Self-Attentive Adversarial Domain Adaptation for EEG-Based Emotion Recognition

Zenan Li, Xian Xu
{emiyali, jason73}@sjtu.edu.cn

Date: March 23, 2022

Abstract

EEG-based emotion recognition is a challenging task due to the individual differences across subjects and non-stationary characteristic (i.e. domain shift). Thanks to the development of transfer learning, a number of domain adaptation methods have been introduced into this field to promote the effect of emotion recognition. However, current works often treat the inputs equivalently, i.e. they ignore the different transferable effects of the inputs. In this paper, we propose an self-attentive domain adaptation method along with class-level alignment and data augmentation. We implement multiple baselines, including conventional machine learning and deep learning methods, domain adaptation methods and domain generalization methods. Extensive experiments are done on the SEED dataset, whose results strongly support our method's effectiveness (the target domain accuracy exceeds DANN by nearly 4%).

1 Introduction

Emotion recognition focuses on the recognition of human emotions based on a variety of modalities, such as audio-visual expressions, body language, physiological signals, etc. In recent years, due to the rapid development of noninvasive, easy-to-use and inexpensive electroencephalography (EEG) recording devices, several EEG datasets (e.g. SEED Zheng et al. (2018)) have been open-sourced and EEG-based emotion recognition has received an increasing amount of attention in both research and applications.

As reported in Zheng and Lu (2016), due to the individual differences across subjects and non-stationary characteristic (i.e. domain shift) of EEG, conventional classifiers like SVM or MLP usually perform poorly in the EEG recognition tasks. Fortunately, along with the quick development of transfer learning Torrey and Shavlik (2010), a lot of domain adaption methods have been introduced into this field, helps to pave the way for effective EEG-based emotion recognition. For example, Li et al. (2018a) adapts the framework of DANN Ganin et al. (2016), promoting the emotion classification accuracy by incorporating with domain adversarial training. Li et al. (2018b) refines DANN with bi-hemispheres (BiDANN) to better fit in with EEG recognition tasks, while Luo et al. (2018) introduces two-step training procedure and WGAN loss for stability and better convergence.Typically, the EEG emotion recognition task can be roughly partitioned into

two steps: feature extraction and classifier design. Besides, domain adaptation is incorporated into the feature extraction stage to mitigate the domain shift effect between the source and target domain, i.e. learn a similar feature mapping for both source and target domain inputs.

However, current domain adaptation methods in EEG usually treat all the inputs equivalently, which may lead to under transfer or negative transfer Wang and Zhang (2020). To distinguish between inputs with different transferable effects, we develop an entropy-based self-attentive adversarial domain adaptation method to relax the alignment on well aligned samples (i.e. with low entropy) and focus on those poorly aligned samples (i.e. with high entropy). We also use WGAN Gulrajani et al. (2017) to generate extra data Luo et al. (2020) for domain generalization. Besides, we go a step beyond domain-level alignment, using the **triplet loss** Deng et al. (2018) to separate apart samples with different class labels, further enhancing classifier's discriminative capability.

In a nutshell, this paper's contribution are three-folds:

- We implement abundant baselines for the EEG recognition task, including conventional classifiers: SVM, MLP, and ResNet He et al. (2016), domain adaptation models: DANN Ganin et al. (2016) and ADDA Tzeng et al. (2017), domain generalization methods: IRM Arjovsky et al. (2019) and REx Krueger et al. (2021).
- We come up with a novel **self-attentive** adversarial domain adaptation framework, along with **data augmentation** and **class-level alignment**.
- We conduct experiments on the SEED dataset Zheng et al. (2018), using the leave-one-subject strategy to do cross validation, whose results strongly support our method's effectiveness.

2 Preliminary

In this section, we will briefly review exsisting methods for Out-of-Distribution (OOD) problems, laying the foundation for our approach. Typically, we divide the existing methods into two classes, domain adaptation and domain generalization, based on whether auxiliary data (from different domains) is used during training.

2.1 Domain Adaptation

Domain adaptation methods incorporate with unlabeled target domain inputs to learn a domain-level invariant feature extractor.

One of the most famous domain adaptation framework is DANN Ganin et al. (2016), whose framework is shown in Figure 1. Its main insight is to introduce a **domain classifier** to distinguish between features from source and target domain. Assuming the model works with input samples $\mathbf{x} \in X$, where X is some input space and certain labels y from the labels space Y. Suppose there exist two distributions $\mathcal{S}(x, y)$ and $\mathcal{T}(x, y)$ on $X \otimes Y$, which are unknown, similar but different. Our ultimate goal is to predict labels y given the input $\mathbf{x}$ from target distribution with only access to labeled samples from the source domain. We use the binary

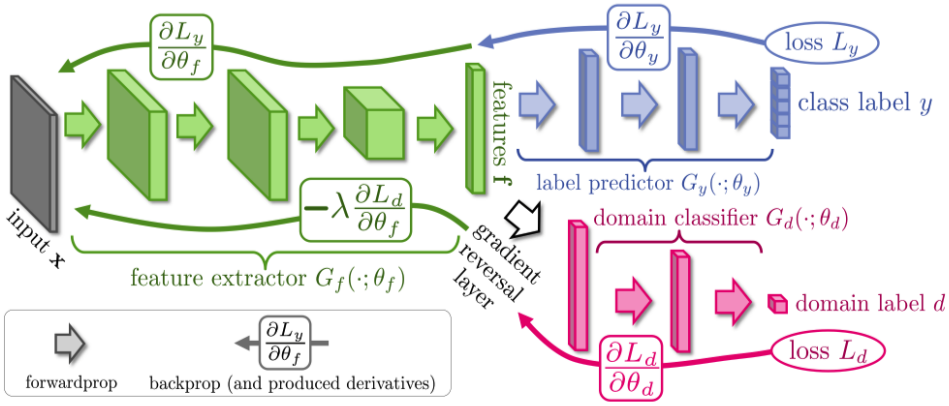


Figure 1: DANN framework (cited from Ganin et al. (2016)).

variable $d \in \{0, 1\}$ to denote whether a sample is from the source domain or the target domain.

For the feed-forward architecture in Figure 1, the input $\mathbf{x}$ is first mapped by a mapping G_f (the feature extractor) to a D-dimensional feature vector $\mathbf{f} \in \mathbb{R}^D$. The parameters of all layers in this mapping is denoted as θ_f, so we have $\mathbf{f} = G_f(\mathbf{x}; \theta_f)$. Then, the feature vector $\mathbf{f}$ is fed into the label classifier G_y with parameter θ_y to get the predicted label $\hat{y} = G_y(\mathbf{f}; \theta_y)$. It will also be fed into the domain classifier G_d to get the predicted label $\hat{d} = G_d(\mathbf{f}; \theta_d)$.

During the learning phase, we aim to minimize the label prediction loss on the source domain. So we can optimize the cross-entropy loss for source domain inputs, which ensures the discriminative power of the features $\mathbf{f}$ and the overall good prediction performance of the combination of the feature extractor and the label classifier on the source domain. At the same time, we wish to make the features $\mathbf{f}$ domain-invariant. That is, we want to minimize the difference between distributions $\mathcal{S}(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f)|\mathbf{x} \sim \mathcal{S}(\mathbf{x})\}$ and $\mathcal{T}(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f)|\mathbf{x} \sim \mathcal{T}(\mathbf{x})\}$, which can make the label prediction accuracy on the target domain to be in according with the source domain.

The challenge is that $\mathbf{f}$ is high-dimensional and the two distributions are non-trivial. So DANN's idea is to measure the dissimilarity by looking at the loss of the domain classifier G_d. At training time, in order to obtain domain-invariant features, we seek the parameters θ_f of the feature mapping that **maximize** the loss of the domain classifier (making the two feature distributions as similar as possible), while simultaneously seeking parameters θ_d to **minimize** the loss of the domain classifier. Specifically, the loss function can be formulated as:

$$\max_{\theta_d} \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d) = \sum_{i=1..N, d_i=0} L_y^i(\hat{y}_i, y_i|\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\hat{d}_i, d_i|\theta_f, \theta_d) \tag{1}$$

where L_y and L_d are the cross-entropy losses for the label and the domain, respectively. We can see that θ_f and θ_d play an adversarial game in Equation 1, so we need to modify the model architecture to correct the backpropagation. As for this purpose, DANN proposes to put a **gradient reversal layer** (GRL) between the feature extractor and the domain classifier. Hence, the parameters θ_f will be updated to maximize the loss L_d during stochastic gradient descent. Specifically, GRL's pesudo-function $R_\lambda(\mathbf{x})$ is (you can find our

implementation in `models.py`):

$$R_\lambda(\mathbf{x}) = \mathbf{x},\ \frac{dR_\lambda}{d\mathbf{x}} = -\lambda\mathbf{I} \tag{2}$$

As another domain adaptation example, ADDA Tzeng et al. (2017) introduces two separately feature extractors for the source and target domain to obtain more flexibility for domain alignment. Besides, it uses the two-step training procedure. In the pre-training phase, only inputs from source domains will be fed into the framework to update the parameters of the source feature extractor and the label classfier. In the adv-training phase, the parameters of the source feature extractor and label classifier will be fixed, both inputs from source and target domain will be fed into the framework to update the parameters of the target feature extractor and the domain classifier. Additionally, ADDA resorts to the GAN Goodfellow et al. (2014) loss to replace the min-max game in DANN for more stable adversarial training between target feature extractor and domain classifier:

$$\begin{aligned}\min_{\theta_d,\theta_f^t} L_{adv_d} &= -\mathbb{E}_{\mathbf{x}_s\sim\mathcal{S}(\mathbf{x})}[\log G_d(G_f^s(\mathbf{x}_s))] - \mathbb{E}_{\mathbf{x}_t\sim\mathcal{T}(\mathbf{x})}[\log(1 - G_d(G_f^t(\mathbf{x}_t)))] \\ \min_{\theta_f^t} L_{adv_f^t} &= -\mathbb{E}_{\mathbf{x}_t\sim\mathcal{T}(\mathbf{x})}[\log G_d(G_f^t(\mathbf{x}_t))]\end{aligned} \tag{3}$$

where G_f^s and G_f^t denotes the source and target feature extractor, respectively.

2.2 Domain Generalization

Domain generalization wishes to learn generalizable feature representations without access to inputs from target domain. Thus, domain generalization has stricter conditions, and we need to come up with methods to generalize to unseen domains only utilizing the training data.

A common assumption for domain generalization is that the data points are generated from different environments $\mathcal{E}$, while training data is generated from $\mathcal{E}_{tr}$. Our wish is that the feature extractor $G_f : X \to F$ can learn environment-invariant feature representations, such that the classifier $G_y : F \to Y$ can be simultaneously optimal for all environments. IRM Arjovsky et al. (2019) formulates this assumption as:

$$\begin{aligned}&\min_{G_f,G_y} \sum_{e\in\mathcal{E}_{tr}} R^e(G_y \circ G_f) \\ &s.t.\ G_y \in \arg\min_{\overline{G_y}} R^e(\overline{G_y} \circ G_f),\ \text{for all } e \in \mathcal{E}_{tr}.\end{aligned} \tag{4}$$

where R^e is the empirical risk in environment e. Equation 4 is a challenging bi-leveled optimization problem, IRM goes a step further and instantiate it into the practical version:

$$\min_{G_f} \sum_{e\in\mathcal{E}_{tr}} R^e(G_f) + \lambda||\nabla_{G_y|G_y=1.0} R^e(G_y \circ G_f)||^2 \tag{5}$$

here $G_y = 1.0$ is a scalar and fixed "dummy" classifier, whose gradient norm is used to measure the optimality of the dummy classifier at each environment e. Equation 5 is treated as a domain generalization baseline in our experiment.

As another domain generalization example, risk extrapolation (REx) Krueger et al. (2021) proposes to

use a variance penalty across different environments, that is:

$$R_{\text{V-REx}} = \mathbb{V}_{e\in\mathcal{E}_{tr}}[R^e] + \sum_{e\in\mathcal{E}_{tr}} R^e \tag{6}$$

We believe that the variance penalty can help to make the learned representation invariant across different environments, Equation 6 is also treated as a domain generalization baseline in our experiment.

3 Methodology

In this section, we will introduce our proposed model in detail, an overview of our model is illustrated in Figure 2. We will succeed the notations in Section 2.

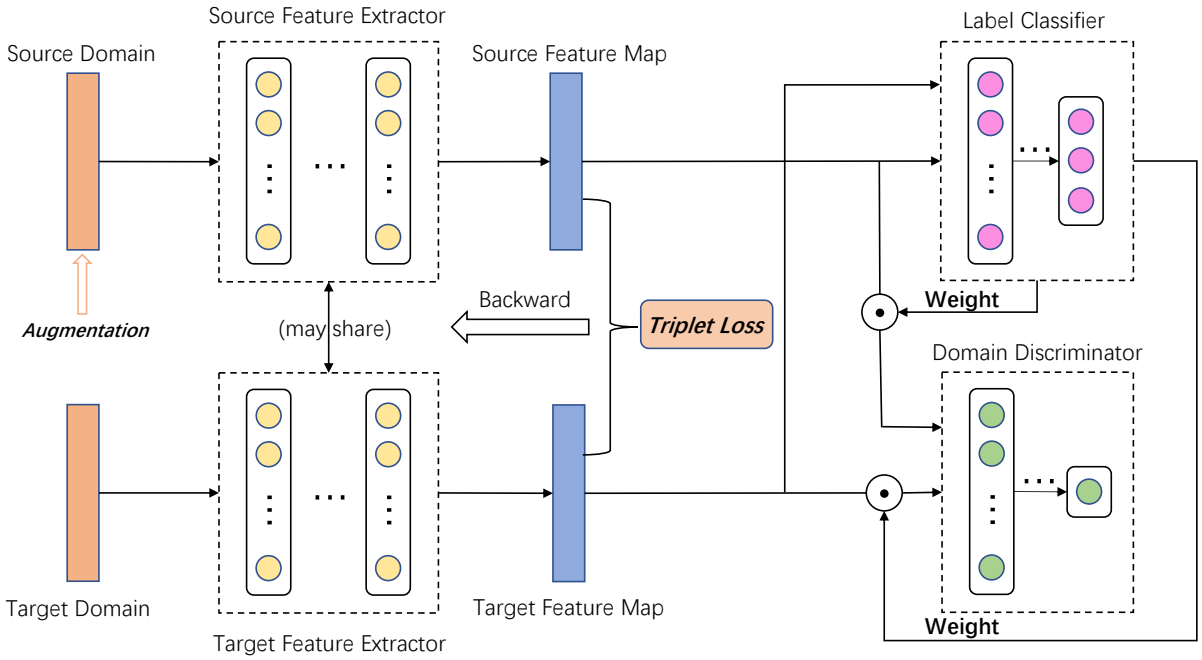


Figure 2: Overview of our proposed model. Typically, we will do data augmentation for inputs from source domain, the source and target feature extractor can choose to share weights or not. We add an additional triplet loss to achieve the class-level alignment for discriminative capability. Finally, we calculate the entropy functional for each input after the label classifier and use it to re-weight the adversarial loss.

3.1 Data augmentation

We start with data augmentation, the selective plug-in component in our framework. We take the idea from Luo et al. (2020) and use WGAN to generate realistic-like EEG data. Wasserstein distance can provide useful and smooth gradients for GAN training even if the two distributions have no overlaps. However, it has an intractable 1-Lipschitz constraint, so we use the gradient penalty Gulrajani et al. (2017) to instantiate the

practical loss function:

$$\min_{\theta_G}\max_{\theta_D} L(\mathbf{x}_r,\mathbf{x}_g)=\mathbb{E}_{\mathbf{x}_r\sim X_r}[D(\mathbf{x}_r)]-\mathbb{E}_{\mathbf{x}_g\sim X_g}[D(\mathbf{x}_g)]-\lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\hat{X}}[(||\nabla_{\hat{\mathbf{x}}}D(\hat{\mathbf{x}})||-1)^2] \tag{7}$$

where $\mathbf{x}_r,\ \mathbf{x}_g$ denote the real/generated data, $D(;\theta_D),\ G(;\theta_G)$ denotes the discriminator/generator in the WGAN, and $\hat{\mathbf{x}}$ represents the data points sampled from the straight line between the real distribution X_r and generated distribution X_g. Simply optimize according to Equation 7 can generate unreliable data which deviates from the source domain seriously. To mitigate this effect, we introduce a calibration classifier (specifically, a ResNet He et al. (2016)) to selectively add the generated data points. Typically, only those data points with classification probability $> \tau$ will be added into the training set. The data augmentation procedure is summarize in Algorithm 1.

Algorithm 1: The data augmentation procedure of WGAN.

Input: Real dataset $X_r=\{\mathbf{x}_r^i\}_{i=1}^m$, calibration classifier $C: X\rightarrow Y$, probability threshold τ, expected generated data number n.

Output: Generated dataset X_g, Y_g.

```
X_g, Y_g ← {}, {};
while len(X_g)<n do
    X_all_g=WGAN(X_r,noise);
    Y_all_g, class_conf = C(X_all_g);
    for i in X_all_g do
        if class_conf[i]>τ then
            X_g, Y_g=X_g ∪ X_all_g[i], Y_g ∪ Y_all_g[i]; # add generated data selectively
return X_g, Y_g
```

3.2 Class-level Alignment

Current domain adaptation methods usually focus on the domain-level confusion, the discriminative power between classes is not involved. However, the discriminative power of feature extractor can be hurt after adversarial training since it needs to update its parameters to narrow the gap between the source and target feature map. Thus, we need an additional regularization term to pull together samples with the same labels in the embedding space while separate apart those with different labels.

Triplet loss Deng et al. (2018) satisfies this need perfectly. The loss tries to enforce a margin m between each pair of samples from one class to all other classes. It allows samples to enforce the distance and thus discriminative to other classes. Typically, it can be formulated as:

$$L_{tri}(\theta_f)=\sum_{\substack{\mathbf{x}_i\in\mathcal{S}(\mathbf{x})\cup\mathcal{T}(\mathbf{x})\\ y_a=y_p\neq y_n}}[m+d_{a,p}-d_{a,n}]_+ \tag{8}$$

where d is a certain distance metric, which is instantiated as the cosine similarity in our experiment.

One remaining problem is how to sample data from the dataset. Intuitively, a target generalized classifier need to distinguish accurately between inputs from the target domain. Since inputs from the target domain are unlabeled, we propose to use assign pseudo-labels to target samples and optimize the triplet loss as if they were true labels. Similar to Section 3.1, we also set a probability threshold τ and only sample those target inputs with classification confidence larger than τ. Finally, we follow the sampling strategy in Deng et al. (2018) to randomly select samples and conduct gradient descent according to Equation 8.

3.3 Self-Attentive Adversarial Domain Adaptation

From Equation 1 and Equation 3, we can see that both DANN and ADDA treat the inputs equivalently, i.e. they ignore the different transferable effects of the input samples. Thus, the previous domain adaptation methods without exploiting underlying multimode structures of the EEG dataset maybe prone to either under transfer or negative transfer. To promote positive transfer and combat negative transfer, we should find a technology to reveal the transferable degree of samples and then re-weight them to force the underlying distribution closer.

Now we need a measure to quantify the adaptable degree for different input samples. From information theory, the entropy functional is an uncertainty measure which nicely meets our need. If the sample can get a low entropy, it can be regarded as a well-aligned transferable sample, else it is a poorly aligned sample. We adopt the conditional entropy as the indicator to weight the adversarial loss Wang and Zhang (2020) and the second term in Equation 1 (the adversarial loss) is extended as:

$$\begin{aligned} L_{adv}(\theta_f, \theta_d) &= -\sum_{i=1..N} (1 + H_p^i) L_d^i(\hat{d}_i, d_i | \theta_f, \theta_d) \\ where \quad H_p^i &= -\frac{1}{C} \sum_{c=1}^{C} p_c^i \log(p_c^i) \end{aligned} \tag{9}$$

We can see from Equation 9 that samples with higher entropy will be added weight in the adversarial loss for better alignment, while those with lower entropy will be payed less attention for they have already been aligned well.

3.4 Pipeline

For the training pipeline, as discussed in Section 2.1, we adopt the same framework as ADDA. That is, we first pretrain the source feature extractor and the label classifier, then fix their parameters and train the target feature extractor and the domain classifier adversarially. The training procedure is summarized in Algorithm 2.

Algorithm 2: The training pipeline of the proposed Self-attentive DA framework.

Input: Source domain dataset $X_s = \{\mathbf{x}_s^i\}_{i=1}^m$, $Y_s = \{y_s^i\}_{i=1}^m$ and target domain dataset $X_t = \{\mathbf{x}_t^i\}_{i=1}^n$, # pretraining iterations N_{pre}, # adversarial training iterations N_{adv}, # domain classifier iterations N_d.

Output: Predicted target domain dataset labels $\hat{Y}_t$.

for *iterations in* N_{pre} **do**
    Update θ_f^s and θ_y by gradient descending according to Equation 1. # pretraining.
Initialize θ_f^t with θ_f^t;
for *iterations in* N_{adv} **do**
    for *sub-iterations in* N_d **do**
        Update θ_d by gradient descending according to Equation 9. # train the domain classifier.
    Update θ_f^t by gradient ascending according to Equation 9. # train the target feature extractor.
Predict target domain dataset labels $\hat{T}_t = G_y(G_f^t(X_t))$.
return $\hat{Y}_t$

4 Experiments

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

4.1 Experiment Setup

For the environment, all the experiments are done on i9-10920X CPU @3.50GHz along with a GeForce RTX 3090 GPU. The SVM baseline is implemented by the scikit-learn Pedregosa et al. (2011) library, while other deep learning based models are all implemented by PyTorch.

The experiment are done on the SEED Zheng et al. (2018) dataset, which is composed of three types of emotions: happy, neutral and sad, from 15 subjects' EEG emotional signals. We use the released handcraft features, namely, the differential entropy (DE) in SEED, as the input to feed our model. The dataset has 62 channels, thus, the input dimension is $5 \times 62 = 310$.

We also list the hyperparameter setting here for reproducibility:

- We use the linear kernel for SVM by default, and its regularization parameter C is set as 0.1.
- For all the models, We use the Adam optimizer with learning rate set to 1e-5. We do not use weight decay or dropout since they have been tested to hurt the model's performance. The batch size is set to 32. The other hyperparameters are kept the same with the original paper (e.g. $\lambda = 0.5$ in DANN).
- We use the same backbone structure for all the models: The feature extractor is a two-layer MLP with hidden size 64, the domain classifier has three layers with hidden size 32, and the label classifier has three layers with hidden size 32. The activation function is ReLU.

4.2 General Results

We report the main experiment results in Table 1. Specifically, we record the highest accuracy on the target domain for each fold, and calculate the mean and std w.r.t. the results of all the 15 folds.

Table 1: 15-fold cross validation results on SEED dataset. Specifically, we compare our methods performance to multiple baselines, including conventional machine learning methods, domain adaptation methods and domain generalization methods. We only list 5 folds' and the average results here due to the space limit.

Method	Model	Fold 0	Fold 3	Fold 6	Fold 9	Fold 12	Mean(↑)±Std(↓)
Conventional	SVM	0.3444	0.7516	0.5687	0.3300	0.7166	0.5504±0.1531
	MLP	0.6909	0.7319	0.6405	0.7065	0.9582	0.6788±0.1098
	ResNet	0.8180	0.8259	0.6747	0.7837	0.9596	0.6957±0.1565
Domain Generalization	IRM	0.8014	0.7398	0.6210	0.7257	0.9611	0.7029±0.1174
	V-REx	0.8008	0.7537	0.6458	0.7401	0.9634	0.7245±0.1236
Domain Adaptation	DANN	0.8226	0.8869	0.8533	0.8757	0.9753	0.8189±0.0906
	ADDA	0.8303	0.8755	**0.8616**	0.8857	0.9704	0.8302±0.0924
Self-Attentive	SADA (Ours)	**0.8324**	**0.9284**	0.8362	**0.8940**	**0.9860**	**0.8541±0.0893**

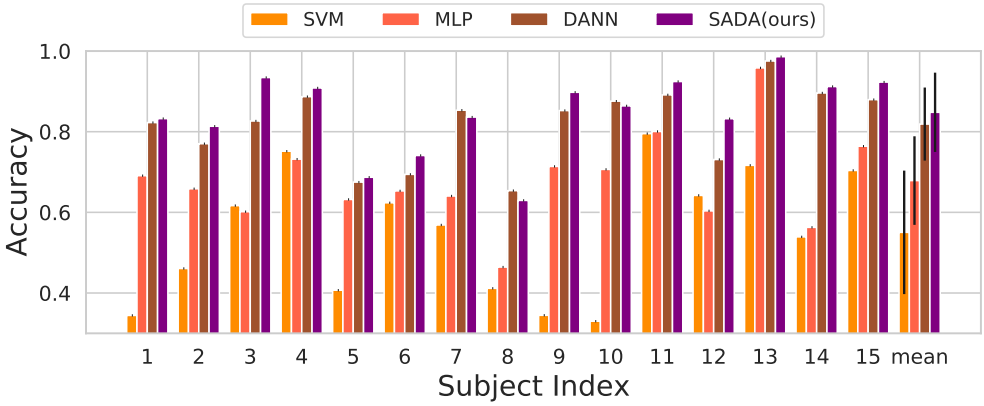
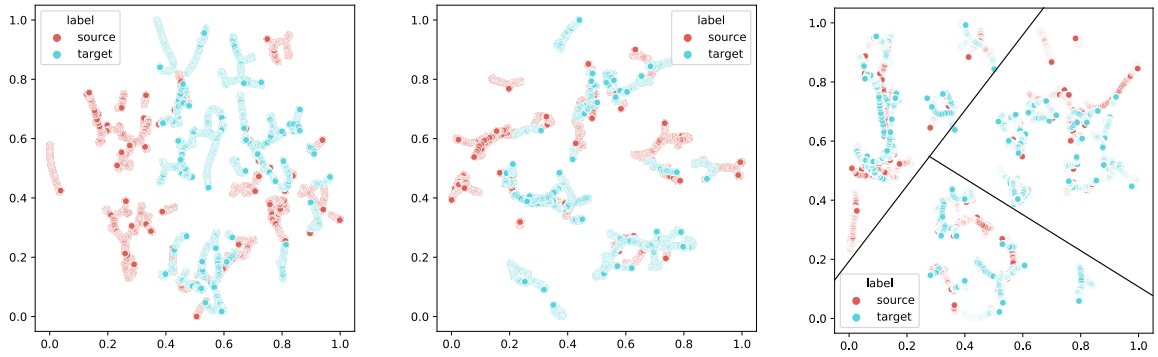


Figure 3: Comparisons between different methods across 15-fold validation.

From Table 1 we can see that conventional machine learning models like SVM can perform really poor on the EEG emotion recognition task. SVM can only achieve an average target domain accuracy of 55%, on some fold (fold 9) even don't work and perform randomly (33%=1/3 classes). The traditional deep learning methods like MLP and ResNet can promote the performance to some extend (from 55% to approximately 70%), but still not very ideal yet. We can also see that domain generalization methods only boosts the performance marginally, which may be the result of the unknown data generating process. As reported in Federici et al. (2021), there doesn't exist a universal domain generalization method due to the multiple types of data generating process. In comparison, the domain adaptation does perform well with access to the auxiliary target domain data, DANN and ADDA can promotes the target domain accuracy to be higher than 80%, which is a huge leap forward compared to the other baselines.

For our self-attentive domain adaptation method (referred as SADA in the table), we can see the results more intuitively in Figure 3. The figure illustrates that our method can achieve the best accuracy in 12 out of 15 folds, it can even exceed DANN by approximately 10% on fold 3. Our method also achieve the best mean (exeeds DANN by 4%) and std of the accuracy on the target domain.



(a) MLP (b) DANN (c) SADA (Ours)

Figure 4: Embedding Visualization for different methods.

To get a more intuitive impression of our method's capability, we use t-SNE Van der Maaten and Hinton (2008) to visualize the embedding after the feature extractor module for MLP, DANN and SADA. As shown in Figure 4, the embeddings from source and target domain of MLP are separated out, increasing the difficulty of generalization. The embeddings of MLP also cannot be classified well since they are almost in one cluster. As for DANN, we can see that the embeddings from source and target are much better aligned with each other than MLP. The classification is of course simpler for DANN, but the embeddings are still not clustered well and there are still many outliers in Figure 4(b). For our method (SADA), we can see from Figure 4(c) that the source and target embeddings are even better aligned than DANN. Besides, both source and target domain embeddings can be divided perfectly into three clusters (in according with SEED's three label classes), strongly supports our method's effectiveness. From the results we can believe that the self-attentive module does help to align the domain embeddings, while the triplet loss makes sense to cluster out inputs with different labels.

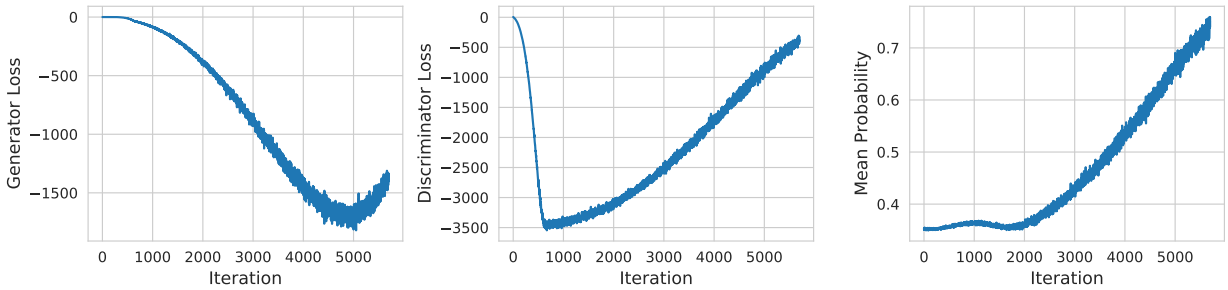


Figure 5: Training curve for WGAN augmentation.

For the data augmentation process, we visualize the loss and calibration probabilities curve of the WGAN. We can clearly see from the middle curve that the loss of the generator decreases sharply in the starting iterations and increases as following. In comparison, the left curve illustrates that the loss of the generator keeps stable in the starting iterations then decreases afterwards, performing adversarially to the discriminator. These two curves validate our implementation's validity. The right curve reports the average classification confidence of the calibration classifier (in our experiment, a ResNet), we can see the probability increases as the generator loss drops.

4.3 Ablation Study

We do ablation studies on the three key components of our model: data augmentation, triplet loss and self-attentive re-weighting, to validate their effectiveness.

Table 2: Abalation study results on SADA. We remove the three key components respectively to measure their significance in our model. Here "aug" denotes data augmentation, "trip" represents the triplet loss, "SA" is the self-attentive re-weighting.

Model	SADA w/o aug	SADA w/o trip	SADA w/o SA	SADA
Mean(↑)	0.8338	0.8409	0.8315	**0.8541**
Std(↓)	0.0944	**0.0871**	0.0915	0.0893

The results in Table 2 show that each one of the modules will hurt the performance greatly (more than 1% 3%), while the self-attentive re-weighting module is the most significant one (without which the performance drops approximately 2.5%) among the three components. We can also find an interesting aspect that reduce the triplet module can decrease the std of the accuracy on the target domain. We attribute this effect to the conflict between the domain and class-level alignment.

5 Conclusion

In this paper, we propose a self-attentive adversarial domain adaptation method along with class-level alignment and data augmentation. We implement multiple baselines, including conventional machine learning (SVM) and deep learning (MLP, ResNet) methods, domain adaptation methods (DANN, ADDA), and domain generalization methods (IRM, V-REx). Extensive experiments have been done on the SEED dataset and we find that domain adaptation methods usually outperforms domain generalization methods with access to the auxiliary target domain data. Results our methods have achieved the best mean (exceeds DANN by 4%) and std accuracy on the target domain, proving its effectiveness. Finally, we conduct ablation study on the three key components of our model and find that delete each one of the modules will hurt the performance greatly, while the self-attentive re-weighting module is the most significant one among the three components.

Acknowledgements

First, we would like to thank every team member for our dedication to this project. Although both of us was responsible for a different part of the project, we have cooperated well and completed the project efficiently. Second, we sincerely thank Prof. Lu for his guidance in transfer Learning and excellent teaching, this project is interesting and we have learned a lot from this experience. Finally, thank you for reading till this end, hope this paper can inspire you to some extend.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. *arXiv preprint arXiv:1907.02893*.

Deng, W., Zheng, L., and Jiao, J. (2018). Domain alignment with triplets. *arXiv preprint arXiv:1812.00893*.

Federici, M., Tomioka, R., and Forré, P. (2021). An information-theoretic approach to distribution shifts. *Advances in Neural Information Processing Systems*, 34.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. (2021). Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR.

Li, H., Jin, Y.-M., Zheng, W.-L., and Lu, B.-L. (2018a). Cross-subject emotion recognition using deep adaptation networks. In *International conference on neural information processing*, pages 403–413. Springer.

Li, Y., Zheng, W., Cui, Z., Zhang, T., and Zong, Y. (2018b). A novel neural network model based on cerebral hemispheric asymmetry for eeg emotion recognition. In *IJCAI*, pages 1561–1567.

Luo, Y., Zhang, S.-Y., Zheng, W.-L., and Lu, B.-L. (2018). Wgan domain adaptation for eeg-based emotion recognition. In *International Conference on Neural Information Processing*, pages 275–286. Springer.

Luo, Y., Zhu, L.-Z., Wan, Z.-Y., and Lu, B.-L. (2020). Data augmentation for enhancing EEG-based emotion recognition with deep generative models. *Journal of Neural Engineering*, 17(5):056021.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Torrey, L. and Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global.

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

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Wang, S. and Zhang, L. (2020). Self-adaptive re-weighted adversarial domain adaptation. *arXiv preprint arXiv:2006.00223*.

Zheng, W., Liu, W., Lu, Y., Lu, B., and Cichocki, A. (2018). Emotionmeter: A multimodal framework for recognizing human emotions. *IEEE Transactions on Cybernetics*, pages 1–13.

Zheng, W.-L. and Lu, B.-L. (2016). Personalizing eeg-based affective models with transfer learning. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence*, pages 2732–2738.