Domain Adaptation on SEED

Comparison and Improvement

Ziyin Zhang
daenerystargaryen@sjtu.edu.cn

Junbo Wang
sjtuwjb3589635689@sjtu.edu.cn

Abstract—Electroencephalography features based emotion recognition has been a long pursuit and challenge of affective brain-computer interface community. However, limited by the electroencephalography signal's unstationary and individual difference, traditional supervised learning algorithm cannot empower neural network the ability to predict the emotion across different subjects. In this project, we adapt existing domain adaptation algorithms to a subset of SJTU Emotion EEG Dataset and compare their prediction results on it. Further, we do some improvement on these algorithms to make them perform better on this dataset. Among them, the best-performed algorithm reaches 80% around prediction precision, in contrast to 55% around by baseline algorithm. In addition, we deploy parallel training to gain training speed increase, which gives us a factor of 2 to 3 speed up.

Index Terms—brain-computer interface, domain adaptation, parallel machine learning

I. Introduction

Transfer learning has been increasingly gaining popularity in recent years, fueled by exploding volumes of data, development in computation power, as well as advance in pretrained models. Transfer learning is the branch of machine learning that deals with data from heterogeneous domains, where the disparity between the distributions of training data and test data pose novel challenges to classical learning algorithms. It has been widely studied in various fields, from dealing with images of different styles in computer vision to applying the knowledge learned from one language to another in natural language processing. It is also vital in computational psychology and neuroscience, as in these fields the data are usually collected from multiple human subjects, each being an independent domain.

Generally, transfer learning methods can be categorized into *domain adaptation* and *domain generalization*. Domain adaptation is the scenario where unlabeled test data can be accessed in the training stage, whereas in domain generalization any information about the test data is assumed unavailable during training. Being able to access the test data, domain adaptation is usually an easier task than domain generalization, as it may model the distribution discrepancy between domains and adjust the classifiers (or other task-specific models) accordingly. In this project, we focus on domain adaptation on the SJTU Emotion EEG Dataset, but also briefly touch upon domain generalization.

The rest of the work is organized as follows. In Section II, we summarize the related works on domain adaptation, and Section III gives a brief introduction to the dataset used in this project. Through Section IV to VII, we experiment with and give detailed analysis about different domain adaptation algorithms. Finally, we explore the application of parallel machine learning in domain adaptation in Section VIII, and draw the conclusion in Section IX.

II. Related Works

Transfer learning, especially domain adaptation, has long been subject to wide studies. The current approaches for domain adaptation in the community can be broadly classified into three categories.

The first is *domain shift*, where the discrepancy between heterogeneous domains is narrowed by either domain projection or sample selection. For the domain projection part, *Domain Invariant Projection* [2] and *Transfer Component Analysis* [13] both aim to minimize the distance between centroids of data samples after projection and kernelization. *Subspace Alignment* [7] and its variant *Landmarks Selection-based Subspace Alignment* [1] close the gap between source domain and target domain by aligning the two domains' principal subspace, while *Correlation Alignment* [15] achieves that end by aligning the second order moment between the two domains.

Sample selection based methods, on the other hand, assign different weights to different data samples in the source domain and target domain during the training stage, so that the classifier being trained learns to accommodate the target domain as well as the source domain. *Kernel Mean Matching* [10] is one example of this idea, where data points from the source domain are assigned weights to minimize the distance between the two domains' data distribution, and then the classifier is trained on the weighted source data. *Selective Transfer Machine* [6] is an extension to Kernel Mean Matching where the two steps are merged into one, while *Domain Adaptation SVM* [4] takes a different approach and gradually takes target data points into consideration during the training stage in a bootstrapping fashion.

With the rise of deep learning, these ideas have also been applied to neural networks in pursuit of higher performance. [12] applies kernelized Maximum Mean Discrep-

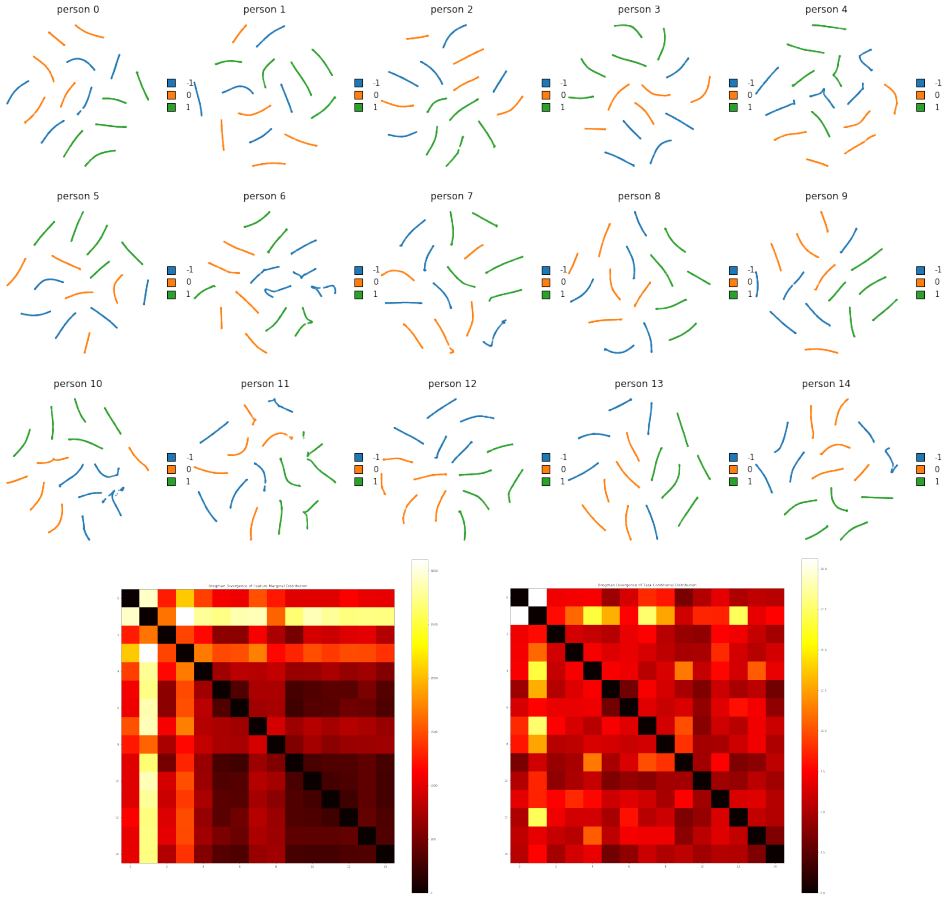

Fig. 1. Feature distributions visualization (top), Bregman divergence of feature marginal distribution (left) and Bregman divergence of task conditional distribution (right)

ancy minimization to features extracted by the network, while [16] performs Correlation Alignment instead. [9] applies Subsapce Alignment to deep learning features, and [11], [5] close the gap between source domain and target domain by simple batch normalization.

The second class of domain adaptation approaches, which we refer to as *parameter transfer*, models the data of source domain and target domain separately, and assumes a function from the domain information to the models' parameters, similar to the idea of meta learning. The most representative of these approaches is *Transductive Parameter Transfer* [14], and [20] is also similar in essence.

The third category is *adversarial learning* based on neural networks and gradient descent. [3] divides the network into shared encoder and private encoders to better extract features shared by source domain and target domain, and [18] proposes domain confusion loss to help the network learn domain independent features. *Domain Adversarial Neural Network* [8] and *Adversarial Discriminate Domain Adaptation* [17] further extend this idea by introducing the technique of gradient reversing.

III. SEED

The dataset we use here is a subset of *SJTU Emotion EEG Dataset* (*SEED*) [22], which includes 15 human subjects' 310-dimensional *electroencephalography* (*EEG*) features at 3394 time points. The EEG signals are recorded while the subjects are exposed to 15 segments of video clips intended to induce different emotions: positive, negative, and neutral. Our aim is to predict these 3 classes of emotions. Since every two subjects' EEG features distributions are not necessarily the same, i.e. non-i.i.d., and the sampled 15 subjects are few to some extent and may not be able to properly represent the emotions of all human kind, we need to distinguish these 15 subjects and determine whether they are similar enough.

To get a taste of the features distribution, we visualize the 15 subjects' performance by dimensionality reduction using *t-distributed Stochastic Neighbor Embedding* (*t-SNE*) [19], which can roughly show that each subject's performance is dramatically different from one another. To measure their EEG features distribution discrepancy more deeply and determine the granularity of domains, we figure out the *Bregman Divergence* [21] between each two subjects. Specifically, we measure the Bregman divergence

of the feature marginal distribution $P(X)$ and the task conditional distribution $P(Y|X)$, by which a distinct domain is characterized. The results shown in Figure 1 illustrate that both feature marginal distribution and task conditional distribution are different from each other, so it is necessary for us to treat each subject as an independent domain.

IV. Baseline

As baseline, we train a *Support Vector Machine* (*SVM*) on the mingled data of all fourteen source subjects. SVM is one of the most widely used and most robust machine learning algorithms, even after the rise of deep learning. Its main idea is to maximize the margin between different categories of the labeled data, with a parameter C to control the trade-off between fitting the majority of training data and making sure that samples fall on the right side of classification margins. The objective can be expressed as

$$\begin{aligned} \min_{\mathbf{w},b,\xi_i} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i, \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i, \ \forall i, \\ & \xi_i \geqslant 0, \ \forall i. \end{aligned} \tag{1}$$

After conducting cross-validation, we set C to 1×10^{-4}, and the classifier scores an average accuracy of 54.78% on the test set, leaving much room for improvement - as the data are uniformly distributed across the three sentiment classes in all domains, even an uninformed guess should achieve an accuracy around 33%.

As SVM owes its success to the technique of kernelization to a great extent, we also trialed the performance of SVM on SEED after kernelization. The motivation behind kernelization is to increase the expressivity of machine learning models by mapping the original data to a feature space of higher dimension, and the key trick in its implementation is that instead of explicitly encoding these feature vectors, we design the scalar-valued dot product - the kernel - of two feature vectors instead, and construct the model in such ways that only these kernels are used in computation. The most common forms of kernel include linear kernel, polynomial kernel, Gaussian (radial basis function, or RBF) kernel, and Sigmoid kernel. We conduct experiments with them, and gain an increase of about 6.5% in average test accuracy, as can be seen in Figure 2.

V. Domain Shift

The idea behind domain shift based adaptation methods is to narrow the disparity between source and target domains by adjusting them to accommodate each other. This is achieved by either projecting the data into a common - or at least similar - space, or weighting the samples. In this section, we illustrate three methods based on these ideas - Subspace Alignment, Correlation Alignment, and Kernel Mean Matching.

A. Subspace Alignment

One of the most intuitive ideas in accommodating data from different domains is to project it into a common space where the discrepancy between domains are narrowed. *Subspace Alignment* (*SA*) [7], as its name suggests, achieves this goal by performing a straightforward *Principal Component Analysis* and aligning the principal subspace of source domain to the target domain. Let $\bar{\mathbf{P}}_s, \bar{\mathbf{P}}_t$ denote the matrix consisting of principal components of source domain and target domain respectively, and the data are projected by

$$\bar{\mathbf{X}}_s = \bar{\mathbf{P}}_t\bar{\mathbf{P}}_s^T\bar{\mathbf{P}}_s\mathbf{X}_s, \tag{2}$$

$$\bar{\mathbf{X}}_t = \bar{\mathbf{P}}_t\mathbf{X}_t, \tag{3}$$

and downstream classification tasks are then performed on the transformed data.

However, this is only applicable in scenarios where there is one source domain and one target domain. When there are multiple source domains, as is the case of SEED, one possible solution is to treat all the source subjects as one domain as does the baseline, but this results in poor performance, as the source data are not identically distributed. An alternative is to align all the subspace of fifteen domains in a chained fashion, but this leads to the magnification of errors in each step and results in low accuracy as well. To tackle this problem, we propose a simple, yet powerful *voting mechanism*: for each subject in the source data, we train a classifier that's adapted to the test subject, and take the majority vote of the fourteen classifiers as the final prediction. Without further specification, we will use this mechanism by default in the rest of this work.

After cross-validation, we choose to project the data into 64-dimensional subspace, and a performance increase of 7% over baseline is observed on the target domain, as demonstrated in Figure 3.

B. Correlation Alignment

Correlation Alignment (*CORAL*) [15] is another classical domain adaptation method achieved through matrix decomposition. Unlike SA, though, it aligns not the data's principal components, but second order information by performing *Singular Value Decomposition* on the covariance matrices of data from both domains. Specifically, let $\mathbf{C}_s, \mathbf{C}_t$ denote the covariance matrices of source and target domain, the objective of CORAL is

$$\min_{\mathbf{A}} ||\mathbf{A}^T\mathbf{C}_s\mathbf{A} - \mathbf{C}_t||_F^2, \tag{4}$$

where $\mathbf{A}$ is the projection matrix for source domain. Solving this optimization problem, we can get the optimal solution

$$\mathbf{A}^* = \mathbf{U}_s\mathbf{\Sigma}_s^{-1/2}\mathbf{U}_s^T\mathbf{U}_t[1:r]\mathbf{\Sigma}_t[1:r]^{1/2}\mathbf{U}_t[1:r]^T, \tag{5}$$

where $r = \min(rank(\mathbf{C}_s), rank(\mathbf{C}_t))$. The classifier, as in SA method, is then trained and evaluated on the transformed data $\mathbf{A}^{*T}\mathbf{X}_s$ and $\mathbf{X}_t$.

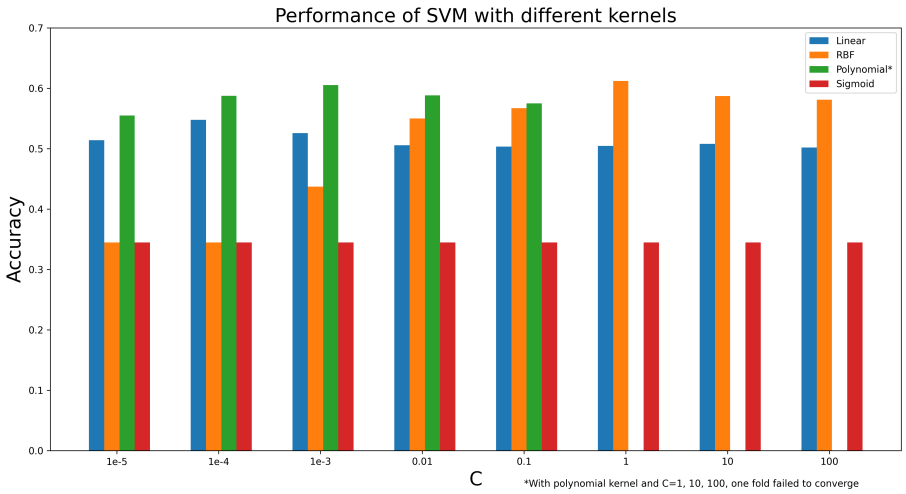

Fig. 2. Performance of baseline and kernelized support vector machines

As is shown in Figure 3, a dramatic boost of nearly 23% in target accuracy is obtained after aligning the source domain's covariance with that of the target domain.

C. Kernel Mean Matching

Apart from explicitly projecting data samples of source domain and target domain to a common feature space, another idea in domain shift based adaptation is to assign a different weight to each source sample during training stage, and learn these weights from the relation between source and target domains so that the learned classifier can better acclimate to target data. *Kernel Mean Matching* (*KMM*) [10] is an approach following this idea, where the weights of kernelized training samples are learned to minimize the *Maximum Mean Discrepancy* (*MMD*) between source domain and target domain:

$$\min_{\beta_i} ||\frac{1}{n_s}\sum_{i=1}^{n_s}\beta_i\Phi(\mathbf{x}_i^s) - \frac{1}{n_t}\sum_{i=1}^{n_t}\Phi(\mathbf{x}_i^t)||^2, \tag{6}$$

and the classifier is then trained on the weighted training samples.

For simplicity, we omit the application of kernel and directly use the existing EEG features, and the computation of weights is a quite standard optimization problem. We restrict all the weights to be no more than 1, but performs no normalization on them. The results of SVM classifier's performance trained on the weighted source data are comparable to those of CORAL, with an average target performance of 77.97% across all subjects. The prediction accuracies for each subject are also plotted in Figure 3.

VI. Parameter Transfer

Compared with domain shift methods where the voting mechanism is required to accommodate multiple source domains, parameter transfer is inherently more suitable for scenarios with multiple source domains.

Let $\mathbf{X}_i^s, i = 1, 2, \cdots, N$ and $\mathbf{X}^t$ denote data samples from the N source domains and the target domain. The key assumption of *Transductive Parameter Transfer* (*TPT*) [14] is that these samples are drawn from their respective distribution $\mathbf{X}_i^s \sim P_{\mathcal{X}_i^s}, \mathbf{X}^t \sim P_{\mathcal{X}^t}$, and that there exists a function mapping from these distributions to the parameter space of the classification models of each domain $f : P \to \Theta$. However, as the distribution of data can not be easily depicted, we simply substitute the probability distribution with the feature space: $\hat{f} : \mathcal{X} \to \Theta$, which can be easily learnt by any regression model.

TPT uses Support Vector Classification (SVC) to model the labeled data of each source domain, and Support Vector Regression (SVR) to model the SVC's parameters. However, as the parameters of SVC are intrinsically related to the data samples, we use neural network instead for the classification task to enable better generalization ability. Following this idea, we first evaluate the performance of neural networks trained on $\mathbf{X}^s$ and directly applied to $\mathbf{X}^t$ (which can be viewed as a naive form of domain generalization). We use a three-layer network with 128, 32 nodes in the two hidden layers respectively, and it achieves an average accuracy across all folds roughly on par with baseline SVM (Figure 4).

Inspired by [11] and [5], we also apply normalization to the data to reduce the difference between subjects, which is also a form of domain shift adaptation in our taxonomy. We experiment with two different normalization methods. The first is the usual normalization employed in training neural networks on independent and identically distributed data, i.e. subtracting the mean of training samples from both training and testing data, which brings a 7% increase in performance. The second is to subtract

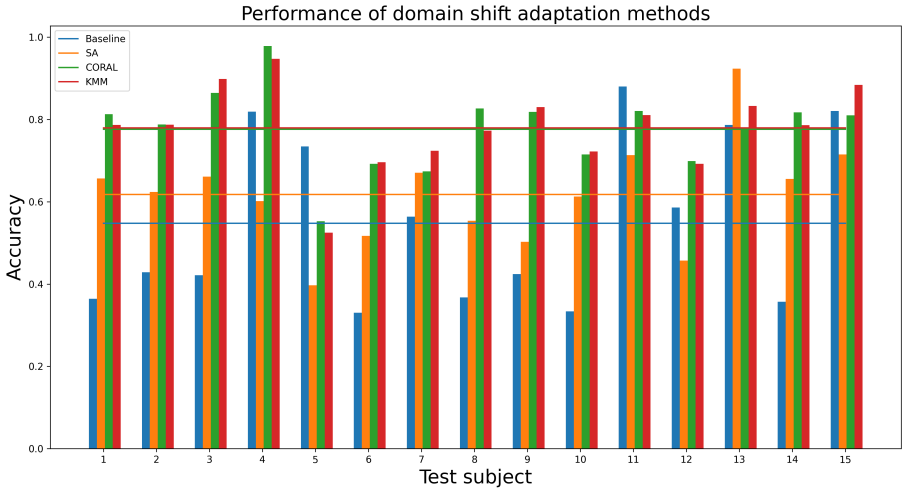

Fig. 3. Performance of domain shift based adaptation methods

from each subject's data its own mean so as to pull the fifteen domains closer together. In this case, the network's performance on test subjects surpasses all the previous methods and rises all the way to a staggering 79.11%, as plotted in Figure 4. In the rest of this work, we will refer to this naive yet powerful method as *Neural Network Domain Adaptation* (*NNDA*).

When conducting parameters transfer, our experiments show that when the model has too many parameters to be transferred, it suffers from high variance. Thus, to stabilize the learning process, we first perform dimensionality reduction on both source subjects and test subject by simply taking the 32-dimension hidden states obtained from NNDA, and then train a classification network on each source subject's reduced features. The network has two layers and 16 hidden units, with a total number of about six hundred parameters - a reasonable number considering that we have only fourteen training sample when transferring these parameters. They are then transferred by multi-output SVR, and the transferred networks achieve an average classification accuracy of 78.08% across all folds.

VII. Adversarial Learning

Unlike previous methods that work on fixed feature representations, adversarial learning combines domain adaptation and deep feature learning within one training process. It makes use of domain confusion loss to help the neural network learn domain-independent features, which are both discriminative and invariant to the change of domains. One of the most groundbreaking work on introducing adversarial learning is Domain-Adversarial Neural Network, and we will illustrate our improvement on it. We will also illustrate the generalization on it, which is Adversarial Discriminative Domain Adaptation.

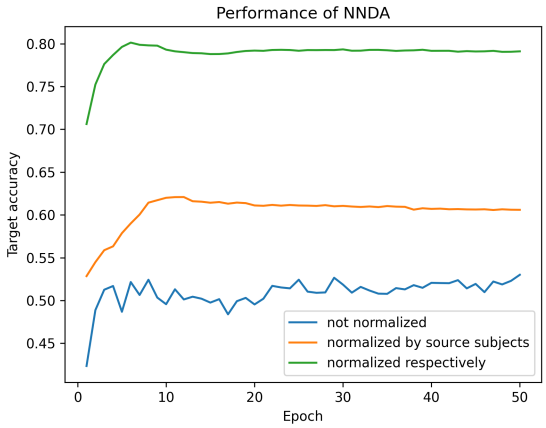

Fig. 4. Performance of NNDA with different normalization methods

A. Domain-Adversarial Neural Network

Domain-Adversarial Neural Network (*DANN*) [8] operates by jointly optimizing the underlying features as well as two discriminative classifiers operating on these features. First, a *feature extractor* G_f is used to transform the original feature into the underlying feature space. Then the *label predictor* G_y uses the extracted features to predict class labels, and the *domain classifier* G_d uses the extracted features to discriminate between the source and the target domains.

During the learning stage, DANN aims to minimize the label prediction loss on the annotated part (i.e. the source part) of the training set, and so the parameters θ_f and θ_y of both the feature extractor and the label predictor are thus optimized in order to minimize the empirical loss for the source domain samples, which ensures the discriminativeness. At the same time, to

make the features domain-invariant, the objective of the feature extractor should make the domain classifier unable to distinguish the source domain distribution and the target domain distribution, while the domain classifier is trained to discriminate the two distributions. And so it comes into adversary that the parameters θ_f of the feature extractor is optimized to maximize the domain classification loss while the parameters θ_d of the domain classifier is optimized to minimize the domain classification loss. Formally, consider the functional:

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1\dots N \\ d_i=0}} L_y\left(G_y\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_y\right), y_i\right) \\ & - \lambda \sum_{\substack{i=1\dots N \\ d_i=0,1}} L_d\left(G_d\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_d\right), d_i\right) \\ = & \sum_{\substack{i=1\dots N \\ d_i=0}} L_y^i\left(\theta_f, \theta_y\right) - \lambda \sum_{\substack{i=1\dots N \\ d_i=0,1}} L_d^i\left(\theta_f, \theta_d\right) \end{aligned} \tag{7}$$

Here, $L_y(\cdot,\cdot)$ and $L_d(\cdot,\cdot)$ are the losses for label prediction and domain classification respectively, and $d_i = 0$ denotes the i-th training example in source training set, while $d_i = 1$ denotes the i-th example in target set, and λ is a ratio that control the domain loss weight.

Then the parameters aim to seek form a saddle point of the functional (7):

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E\left(\theta_f, \theta_y, \hat{\theta}_d\right), \tag{8}$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E\left(\hat{\theta}_f, \hat{\theta}_y, \theta_d\right). \tag{9}$$

And these lead to the gradient descent updates, where μ denotes the learning rate:

$$\theta_f \longleftarrow \theta_f - \mu\left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f}\right), \tag{10}$$

$$\theta_y \longleftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y}, \tag{11}$$

$$\theta_d \longleftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}. \tag{12}$$

Such updates can be accomplished by applying a special *gradient reversal layer* (*GRL*) before passing the transformed features into the domain classifier. During the forward propagation, GRL acts as an identity transform. During the backward propagation though, GRL takes the gradient from the subsequent level, multiplies it by -λ and passes it to the preceding layer. Then the pseudo objective function can be defined as:

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1\dots N \\ d_i=0}} L_y\left(G_y\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_y\right), y_i\right) \\ & + \lambda \sum_{\substack{i=1\dots N \\ d_i=0,1}} L_d\left(G_d\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_d\right), d_i\right), \end{aligned} \tag{13}$$

such that normal gradient descent updates can be applied.

Based on this idea, we implement the DANN architecture on SEED dataset, where we design our feature extractor G_f as a one-layer network mapping from 310-dimensional original feature vector into 64 dimensional feature vector, label predictor G_l and domain classifier G_d also as one-layer logistic regression. During training, the ratio λ is set to 0 during the first five warm up epochs, and then set to 1. And we get the average result of 60.5% across all folds.

Because of the hardness of training adversary, we borrow the idea of domain shift to help the training for domain invariance. Specifically, we add a *MMD loss* L_{MMD} to decrease the discrepancy between the distributions of transformed source domain features and transformed target domain features as

$$L_{MMD} = ||\frac{1}{n_b}\sum_{i=1}^{n_b} G_f(\mathbf{x}_i^s;\theta_f) - \frac{1}{n_b}\sum_{i=1}^{n_b} G_f(\mathbf{x}_i^t;\theta_f)||^2. \tag{14}$$

That means we use one batch's features to estimate the whole domain features distribution, and enclose the discrepancy between source domain and target domain.

While MMD loss helps measure the first order statistics of domain invariance, we can also borrow a *CORAL loss* L_{CORAL} to decrease the second order discrepancy:

$$L_{CORAL} = ||G_f(\mathbf{X}_b^s;\theta_f)^T G_f(\mathbf{X}_b^s;\theta_f) - G_f(\mathbf{x}^t;\theta_f)^T G_f(\mathbf{X}^t;\theta_f)||_F^2. \tag{15}$$

The trade-off between DANN's loss and MMD loss, CORAL loss is controlled by two ratio factors λ_{MMD} and λ_{CORAL} which we empirically set to 0.3 and 1 respectively. The overall loss function of our network can then be expressed as

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1\dots N \\ d_i=0}} L_y\left(G_y\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_y\right), y_i\right) \\ & + \lambda \sum_{\substack{i=1\dots N \\ d_i=0,1}} L_d\left(G_d\left(G_f\left(\mathbf{x}_i;\theta_f\right);\theta_d\right), d_i\right) \\ & + \lambda_{MMD} \sum_{b\in\mathcal{B}} L_{MMD}(G_f(\mathbf{X}_b^s;\theta_f), G_f(\mathbf{X}_b^t;\theta_f)) \\ & + \lambda_{CORAL} \sum_{b\in\mathcal{B}} L_{CORAL}(G_f(\mathbf{X}_b^s;\theta_f), G_f(\mathbf{X}^t;\theta_f)), \end{aligned} \tag{16}$$

where $\mathcal{B}$ denotes the collection of training batches.

We train our network for 800 epochs, and it achieves an average accuracy of 78.68% across the 15 test folds. To analyze the contribution of each loss, we conduct ablation studies, as shown in Figure 5. With only the gradients from the label predictor being used for update, the network performance is even worse than baseline. The addition of domain adversarial loss brings a 10% performance boost, while MMD loss increases the accuracy by nearly 25%. Further applying DA loss and CORAL loss on top of MMD raises the accuracy by about 1% respectively. It is also found that normalization is vital for the invariance to

hold across domains, as it adds about 5% to the accuracy on target domain.

B. Adversarial Discriminative Domain Adaptation

Based on DANN, *Adversarial Discriminative Domain Adaptation* (*ADDA*) [17] extends the idea of domain-invariant feature extraction by using two feature extractors, one for source domain and the other for target domain. The training procedure is also divided into two stages, first training the source domain feature extractor to minimize the label prediction loss, than using the source domain feature extractor's parameters to initialize the target domain feature extractor and start the adversarial learning described in DANN.

However, as ADDA decouples the processes of training for class discrimination on source domain and adapting to target domain in pursuit of generalization, it also loses some power in capturing the disparity between source and target domains. For comparison, we designed our ADDA network to be the same as that used for DANN, and it achieves an average accuracy of about 72% on test subjects. Applying MMD loss to the network helps with adaptation performance as well, but only lifts the accuracy to 75.20%, still inferior to DANN.

VIII. Parallel Transfer Learning

With the exploding amount of training samples present in the era of big data, parallel machine learning has been quickly gaining attention from the research community. The voting mechanism we proposed in Section V is inherently suited to parallel learning, and all the experiments in this work are conducted with multiprocessing on a single machine with 16 CPU cores to speed up training (models involving neural networks are trained on an RTX 3090). In Figure 6 we give the comparison between the training time of some domain adaptation methods with and without parallel learning.

It can be observed that while parallel learning does increase the training speed, it does not achieve the efficiency that one would expect. For example, in the case of baseline SVM, we trained the fifteen classifiers of all folds at the same time, but the overall training speed is only increased by a factor of 2.7. We suspect that this is in part because the underlying libraries (in this case, the scikit-learn module) have builtin usages of multiprocessing. This is even more obvious in CORAL and KMM, where multiprocessing does not help with the training speed at all.

As for the neural network based methods, the main limitation we face is the computation power at our command. As we have only one GPU, sometimes training too many models at one time would slow down the training significantly, as the device spends too much time on scheduling processes and switching contexts. To tackle this problem, we manually estimated the computation overhead of each method, and specified the maximum number of parallel processes for them. Overall, their efficiency is increased by roughly two times.

Lastly, we also note that the voting mechanism we propose does not scale well with respect to the number of domains, especially in a time when the energy consumed by computation devices is increasingly gaining public awareness. We leave it for future works to deign more efficient, as well as scalable methods for domain adaptation.

TABLE I
Comparison of domain adaptation methods on SEED

	Method	Accuracy
no adaptation	SVM	54.78
	kernelized SVM	61.22
domain shift	SA	61.74
	CORAL	77.63
	KMM	77.97
	NNDA	79.11
parameter transfer	TPT	78.08
adversarial learning	ADDA	75.20
	improved DANN	78.68

IX. Conclusion

In this work, we experimented with three families of domain adaptation methods on SEED: domain shift, parameter transfer, and adversarial learning, the results of which are summarized in Table I. With domain shift methods, domain projection based CORAL and sample selection based KMM both reached about 78% prediction accuracy on the test set, while neural network based NNDA neared 80%. TPT proves to be a useful algorithm exploiting the idea of parameter transfer, but suffers from volatility when dealing with large models. Nonetheless, by applying dimensionality reduction on the dataset first it also gained a more than 22% performance increase over baseline. For adversarial learning approaches, we reproduced DANN and ADDA algorithms - both of which achieved more than 75% test accuracy - and applied MMD loss and CORAL loss to DANN, gaining a slight increase in performance. To deal with SEED's multiple source domains, we proposed a voting mechanism that divides the adaptation task into easier one-to-one adaptation subtasks and applied it to all the adaptation methods, and deployed parallel learning techniques as well to speed up training.

References

[1] Rahaf Aljundi, Rémi Emonet, Damien Muselet, and Marc Sebban. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 56–63, 2015.

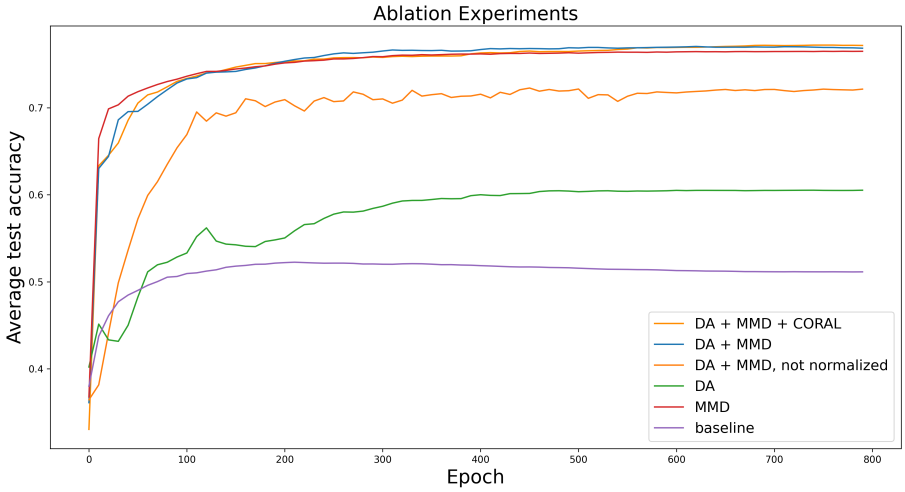

Fig. 5. Ablative comparison of deep learning based adaptation methods: Domain Adversary (DA), Maximum Mean Discrepancy (MMD), and CORrrelation ALignment (CORAL)

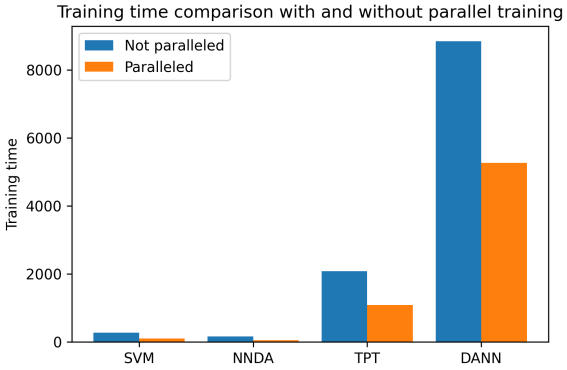

Fig. 6. Time efficiency brought by parallel learning

[2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In 2013 IEEE International Conference on Computer Vision (ICCV), pages 769–776, Los Alamitos, CA, USA, dec 2013. IEEE Computer Society.

[3] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, page 343–351, Red Hook, NY, USA, 2016. Curran Associates Inc.

[4] Lorenzo Bruzzone and Mattia Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(5):770–787, 2010.

[5] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 5077–5085, 2017.

[6] Wen-Sheng Chu, Fernando De la Torre, and Jeffery F. Cohn. Selective transfer machine for personalized facial action unit detection. In 2013 IEEE Conference on Computer Vision and Pattern Recognition, pages 3515–3522, 2013.

[7] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In 2013 IEEE International Conference on Computer Vision, pages 2960–2967, 2013.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, page 1180–1189. JMLR.org, 2015.

[9] Judy Hoffman, Eric Tzeng, Jeff Donahue, Yangqing Jia, Kate Saenko, and Trevor Darrell. One-shot adaptation of supervised deep convolutional models. 2013.

[10] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems, volume 19. MIT Press, 2006.

[11] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings. OpenReview.net, 2017.

[12] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, page 97–105. JMLR.org, 2015.

[13] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 22(2):199–210, 2011.

[14] Enver Sangineto, Gloria Zen, Elisa Ricci, and Nicu Sebe. We are not all equal: Personalizing models for facial expression analysis with transductive parameter transfer. In Proceedings of the 22nd ACM International Conference on Multimedia, MM '14, page 357–366, New York, NY, USA, 2014. Association for Computing Machinery.

[15] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, page 2058–2065. AAAI Press, 2016.

[16] Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation, pages 443–450. 11 2016.

[17] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages

2962–2971, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[18] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, page 4068–4076, USA, 2015. IEEE Computer Society.

[19] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008.

[20] Ying WEI, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 5085–5094. PMLR, 10–15 Jul 2018.

[21] Shujian Yu, Ammar Shaker, Francesco Alesiani, and Jose C Principe. Measuring the discrepancy between conditional distributions: Methods, properties and applications. arXiv preprint arXiv:2005.02196, 2020.

[22] Wei-Long Zheng and Bao-Liang Lu. Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks. IEEE Transactions on Autonomous Mental Development, 7(3):162–175, 2015.