this part are shown in the following table. Which is as we expected, that we have a significant increment performing training with DANN adaptation.

4. Results

As stated in the assignment, the whole photo dataset has been used for training the model and the whole art painting dataset has served for model testing. We initialized with a set of reasonable starting hyperparameters. Moreover, since I skipped the Cross-Domain Validation, I ran the model multiple times with different parameters to find the optimum set of initializing parameters. the results regarding this part are shown in the following table and I have plotted the most significant results.

Hyper-parameters		**Test accuracy**	
LR	**Batch size**	**without DANN**	**with DANN**
5.00E-03	256	50.35%	54.52%
5.00E-03	128	50.90%	52.46%
1.00E-03	256	48.33%	51.10%
1.00E-03	128	49.03%	51.60%
1.00E-02	256	49.19%	50.20%
1.00E-02	128	50.18%	53.56%





As a Final report, the best score is obtained with LR=1e-3 and batch size=128 leading to a test accuracy (i.e. on Art Painting domain) equal to 53.56%.

5. Conclusion

Good unsupervised DA methods should provide ways to set hyperparameters (such as λ, the learning rate, the momentum rate, the network architecture for our method) in an unsupervised way, i.e. without referring to labeled data in the target domain. In our method, one can assess the performance of the whole system and the effect of changing hyper-parameters by observing the test error on the source domain and the domain classifier error. In general, we observed a good correspondence between the success of adaptation and these errors where it can be seen adaptation have better results when the source domain test error is low, while the domain classifier error is high.

What makes our approach significant is achieving alignment through standard backpropagation training. The approach is therefore rather scalable, and can be implemented using any deep learning package. Moreover, implementing a pre-trained AlexNet model speed up the model development process.

References

[1] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2017. “Deeper, broader and artier domain generalization”. In ICCV,

[2] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2018. “Learning to generalize: Meta-learning for domain generalization”. In AAAI,

[3] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. 2012. “ImageNet Classification with Deep Convolutional Neural Networks”. Neural Information Processing Systems. 25. 10.1145/3065386.

[4] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V.S. 2016. “Domain-Adversarial Training of Neural Networks”. ArXiv, abs/1505.07818.

[5] https://github.com/fungtion/DANN

[6] https://towarddatascience.com/DANN