CS3507 Course Project
Transfer Learning on EEG Sentiment Classification

Rong Shan
CS Department of SJTU
520021911331
shanrong@sjtu.edu.cn

Hang Zheng
CS Department of SJTU
520021911347
azure123@sjtu.edu.cn

Abstract

Sentiment analysis using electroencephalogram (EEG) signals has gained attention in recent years due to its potential applications in various fields such as healthcare and marketing. Transfer learning has been proposed as an effective approach to improve the performance of EEG sentiment classification models by leveraging knowledge learned from related domains. In this study, we compare the performance of four transfer learning algorithms, including Domain Adversarial Neural Networks (DANN), Adversarial Discriminative Domain Adaptation (ADDA), Domain Generalization with Domain-Adversarial Training (DG-DANN), and Deep Residual Learning with Domain-Adversarial Training (DResNet), on a four-class sentiment dataset called SJTU seed-IV. Our experimental results show that all the algorithms achieve accuracy close to 0.6, except for ADDA which has a significantly lower performance. Furthermore, we analyze the impact of data augmentation and visualize the learned feature representations using t-distributed stochastic neighbor embedding (tSNE). Experiments are also conducted to explore how much data is enough for the models to learn a good representation. Our findings suggest that transfer learning can effectively improve the performance of EEG sentiment classification models and that different algorithms may have varying levels of effectiveness depending on the specific dataset and task.

1. Introduction

Electroencephalogram (EEG) sentiment analysis has emerged as a promising research area with potential applications in various domains such as healthcare, marketing, and human-computer interaction. EEG signals have been shown to provide valuable information about human emotional states, making them a useful tool for sentiment analysis tasks. However, EEG sentiment analysis faces significant challenges due to the high dimensionality and variability of EEG signals, as well as the lack of large-scale labeled datasets.

Transfer learning has been proposed as an effective approach to tackle these challenges and improve the performance of EEG sentiment classification models. By leveraging knowledge learned from related domains, transfer learning can help models generalize better to new domains with limited labeled data. Two common transfer learning techniques are domain adaptation and domain generalization. Domain adaptation methods aim to adapt the model from the source domain to the target domain by reducing the domain shift, while domain generalization methods aim to learn a domain-invariant representation that can generalize to unseen domains.

In this study, we explore the effectiveness of four transfer learning algorithms for EEG sentiment classification: Domain Adversarial Neural Networks (DANN), Adversarial Discriminative Domain Adaptation (ADDA), Domain Generalization with Domain-Adversarial Training (DG-DANN), and Deep Residual Learning with Domain-Adversarial Training (DResNet). DANN and ADDA are domain adaptation methods, while DG-DANN and DResNet are domain generalization methods. We evaluate the performance of these algorithms on a four-class sentiment dataset called SJTU seed-IV, which is a challenging benchmark for EEG sentiment analysis.

In addition to comparing the performance of the four transfer learning algorithms, we also conduct experiments to investigate the impact of data augmentation and the amount of labeled data on the performance of the models. Specifically, we explore how much labeled data is needed for the models to learn a good representation. Furthermore, we visualize the learned feature representations using t-distributed stochastic neighbor embedding (tSNE) to gain insights into the effectiveness of the transfer learning algorithms.

The contributions of this study are four-fold. First, we compare the performance of four transfer learning algorithms for EEG sentiment classification on a challenging

dataset. Second, we investigate the impact of data augmentation and the amount of labeled data on the performance of the models. Third, we provide insights into the effectiveness of transfer learning algorithms through feature visualization. Finally, we provide a comprehensive analysis of the results and discuss the implications of our findings for future research in EEG sentiment analysis.

2. Preliminaries

In this section we will elaborate more on the four transfer learning methods we leveraged, i.e. domain adaptation methods DANN and ADDA, domain generalization methods DG-DANN and DResNet.

2.1. Notation and Formulation

2.1.1 Domain Adaptation

Let $D_S = \{(x_i, y_i)\}_{i=1}^{n_S}$ and $D_T = (x_j)j = 1^{n_T}$ represent the labeled source and unlabeled target domains, respectively. Domain adaptation aims to learn a model f that minimizes the discrepancy between the source and target domains while maintaining good classification performance on the target domain. Specifically, domain adaptation seeks to find a feature mapping $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier $h : \mathcal{Z} \rightarrow \mathcal{Y}$ that minimize the following loss function:

$$L_{DA}(f) = L_{cls}(h(\mathcal{F}(D_S))) + \lambda L_{align}(\mathcal{F}(D_S), \mathcal{F}(D_T))$$

where L_{cls} is the classification loss, L_{align} is the domain alignment loss, and λ is a hyperparameter that controls the trade-off between classification and domain alignment.

2.1.2 Domain Generalization

Let $D = D_1, D_2, ..., D_K$ represents a set of K source domains, each with labeled examples $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{n_k}$, and a target domain $D_T = \{(x_j)\}_{j=1}^{n_T}$ with no labeled examples. Domain generalization aims to learn a model f that can generalize to new, unseen domains by extracting domain-invariant features. Specifically, domain generalization seeks to find a feature mapping $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier $h : \mathcal{Z} \rightarrow \mathcal{Y}$ that minimize the following loss function:

$$L_{DG}(f) = \frac{1}{K} \sum_{k=1}^{K} L_{cls}(h(\mathcal{F}(D_k))) + \lambda L_{dis}(\mathcal{F}(D_k)_{k=1}^{K})$$

where L_{cls} is the classification loss, L_{dis} is the domain separation loss, and λ is a hyperparameter that controls the trade-off between classification and domain separation.

2.2. Model Architecture and Training Procedure

2.2.1 DANN

DANN[1] is a domain adaptation approach with deep architectures that can be trained with labeled source domain data and unlabeled target domain data. Its adaptation behavior is achieved by augmenting a normal feed-forward model with an adversarial manner.

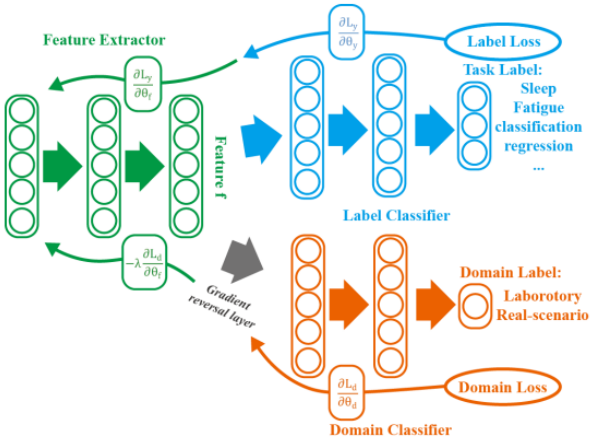

Figure 1. Architecture of DANN

As Fig.1 shows, the first layers works as a feature extractor, it maps the low level features to a non-linear transformed high level feature *f*. The upper layers take as input the feature *f* and outputs the label prediction for the required task. These two parts works like a normal multilayer network.

The lower layers take as input the feature f and outputs the domain label. It tries to distinguish whether the data come from D_S or D_T.

Each classifier back propagates the gradient to update the parameters. The gradient of label classifier passes to the feature extractor normally. The gradient of domain classifier passes to the feature extractor with a reversal parameter $-\lambda$.

The reversal parameter $-\lambda$ makes the domain classifier and feature extractor updates their parameters in an adversarial manner. The feature extractor wants to cheat the domain classifier and tries to hide any domain dependent information by removing it from feature *f*.

When the model converges, the feature f would be containing no domain related information. Thus it works as function of domain adaptation.

2.2.2 ADDA

Like DANN, ADDA[3] is based on deep adversarial network. It aligns the distributions of source domain data and target domain data by training a feature extractor and a domain classifier in an adversarial manner.

First, we train a simple multilayer perceptron on the source domain data with source domain data as input. The network is divided into 3 parts: input layer, feature extractor and label predictor. We can follow the general training

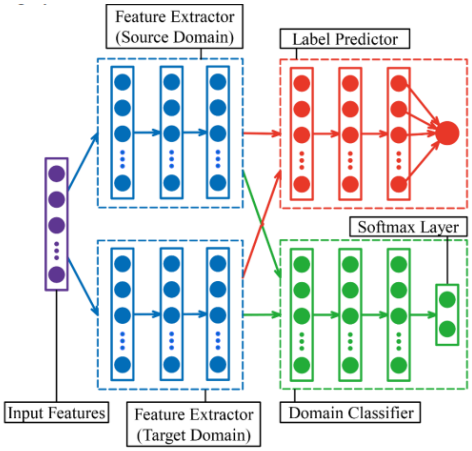

Figure 2. Architecture of ADDA

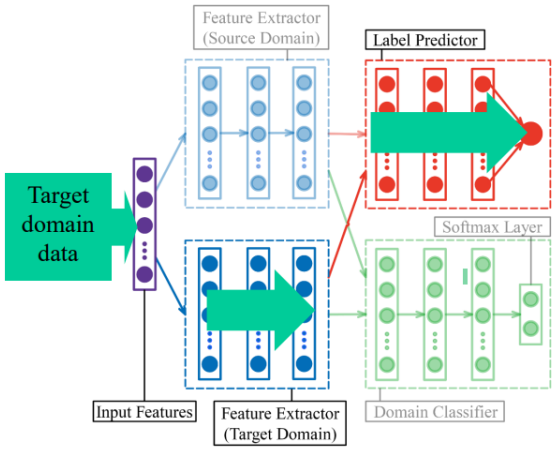

Figure 3. Prediction procedure of ADDA

procedure for this step.

After the network converge, another feature extractor is constructed. The new feature extractor only accepts input from the target domain. Its parameters are initialized by copying the ones from the original feature extractor.

In this step, a domain classifier sub-network is trained when the label predictor and the source domain feature extractor are not updated. The domain classifier is trained to discriminate which domain (source or target) the input is from.

However, the target domain feature extractor is trained to deceive the domain classifier. The domain classifier and the target domain feature extractor compete with each other (i.e., adversarial training).

After the adversarial training converges, we can predict the labels of the target domain data by using the pass as shown in Fig.3.

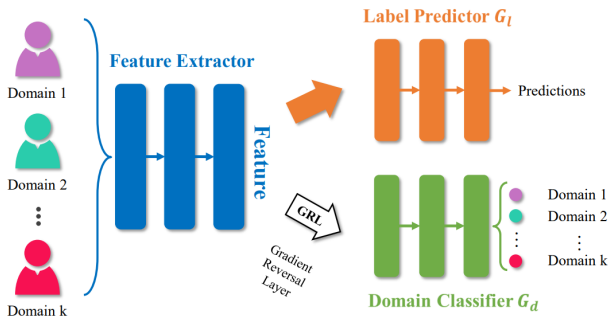

Figure 4. Architecture of DG-DANN

2.2.3 DG-DANN

Domain Generalization with Domain-Adversarial Training (DGDANN) is a domain generalization method that aims to learn a domain-invariant feature representation that can generalize to unseen domains. DGDANN is an extension of Domain Adversarial Neural Networks (DANN) and combines domain adaptation and domain generalization in a unified framework.

DGDANN consists of three components: a feature extractor network G, a task classifier network T, and a domain classifier network D. The feature extractor G is a deep neural network that maps the input samples to a high-level feature representation. The task classifier T is a classifier that takes the features extracted by G and predicts the label of the sample. The domain classifier D is a binary classifier that takes the features extracted by G and predicts the domain label of the sample.

The overall objective of DGDANN is to learn a feature representation that is domain-invariant while still preserving the discriminative information for the classification task. This is achieved by jointly training G, T, and D with an adversarial loss function. The adversarial loss function consists of three terms: a classification loss term, a domain adversarial loss term, and a domain generalization loss term.

The overall objective function of DGDANN can be expressed as $L(G, T, D, X_S, Y_S, X_T, X_1, X_2, ..., X_K) = L_{cls}(G, T, X_S, Y_S) + \lambda_1 L_{adv}(G, D, X_S, X_T) + \lambda_2 L_{gen}(G, X_1, X_2, ..., X_K)$ where λ_1 and λ_2 are hyperparameters that control the trade-off between the classification loss, domain adversarial loss, and domain generalization loss.

The feature extractor G is trained to minimize the classification loss and the domain generalization loss while maximizing the domain adversarial loss. The task classifier T is trained to minimize the classification loss. The domain classifier D is trained to maximize the domain adversarial loss.

By jointly optimizing the three objectives, DGDANN learns a feature representation that is domain-invariant and

can generalize to unseen domains while preserving the discriminative information for the classification task. The domain generalization loss term encourages the feature extractor to learn domain-invariant features across different source domains, which improves the generalization ability of the model to unseen domains.

2.2.4 DResNet

Deep Residual Learning with Domain-Adversarial Training (DResNet) is a domain generalization method that combines deep residual learning and domain adversarial training. DResNet aims to learn a domain-invariant feature representation that can generalize to unseen domains while preserving the discriminative information for the classification task.

DResNet is based on the residual neural network (ResNet) architecture, which utilizes residual blocks to learn deep representations. In DResNet, each residual block is augmented with a domain adversarial training module. The domain adversarial training module consists of a domain classifier network D and a gradient reversal layer (GRL). The GRL is a special layer that reverses the sign of the gradients during backpropagation, which helps to align the feature representations across different domains.

The overall objective of DResNet is to learn a feature representation that is domain-invariant while still preserving the discriminative information for the classification task. This is achieved by jointly training the ResNet and domain adversarial training module with an adversarial loss function. The adversarial loss function consists of three terms: a classification loss term, a domain adversarial loss term, and a domain generalization loss term.

During training, the ResNet and domain adversarial training module are jointly optimized to learn a feature representation that is domain-invariant and can generalize to unseen domains while preserving the discriminative information for the classification task. The ResNet is trained to minimize the classification loss and the domain generalization loss while maximizing the domain adversarial loss. The domain classifier D is trained to maximize the domain adversarial loss.

By jointly optimizing the three objectives, DResNet learns a feature representation that is domain-invariant and can generalize to unseen domains while preserving the discriminative information for the classification task. The ResNet architecture with domain adversarial training module helps to extract domain-invariant features while minimizing the negative effect of domain shift.

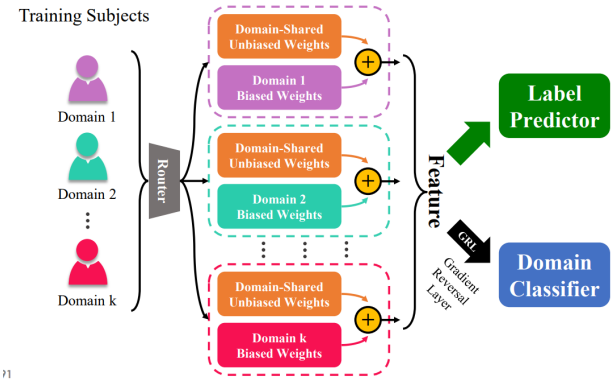

Figure 5. Traing procedure of DResNet

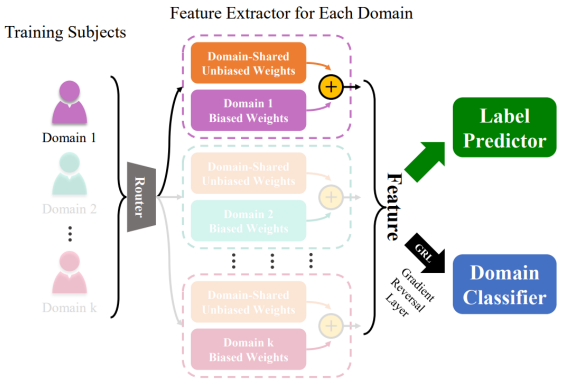

Figure 6. Testing procedure of DResNet

3. Methodology

3.1. Data augmentation

In this project, we implemented four kinds of data augmentation methods, namely Gaussian Sampling, Gaussian Noise, DCGAN(Deep Convolutional Generative Adversarial Network)and Auto Encoder. In this section we will give brief introductions to all of them.

3.1.1 Gaussian Sampling

The simplest and most straightforward data augmentation method. Its main idea is to compute the mean and variance of each feature in the data, and then randomly sampling from the corresponding Gaussian distribution to generate augmented data.

The main advantage of this method is its simplicity and ease of implementation, requiring no additional training data or models and operating at high speeds. However, its disadvantages are also evident, as the randomness of Gaussian sampling can result in a higher probability of collecting outliers, leading to a decrease in the quality of the augmented data.

3.1.2 Gaussian Noise

This method is similar to the one described in the previous section3.1, which also involves computing the mean and variance of each feature in the original data and then sampling from the corresponding Gaussian distribution. However, the difference is that the sampled data is not directly used as the target data, but rather treated as noise and **weighted-averaged with randomly selected data from the original dataset** to obtain a new vector as the data augmentation-generated vector.

Compared to Gaussian sampling, this method can obtain more realistic data by incorporating the original vectors, effectively expanding the dataset while maintaining the stability of the data distribution.

3.1.3 DAGAN(Deep Convolutional Generative Adversarial Network)

DCGAN[2] is a GAN method that utilizes generative adversarial networks to obtain high-quality data. Its principle involves:

1. Constructing a generator and a discriminator using convolutional neural networks;

2. Adversarial training, in which the generator and discriminator are trained alternately, with the generator attempting to deceive the discriminator by generating fake data, while the discriminator must accurately determine whether the input data is real or fake

3. In the process of competing with each other during training, the quality of the generated data gradually improves, and once trained well, the generator can be used as a tool for data augmentation to generate high-quality data.

It is worth noting that training GAN networks is a very complex process, and the evaluation of its training effectiveness is quite subjective, making the training process difficult.

3.1.4 Auto Encoder

Auto Encoder algorithm is a type of unsupervised learning algorithm. In data augmentation, Autoencoder can be used to generate new data samples, thereby expanding the training set and improving the model's generalization performance.

The core idea of Auto Encoder is to compress the input data to a low-dimensional space by encoding it, and then reconstruct it back to the original space through the decoder. The encoder and decoder can be implemented through deep neural networks.

The structure of encoder-decoder network is as shown in Fig 7.

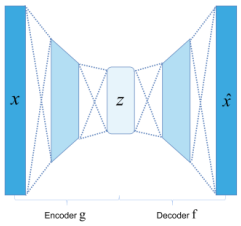

Figure 7. Illustration for Encoder-Decoder Network

The basic process of using Autoencoder for data augmentation is as follows:

1. Train an Auto Encoder model using the original data. During training, input data is encoded into feature vectors by an encoder, and then decoded into reconstructed data by a decoder. The training objective is to make the reconstructed data as close as possible to the original data.

2. After training, the Autoencoder model can be used to generate new data samples. Specifically, randomly select some samples from the original data, input them into the encoder to obtain feature vectors, and then **apply some random transformations to the feature vectors** (such as adding or subtracting noise, scaling, rotating, etc.) before inputting them into the decoder to obtain new data samples.

3. The generated samples can be used together with the original data samples as augmented data for training the model.

The advantages of using Auto Encoder for data augmentation are that the generated data samples retain the features of the original data, which can improve the model's generalization performance. Additionally, Auto Encoder is easy to train, making it a good choice for our task.

4. Experiments

In this section, we will illustrate and discuss our experiment results in detail.

4.1. General Results

In this project, we first conducted 15-fold cross validation on SEED dataset with two conventional methods, SVM and MLP, and four transfer learning methods we mentioned above. Here we report the main experiment results in Table 1.

More intuitively, we draw the results of all the 15 folds with SVM, MLP ADDA and DResNet (the best of each of

Method	Model	Average Accuracy	Highest Accuracy	Std
Conventional	SVM	0.4478	0.6411	0.0872
	MLP	0.545	0.7232	0.0842
Domain Adaptation	ADDA	0.5946	0.8124	0.0858
	DANN	0.5823	0.8047	0.0874
Domain Generalization	DG-DANN	0.6087	**0.8287**	0.0939
	DResNet	**0.6334**	0.7542	**0.0737**

Table 1. General results with different methods

Domain Adaptation and Domain Generalization). The result are shown in Fig 10.

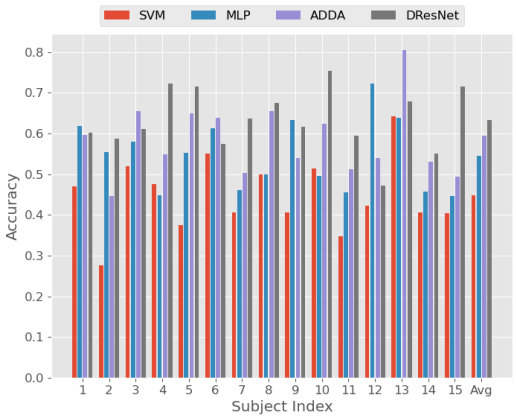

Figure 8. Comparisons between different methods across 15-fold validation

From the results, we can see that:

1. The conventional machine learning methods like SVM and MLP perform poor on the SEED classification task. Specifically, SVM only achieves 0.4478 and MLP achieves 0.545 average accuracy and in some fold they performs even just slightly better than random picking (like SVM in fold2 with accuracy 0.2758).

2. All the transfer learning methods we used outperform the conventional machine learning methods like SVM and MLP. The best method is DResNet which outperforms SVM and MLP by 18.56% and 8.84% respectively.

3. The Domain Generalization methods perform better than the Domain Adaptation methods on this task. The reason for this maybe:

    - The DA method aligns features or instances between the source domain and the target domain. However, in this task, the source domain actually consists of 14 different domains, where complex differences may exist and lead to poor alignment performance by the DA method. In addition, the different sizes of the domains may also affect the effectiveness of the DA method.

    - In contrast, the DG method achieves the desired functionality by generalizing across multiple domains, which can better capture both the commonalities and differences among domains. Under the complex conditions of the source domain in this task, the DG method is capable of achieving better performance.

4.2. Experiment on source domain data size

Sufficient source domain data is crucial for domain alignment and generalization in transfer learning. However, in this task, there is a significant difference in the size of the source and target domains (ratio of 14:1), which may cause the model to underfit the target domain, thus failing to adapt well to the target domain features and leading to a decrease in performance on the target domain. Additionally, larger data volumes can increase model training time and memory usage.

Therefore, we conducted experiments with different training data sizes. The experimental setup is as follows:

- We randomly select a certain proportion of the training data to form a new training dataset and train the model on this new dataset.

- We use both the DANN and DG-DANN algorithms (one for Domain Adaptation and one for DOmain Generalization) for transfer learning for comparison.

The experimental results are shown in Table 2.

Proportion	DANN	DG-DANN
0.001	0.4196	0.4024
0.003	0.4374	0.4561
0.005	0.4675	0.5137
0.008	0.5088	0.5312
0.01	0.5196	0.5024
0.02	0.558	0.5671
0.05	0.5778	0.5924
0.1	0.5797	0.5826
0.2	0.5841	0.6056
0.3	**0.5901**	0.5997
0.5	0.5854	0.6018
0.8	0.5769	**0.6099**
1.0	0.5823	0.6087

Table 2. Performance with different proportion of training data sizes with DANN and DG-DANN

More intuitively, the results are plotted as in Fig 9.

From the result, we can see that:

Figure 9. Performance with different proportion of training data sizes with DANN and DG-DANN

1. In this task, the training data (source domain data) is relatively abundant, and only a small portion (about 5%) of the data is needed to represent the source domain features well (the transfer accuracy is close to the accuracy obtained using all the training data).

2. It can be observed that even when the data volume is extremely small (e.g., $0.1\% \times 35070 = 35$), the transfer learning accuracy can still reach around 40%. This fully demonstrates the effectiveness of transfer learning algorithms such as DANN and DG-DANN, which can **learn the source domain features with only a minimal amount of training data**.

3. Both algorithms achieve their best performance when only a portion of the source domain data is selected (DANN-0.3, DG-DANN-0.8). This may be attributed to the randomness of the sampling process, or it could be due to the significant difference between the source and target domain data (ratio of 14:1). Using all the source domain data may have a negative impact on the effectiveness of transferring to the target domain, resulting in worse performance than when only a portion of the source domain data is used.

4.3. Data augmentation

In this project, we implemented several data augmentation methods and we conducted an experiment on these different methods.

The experimental setting are like:

- We generated augmentation data with the same size as the original data by different augmentation methods including: Gaussian Sampling, Gaussian Noise and Auto Encoder.

- Then we set the training data with three data composition: (a)the original data only, (b)the augmented data only and (c)the original data and the augmented data.

- Using the three setting of the training data, we conducted the DANN algorithm on the SEED dataset.

The experimental results are shown in Table 3.

Augmentation methods	original data	augmented data	combined data
Gaussian Sampling	0.5823	0.5068	**0.5838**
Gaussian Noise	0.5823	0.5989	**0.5955**
Auto Encoder	0.5823	0.5847	**0.5860**

Table 3. Comparison between different augmentation methods and data composition

From the results, we can see that:

1. The Gaussian Noise method and Auto Encoder method are both able to generate data that match the original domain well since when using the augmented data only, the performance of DANN doesn't decrease (indeed,even increase by 1.66% and 0.24% respectively).

2. The Gaussian Sampling method doesn't produce data that perfectly match the original domain and thus makes the performance worse when only using the augmented data. Through comparing the augmented data and the original data, we figure that the reason for this maybe that the values of the original data vector belonging to different classes are actually very close. So when conducting Gaussian Sampling, some outliers will confuse the model and thus significantly affect the performance.

3. The last two data augmentation methods are useful and data augmentation is needed in this task for that when using the combined data, all the three augmentation methods bring improvement on the performance (even the relatively inappropriate Gaussian Sampling).

4. The Auto Encoder method is not as good as the Gaussian Noise method and we figure that it results from the limitation of the size of the training dataset. There are no adequate data to train the Auto Encoder model to generate data that are premium enough.

Furthermore, indeed we implemented the data augmentation method DCGAN as well. But maybe due to the size of the dataset or the difficulty of training a GAN network, the DCGAN doesn't generate rational and high-quality data as we expected.

4.4. Visualization

As the visualization of features shown, we can see that the points form obvious clusters, demonstrating the effectiveness of transfer learning methods. However, it can also be seen that green and blue points tend to mix up and harder to discriminate than others, which stands for neutral and sad

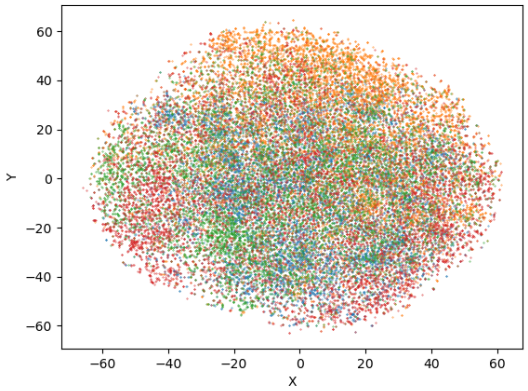

Figure 10. T-SNE visualization after DANN

emotion. These two emotions are key to the improvement the classifier. Further work can be done on how to discriminate these two emotions more accurately.

5. Conclusions

In this project, we implemented multiple methods to do sentiment classification task on the SEED4 dataset, including conventional machine learning method SVM, deep learning method MLP, domain adaptation transfer learning method (ADDA and DANN) and domain generalization transfer learning method (DG-DANN and DResNet). Furthermore, we conducted extensive experiments and explore several topics we are interested in, such as the impact on reduce the size of source domain data, the improvement that data augmentation can bring and so on. We not only deepen our understanding about the transfer learning methods, but also learn about more interesting topics.

References

[1] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[2] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[3] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.