Shanghai Jiao Tong University
School of Electronic and Electrical Engineering

Neural Network Theory and Applications
Assignment 3

Tianyue Cao
119034910071

Contents

Problems

Problem Introduction

Individual differences in EEG signals lead to the poor generalization ability of EEG-based affective models. Transfer learning, as we introduced in the class, can eliminate the subject differences and achieve appreciable improvement in recognition performance. In this assignment, you are asked to build and evaluate a cross-subject affective model using **Domain-Adversarial Neural Networks (DANN)** with the SEED dataset.

For more information about DANN model and transfer learning, please refer to:

- `https://bcmi.cloud:5001/sharing/S8iV7Ea8s`
- `https://bcmi.cloud:5001/sharing/zn9JH5ujg`
- `https://bcmi.cloud:5001/sharing/IREYDL6oa`

Dataset Introduction

The data sets provided in this assignment are from the SEED dataset, which are collected from 5 subjects, and the emotion labels are 1 for positive, 0 for neutral, and - 1 for negative.

The data is stored in python pickle format, to load the data into memory you can use the code (python3) below:

```
import pickle
with open('data.pkl', 'rb') as f:
data = pickle.load(f)
```

Once you load the data into memory, you can use data.keys() to get the subject information (i.e. sub_0, sub_1, ..., sub_5). And you can use data[‘sub_0’].keys() to find out data arrays and labels.

- The data link: `https://bcmi.cloud:5001/sharing/PZEhXmxeD`
- The code link: `https://bcmi.cloud:5001/sharing/Nq3DfMFHP`

Problems

Emotion recognition with DANN and compare with baseline method

You are required to apply leave-one-subject-out cross validation to **classify different emotions** with DANN model and **compare the results** of DANN with a baseline model (you can choose the baseline model on your own).

Under Leave-one-subject-out cross validation configuration, for each subject, an affective model should be trained with one subject as target domain, and other subjects as source domain. In the end, there should be five DANN models for each of the subject, and you should report both the individual recognition accuracy and the mean recognition accuracy.

Here are some suggestions of parameter settings. The feature extractor has 2 layers, both with node number of 128. The label predictor and domain discriminator have 3 layers with node numbers of 64, 64, and C, respectively. C indicates the number of emotion classes to be classified.

Emotion recognition using a different transfer learning model (Optional)

To have a deeper understanding of transfer learning, you can complete this **optional task** in which you are asked to solve the previous problem with a different transfer learning method. Similar to previous problem, you are also required to apply **leave-one-subject-out cross validation** to evaluate. For each subject, an affective model should be trained with one subject as the test set, and other subjects as training sets.

CHAPTER 1

Basic Information

1.1 Environments

In this assignment, the hardware and software environments I used are shown as following:

- **Calculation Platform:** GeForce GTX 1080 Ti
- **CUDA Version:** 10.2
- **Operation System:** Ubuntu 16.04.5 LTS
- **Linux Core Version:** 4.4.0-139-generic
- **Requirements:** Python 3.6, PyTorch 1.3.0, scikit-learn 0.20.1, Numpy 1.15.4, matplotlib, pickle

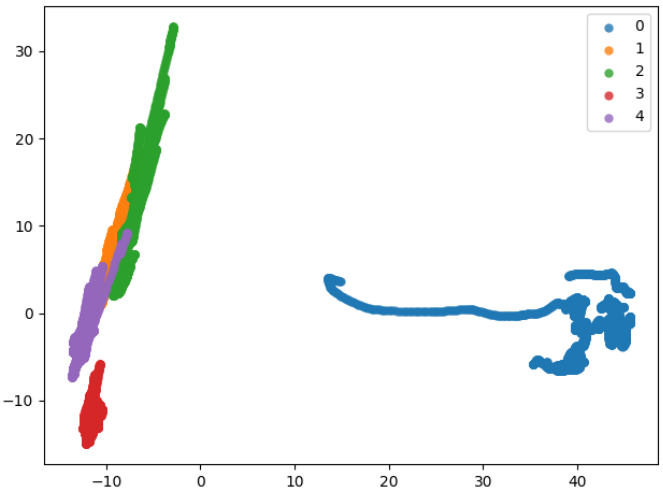

Figure 1.1. Distribution of samples in each subset.

1.2 Datasets

The dataset is the SJTU Emotion EEG Dataset (SEED), which is a three-class classification problem. The dataset is collected from 5 subsets, and the emotion labels are 1 for positive, 0 for neutral, and -1 for negative. Each subset contains 3397 samples, and each sample is represented by a 310 dimension vector.

Using PCA dimension reduction method to visulize the data, each subset is shown in Figure 1.1. From the figure, samples in subsets 1, 2, 3, 4 are distributed relatively consistently, but subset 0 samples have a pretty different distribution.

CHAPTER 2

Solution for Problem 1

In this problem, I use a domain adaptation network DANN to do target domain label prediction, and compare the performance with the baseline model.

2.1 Domain Adaptation

First, we fomulate the problem. A classification task with input space X and label space Y has different distributions over $X \times Y$. The two distributions are called *source domain* D_s and *target domain* D_t. An unsupervised domain adaptation learning algorithm is provided with a *labeled source sample* S drawn *i.i.d* from D_s, and an *unlabeled target sample* T drawn *i.i.d* from D_t^X, where D_t^X is the marginal distribution of D_t over X.

$$S = \{(x_i, y_i)\}_{i=1}^{n} \sim (D_s)^n; \quad T = \{(x_i, y_i)\}_{i=n+1}^{N} \sim (D_t^X)^{n'},$$

where $N = n + n'$ is the total number of samples. The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ with a low *target risk*:

$$R_{D_t}(\eta) = \Pr_{(x,y)\sim D_t}(\eta(x) \neq y).$$

without information of the labels of D_T.

2.2 Domain-Adversarial Neural Networks (DANN)

DANN [1] is a neural network classifier aims to learn a model that can generalize well from one domain to another, while the internal representation of the neural network contains no discriminative information about the origin of the input (source or target) and preserving a low risk on the source examples.

The architecture of DANN is shown in Figure 2.1. The network contains three parts: a feature extractor (pink), a label predictor (green), and a domain discriminator (blue). The network functions and parameters are denoted respectively as $G_f(\cdot), \theta_f, G_y(\cdot), \theta_y, G_d(\cdot), \theta_d$.

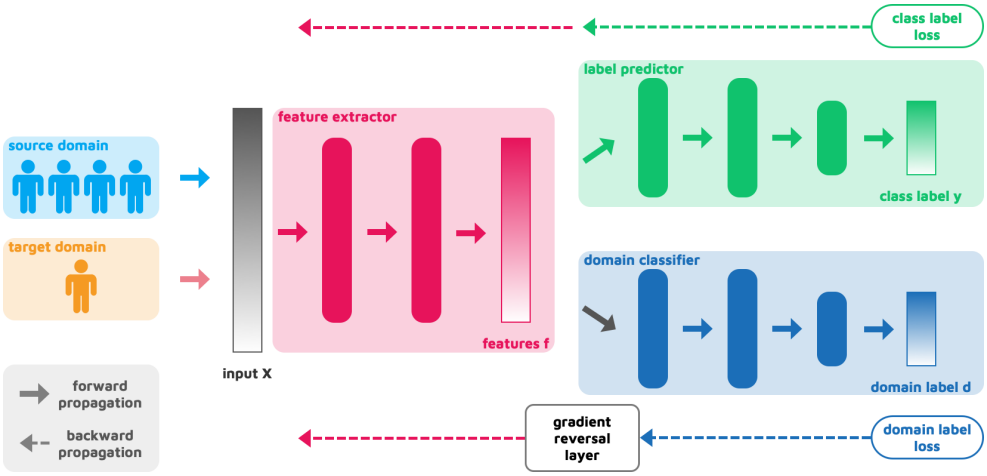

Figure 2.1. The architecture of DANN model.

The forward propogation phase is similar to the ordinary classification network. The feature extractor projects the original input feature to a new feature space, and then forward it into the label classifier $G_y(\cdot)$ and the domain classifier $G_d(\cdot)$ simultaneously. Then two classifiers output the prediction of label and domain.

Most of the backpropagation phase is also the same as the ordinary neural network, but a *gradient reversal layer* is proposed. The loss function of the whole network is

$$\frac{1}{n}\Sigma_{i=1}^{n} L_y(G_y(G_f(x_i)), y_i) + \alpha L_d(G_d(G_f(x_i)), d_i),$$

where $L_y(\cdot)$ is the loss for the label classifier and $L_d(\cdot)$ is the loss for the domain classifier. α is a tradeoff hyperparameter and n is the number of data samples.

For label predictor G_y and domain classifier G_d, the related loss L_y and L_d are propagaed the same as the ordinary networks. Also, the propagation process doesn't change while passing from label predictor to feature extractor. But the gradient is reversed when it is passed from the domain classifier to the feature extractor. It suggests that the feature extractor maximizes the domain classifier loss and extracts the most domian-unrelated features for label classification.

Finally, in test phase, testing data is forwarded to the feature extractor and the label predictor to get the label prediction.

2.3 Model and Training Details

2.3.1 Cross Validation Strategy

In this problem, leave-one-object-out cross validation is applied to classify different emotions with DANN model. For each subject, an affective model is trained with one subject as target domain, and other subjects as source domian.

2.3.2 Model Structure and Details

The DANN model structure is shown in Figure 2.1, source domain contains 4 subjects, and target domain contains 1 subject. The detailed model configuration is shown as following:

- **Feature Extractor (2 layers):**

  Input: 310 dimension input features

  Layer 1: fc layer with 128 hidden units

  Layer 2: fc layer with 128 hidden units

  Output: 128 dimension features

- **Label Predictor (3 layers):**

  Input: 128 dimension input features

  Layer 1: fc layer with 64 hidden units

  Layer 2: fc layer with 64 hidden units

  Layer 3: fc layer with 3 hidden units followed with logsoftmax layer

  Output: label prediction output

- **Domain Classifier (3 layers):**

  Input: 128 dimension input features

  Layer 1: fc layer with 64 hidden units

  Layer 2: fc layer with 64 hidden units

  Layer 3: fc layer with 2 hidden units followed with logsoftmax layer

  Output: domain classification output

For the activation function of each fc layer, I tried *sigmoid function* first, but found the network very hard to converge. After experiments, I use *tanh function* as the activation function for the fc layers. And the baseline model is consist of only a feature extractor and a label predictor.

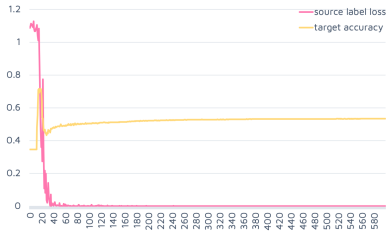

fig a.1. Results for baseline on target subject 0.

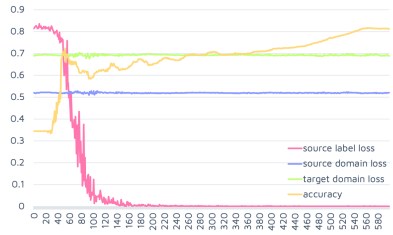

fig b.1. Results for DANN on target subject 0.

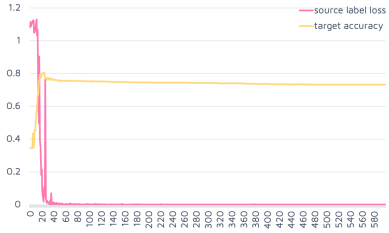

fig a.2. Results for baseline on target subject 1.

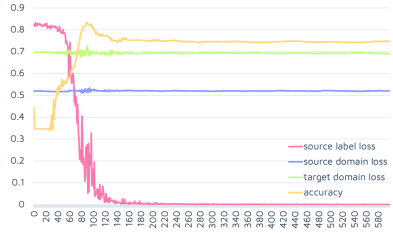

fig b.2. Results for DANN on target subject 1.

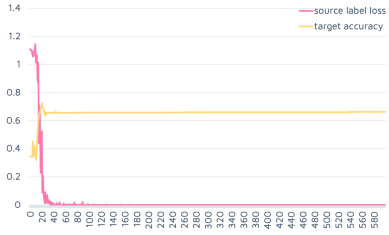

fig a.3. Results for baseline on target subject 2.

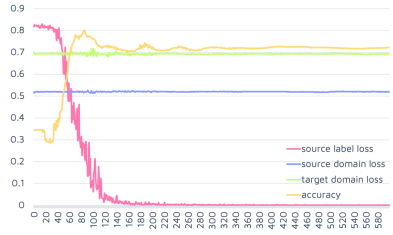

fig b.3. Results for DANN on target subject 2.

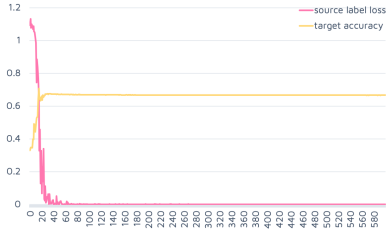

fig a.4. Results for baseline on target subject 3.

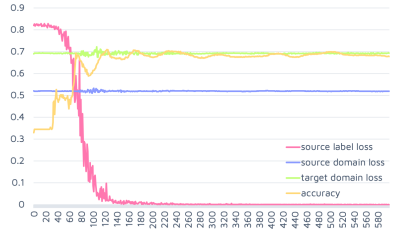

fig a.4. Results for DANN on target subject 3.

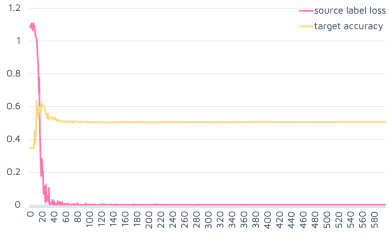

fig a.5. Results for baseline on target subject 4.

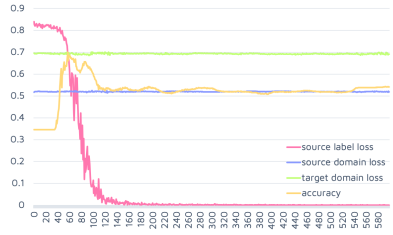

fig b.5. Results for DANN on target subject 4.

Figure 2.2. Baseline model (fig a.1-fig a.5) and DANN model (fig b.1-figb.5) training losses and target domain prediction accuracies.

target subset	0	1	2	3	4	avg
baseline (last 10 epochs avg.)	0.5330	0.7312	0.6618	0.6664	0.5069	**0.6481**
DANN (last 10 epochs avg.)	0.8137	0.7474	0.7209	0.6798	0.5416	**0.7007**
baseline (max)	0.7168	0.8041	0.7253	0.7076	0.6347	**0.7177**
DANN (max)	0.8171	0.8353	0.8003	0.7106	0.6991	**0.7725**

Table 2.1. Baseline model and DANN model target domain label prediction accuracies.

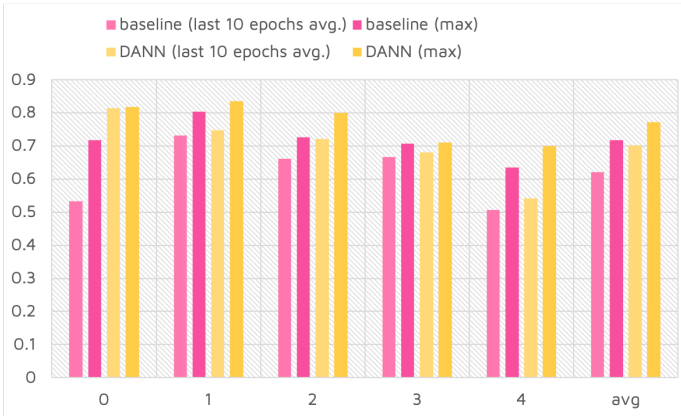

Figure 2.3. Baseline model and DANN model target domain label prediction accuracies.

2.3.3 Training Strategy

For DANN model, in each iteration, data from *source domain* and *target domain* are sampled as proportion 4:1. Source domain samples are used to train the label predictor and both source domain and target domain samples are used to train the domain classifier with a gradient reversal layer between the feature extractor and the domain classifier. For baseline model, samples from source domain are used to train the model and use the model to predict labels for target domain samples.

2.4 Results and Analysis

The training losses and target prediction accuracies are shown in Figure 2.2 and the accuraries are shown in Table 2.1 and Figure 2.3. According to the results in Table 2.1, DANN model can achieve better performance on target domain either on each subset or on average. But according to the training loss curves in Figure 2.2, DANN model is harder to converge, which follows the adversarial idea of DANN model.

According to the data distribution shown in Figure 1.1, subject 0 have pretty different distribution from the other subsets. Corresponding to it, the target accuracy curve

in fig a.1 shows overfitting on the source dataset, and cannot get good prediction accuracy on target domain subset 0. Also, the accuracy on target domain subset 0 on DANN model increases significantly, which suggests that domain adaptation plays an important role on adapting the model from source domain to target domain. The domain losses for both source domain and target domian are tend to not change, that is the consequence of gradient reversal layer. It confuses the domain classifier not to classify the sample into source or target domain.

CHAPTER 3

Solution for Problem 2

In problem 1, DANN [2] model is applied to do domain adaptation from source domain to target domain. However, in problem 1, source domain data actually contains 4 subsets, which have different distributions. The source domain in problem 1 is consists of multiple domains. In this problem, I use MDAN [2] to train a multiple domain adaptation network.

3.1 Multisource Domain Adaptation with Adversarial Neural Networks (MDAN)

MDAN network architecture is shown in Figure 3.1. The main architecture is similar to DANN, but it has k domain classifiers rather than only 1.

In forward phase, the i-th source domain data and target data are fed into the feature extractor. Then the source domain features are fed into the label predictior, the i-th source domain features and the target features are fed into the i-th domain classifier.

In backward phase, label loss are back propagated from label predictor to feature extractor. Each domain classifier can produce a domain loss, two back propagation strategies for the gradient reversal layer are proposed.

- **Hard version:** the source that achieves the minimum domain classification error is back propagated with gradient reversal;
- **Smooth version:** all the domain classification risks over k source domains are combined and back propagated with gradient reversal.

3.2 Model and Training Details

3.2.1 Cross Validation Strategy

In this problem, leave-one-object-out cross validation is applied. For each target domain, 4 source domains are used to train the domain classifiers respectively.

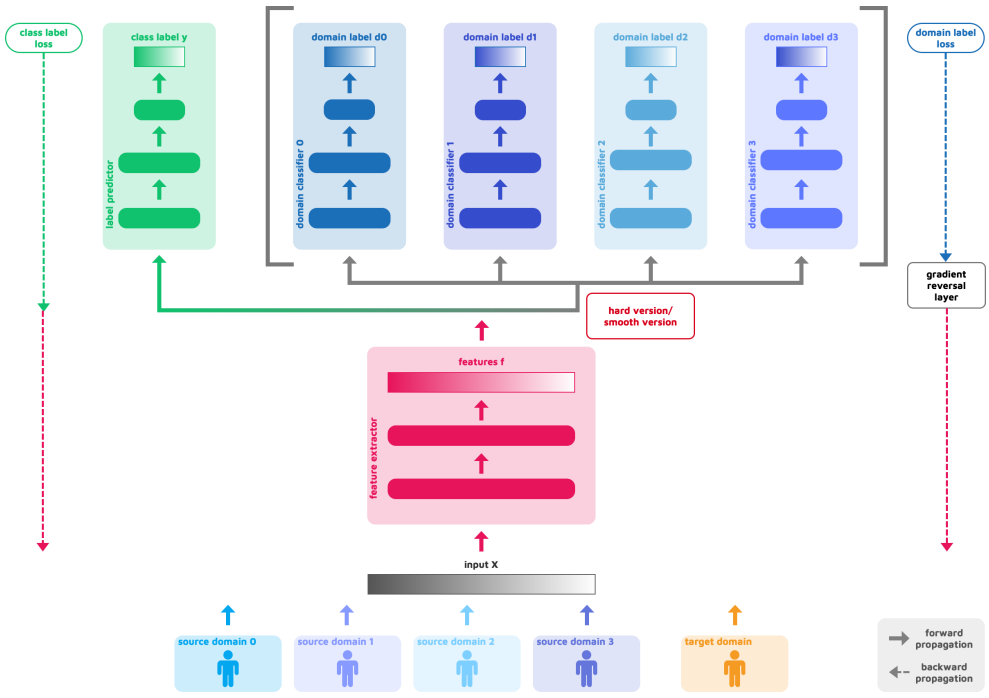

Figure 3.1. The architecture of MDAN model.

3.2.2 Model Structure and Details

The MDAN model structure is shown in Figure 3.1, 4 source domains and 1 target domain each contains 1 subject. The detailed model configuration for each part (*feature extractor, label predictor, 4 domain classifiers*) is the same as the DANN model in problem 1.

3.2.3 Training Strategy

For MDAN model, in each iteration, data from *4 source domains* and *target domain* are fed forward into the feature extractor. The 4 source domain samples are used to train the label predictor, and the i-th source domain samples and target domain samples are used to train the i-th domain classifier. Hard version (maxmin) and smooth version (all risks) back propagation strategies are used for the gradient reversal layer. For the hard version, the maximum of the 4 source domain sample losses and the reversal of the minimum of the 4 domain classifier losses are used to do back propagation. It is implementetd by defining the loss function as:

target subset	0	1	2	3	4	avg
baseline (last 10 epochs avg.)	0.5330	0.7312	0.6618	0.6664	0.5069	**0.6481**
DANN (last 10 epochs avg.)	0.8137	0.7474	0.7209	0.6798	0.5416	**0.7007**
hard version MDAN (last 10 epochs avg.)	0.6586	0.6150	0.5709	0.6900	0.5544	**0.6178**
smooth version MDAN (last 10 epochs avg.)	0.8658	0.6894	0.6913	0.8032	0.5540	**0.7207**
baseline (max)	0.7168	0.8041	0.7253	0.7076	0.6347	**0.7177**
DANN (max)	0.8171	0.8353	0.8003	0.7106	0.6991	**0.7725**
hard version MDAN (max)	0.6932	0.7562	0.7891	0.8068	0.6932	**0.7477**
smooth version MDAN (max)	0.8671	0.6988	0.7885	0.8065	0.6279	**0.7578**

Table 3.1. Baseline, DANN model, hard version and smooth version MDAN model target domain label prediction accuracies.

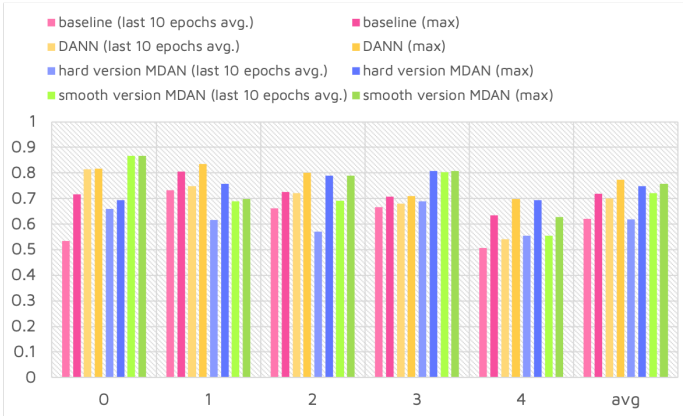

Figure 3.2. Target domain label prediction accuracies on baseline, DANN, hard version, and smooth version MDAN model.

$$L = \max_i L_y^{(i)} + \min_i L_{d_i} \tag{3.1}$$

For the smooth version, all the losses are used to do back propagation. It is implemented by defining the loss function as:

$$L = log\Sigma_i(exp(L_y^{(i)}) + exp(L_{d_i})) \tag{3.2}$$

3.3 Results and Analysis

After tuning hyperparameters, learing rate for target subset 1-4 hard version MDAN model is set to 1e-3, and the other models 5e-3. Lable loss, domain loss and target accuracies of different models are shown in Figure 3.3, and the accuracies on all the models are shown in Table 3.1 and Figure 3.2. *Note* that due to the constraints on

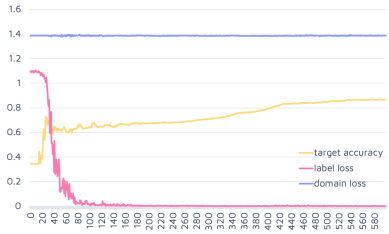

fig a.1. Results for smooth version MDAN on target subject 0.

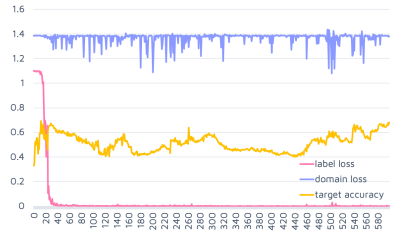

fig b.1. Results for hard version MDAN on target subject 0.

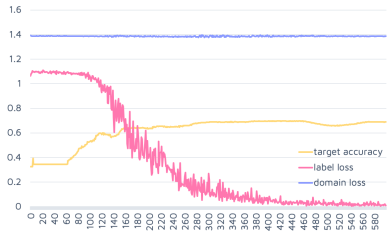

fig a.2. Results for smooth version MDAN on target subject 1.

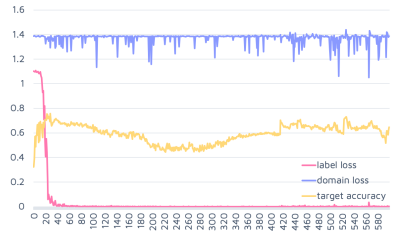

fig b.2. Results for hard version MDAN on target subject 1.

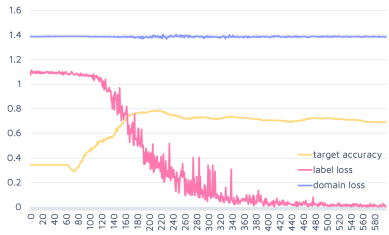

fig a.3. Results for smooth version MDAN on target subject 2.

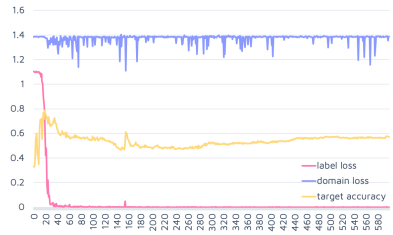

fig b.3. Results for hard version MDAN on target subject 2.

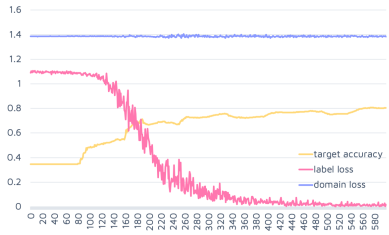

fig a.4. Results for smooth version MDAN on target subject 3.

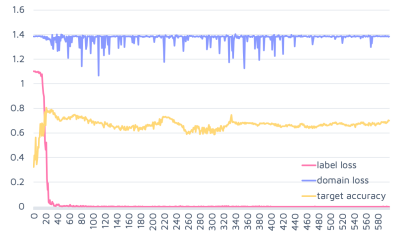

fig b.4. Results for hard version MDAN on target subject 3.

fig a.5. Results for smooth version MDAN on target subject 4.

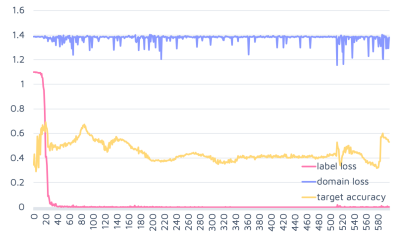

fig b.5. Results for hard version MDAN on target subject 4.

Figure 3.3. Hard version (fig a.1-fig a.5) and smooth version (fig b.1-fig b.5) MDAN model training loss and target domain prediction accuracies.

computing resources and time conditions, both of the MDAN models may not be tuned to the best.

According to Figure 3.3, comparing the label loss curves in fig a.1 and fig b.1, hard version is harder to converge sinci it uses a maxmin strategy to do back propagation. Also, the smooth version tends to overfit compared with the hard version, by observing the test accuracy curves.

According to Table 3.1 and Figure 3.2, the smooth version MDAN model can achieve the best performance among the 4 models. And the model can get pretty high accuracies on target subset 0 and target subset 3, which distribute quite differently from the others. But the hard version MDAN model performs not well, one of the reasons is that it is harder to converge and tune, and I do not have enough computation and time to tune it to the best performance. Also, it is harder to get stable performaces on all the 5 subsets using maxmin strategy.

Conclusion

In this assignment, I solve a 3-class classifiaction problem with 5 domains using 4 models: **a typical feature-extractor-label-classification network, a DANN model, a hard version MDAN model and a smooth version MDAN model**. A leave-one-object-out strategy is used for cross validation. In problem 1, a target domain is left out and the other 4 are treated as the source domain. In problem 2, since MDAN is a multiple source data adaptation network, a target domain is left out and the other 4 are treated as 4 source domains.

According to the results, DANN model can effectively do domain adaptation from source domain to target domain, which can improve the classification accuracy significantly. And MDAN model (smooth version) can perform better than DANN by treating the source domain objects with different distributions differently. Also, the hard version MDAN is hard to train because of the maxmin back propagation strategy.

In conclusion, in this assignment, I leant the effect of domain adaptation neural network and realized the importantness of domain adaptation. The idea of adversarial network implemented by a simple *Gradient Reversal Layer* is a great insparation. Also, by treating the source domain as multiple sources, MDAN model can do domain adaptation better.

Bibliography

[1] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, "Domain-adversarial training of neural networks," *CoRR*, vol. abs/1505.07818, 2015.

[2] H. Zhao, S. Zhang, G. Wu, J. P. Costeira, J. M. F. Moura, and G. J. Gordon, "Multiple source domain adaptation with adversarial learning," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, OpenReview.net, 2018.