ADAST: Attentive Cross-domain EEG-based Sleep Staging Framework with Iterative Self-Training

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Xiaoli Li and Cuntai Guan *Fellow, IEEE*

***Abstract*—Sleep staging is of great importance in the diagnosis and treatment of sleep disorders. Recently, numerous data-driven deep learning models have been proposed for automatic sleep staging. They mainly train the model on a large public labeled sleep dataset and test it on a smaller one with subjects of interest. However, they usually assume that the train and test data are drawn from the same distribution, which may not hold in real-world scenarios. Unsupervised domain adaption (UDA) has been recently developed to handle this domain shift problem. However, previous UDA methods applied for sleep staging have two main limitations. First, they rely on a totally shared model for the domain alignment, which may lose the domain-specific information during feature extraction. Second, they only align the source and target distributions globally without considering the class information in the target domain, which hinders the classification performance of the model while testing. In this work, we propose a novel adversarial learning framework called ADAST to tackle the domain shift problem in the unlabeled target domain. First, we develop an unshared attention mechanism to preserve the domain-specific features in both domains. Second, we design an iterative self-training strategy to improve the classification performance on the target domain via target domain pseudo labels. We also propose dual distinct classifiers to increase the robustness and quality of the pseudo labels. The experimental results on six cross-domain scenarios validate the efficacy of our proposed framework and its advantage over state-of-the-art UDA methods. The source code is available at https://github.com/emadeldeen24/ADAST.**

***Index Terms*—Unsupervised domain adaptation, adversarial training, self-training, cross-dataset sleep stage classification, EEG data.**

I. Introduction

Sleep stage classification is crucial to identifying sleep problems and disorders in humans. This task refers to the classification of one or many different signals including electroencephalography (EEG), electrocardiogram (ECG), electrooculogram (EOG), and electromyogram (EMG) into one of five sleep stages, namely, wake (W), rapid eye movement (REM), non-REM stage 1 (N1), non-REM stage 2 (N2), and non-REM stage 3 (N3). For EEG recordings, they are usually split into 30-second segments, where each segment is classified manually into one of the above stages by specialists [1]. Despite being mastered by many specialists, the manual annotation process is tedious and time-consuming, especially with the large amount of collected EEG data.

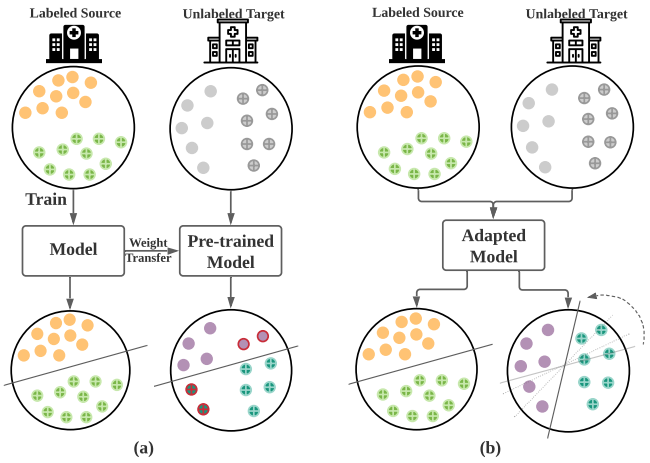


Fig. 1: (a) Direct Transfer (DT) fails due to the domain shift, and (b) Domain Adaptation (DA) solves the domain shift problem.

In recent years, numerous data-driven deep learning approaches have been developed, relying on the availability of a massive amount of labeled data for training. Therefore, many deep learning methods have been proposed to perform sleep staging automatically [2]–[5]. These methods implemented different network structures to process EEG data and trained proper classification models to achieve good performance while testing. Since these methods were able to achieve decent performance, it was expected to be a step forward to reduce the reliance on the manual scoring process. However, many sleep labs were found to keep relying on manually scoring EEG data [6], [7]. The main reason is the high variation between the public training data and the data generated in the sleep labs. These variations can occur due to several factors, *e.g.*, different measuring locations on the skull and different sampling rates of measuring devices. This is well-known as the *domain shift* problem, i.e., the training (*source*) and testing (*target*) data have different distributions. As shown in Fig. 1(a), directly

Emadeldeen Eldele, Chee-Keong Kwoh and Cuntai Guan are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (E-mail: {emad0002, asckkwoh, ctguan}@ntu.edu.sg).

Mohamed Ragab and Xiaoli Li are with Institute for Infocomm Research (I^2R), Centre for Frontier Research (CFAR), Agency of Science, Technology and Research (A*STAR), Singapore, and also with the School of Computer Science and Engineering at Nanyang Technological University, Singapore (E-mail: mohamedr002@e.ntu.edu.sg, xlli@i2r.a-star.edu.sg).

Zhenghua Chen is with the Institute for Infocomm Research (I^2R) and the Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore (E-mail: chen0832@e.ntu.edu.sg).

Min Wu is with the Institute for Infocomm Research (I^2R), Agency for Science, Technology and Research (A*STAR), Singapore (E-mail: wumin@i2r.a-star.edu.sg).

First and second authors are supported by A*STAR SINGA Scholarship.

Min Wu is the corresponding author.

applying the source-pretrained model (i.e., Direct Transfer) on the target data may not well-classify the target domain data due to the domain shift. Consequently, these models suffer significant performance degradation when trained on public datasets and tested on the sleep labs data. In addition, it is difficult for these labs to annotate large enough EEG datasets to re-train the models.

A typical solution for the above issues is to employ transfer learning approaches [7], [8]. For instance, Phan *et al.* [7] applied transfer learning from a large dataset to a different and relatively smaller one. They first pre-trained the model on a large dataset and then fine-tuned it on a smaller dataset. Similarly, the authors in [8] studied the channel mismatch problem while transferring the knowledge from one dataset to another. However, these transfer learning methods require the availability of labeled data from the target domain to fine-tune the model. In reality, the target domain may be completely unlabeled, and it is thus impractical to fine-tune the models.

Unsupervised Domain Adaptation (UDA) is a special scenario of transfer learning that aims to minimize the mismatch between the source and target distributions without using any target domain labels. As shown in Fig. 1(b), UDA aims to use both the labeled source domain along with the unlabeled target domain to train the model in a way that allows it to perform well on both source and target domains. So far, limited studies have investigated UDA in the context of sleep stage classification. For instance, Chambon *et al.* [9] improved the feature transferability between source and target domains using optimal transport domain adaptation. In addition, Nasiri *et al.* [6] used adversarial training-based domain adaptation to improve the transferability of features. However, these methods still suffer from the following limitations. First, they rely on totally shared models (i.e., same architectures with same weights) to extract features from both source and target domains. This may lose the domain-specific features for both source and target domains, which can be harmful to the classification task on the target domain. Second, these approaches only align the global distribution between source and target domains without considering the mismatch of the fine-grained class distribution between the domains. As such, target samples belonging to one class can be misaligned with an incorrect class in the source domain.

To tackle the aforementioned challenges, we propose an **A**dversarial **D**omain **A**daptation framework based on preserving attention mechanism and iterative **S**elf-**T**raining strategy (**ADAST**) for a single channel EEG-based sleep stage classification. Specifically, we first propose a domain-specific attention module to preserve both the source-specific and the target-specific features. This helps to keep the main characteristics of both domains to improve adversarial learning. Second, we propose an iterative self-training strategy to well-classify the fine-grained distribution of the unlabeled target domain using a target domain pseudo labels supervision. Hence, we can adapt the classification decision boundaries according to the target domain classes. Moreover, we design distinct dual classifiers to improve the robustness of target domain pseudo labels.

The main contributions of this work are summarized as follows:

- We propose a novel cross-dataset sleep staging framework that integrates iterative self-training with adversarial learning. Therefore, our framework can effectively classify the fine-grained distribution of the unlabeled target sleep data.
- ADAST utilizes an unshared domain-specific attention module to preserve the key features in both source and target domains during adaptation, which improves the adversarial training and boosts the classification performance on the target domain.
- We design distinct dual classifiers to improve the robustness of the generated pseudo labels in self-training. We also add a similarity constraint on their weights to push them from being identical.
- Extensive experiments demonstrate that our ADAST achieves superior performance for cross-domain sleep stage classification against state-of-the-art UDA methods.

II. Related Works

A. Sleep Stage Classification

Automatic sleep staging with single-channel EEG has been widely studied in the literature. In particular, deep learning-based methods [2], [4], [5] have shown great advances through end-to-end feature learning. These methods design different network structures to extract the features from EEG data and capture the temporal dependencies.

Several studies explored convolutional neural networks (CNN) for feature extraction from EEG data. For example, Supratak *et al.* [2] proposed two CNN branches to extract different frequency features in EEG signals. The same CNN architecture was also adopted by Mousavi *et al.* [10]. Li *et al.* [11] proposed to adopt CNN in addition to a squeeze and excitation block to extract the features from multi-epoch EEG data. Eldele *et al.* [5] developed a multi-resolution CNN with an adaptive features recalibration to extract representative features. Additionally, Qu *et al.* [12] proposed multiple residual CNN blocks to learn features mappings. The above methods further handled the temporal dependencies in EEG epochs. They either used recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) as in [2], [10], or adopted the multi-head self-attention approach as in [5], [12].

Different from relying on CNNs, researchers proposed different ways to handle EEG data. For example, Phan *et al.* [4] designed an end-to-end hierarchical RNN architecture. It consists of an attention-based recurrent layer to handle the short-term features within EEG epochs, besides a recurrent layer to capture the epoch-wise features. In addition, Phan *et al.* [13] used both raw EEG signal and its time-frequency image to design a joint multi-view learning from both representations. Also, Jia *et al.* [14] proposed a graph-based approach for sleep stage classification, where graph and temporal convolutions were utilized to extract spatial features and capture the transition rules, respectively. Finally, Neng *et al.* [15] handled the EEG data in three levels: frame, epoch, and sequence levels to extract a mixture of features that would improve the classification performance. Despite the success of these methods in handling complex EEG data,

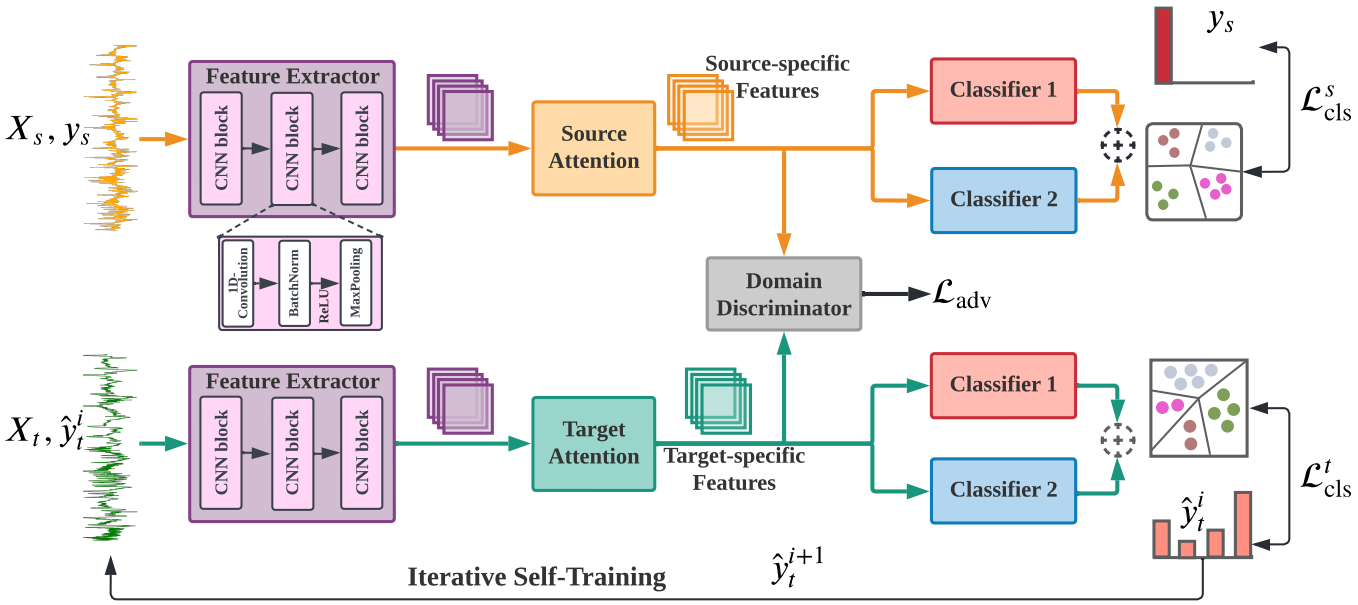


Fig. 2: The overall architecture of the proposed ADAST framework. The shared feature extractor consists of three convolutional blocks, where each block contains 1D-convolution, batch normalization, non-linear ReLU activation, and MaxPooling. The two classifiers share the same architecture, but we apply a similarity constraint on their weights to push them from being identical to each other (best viewed in colors, as blocks with similar colors represent shared components).

their performance for cross-domain (e.g., cross-dataset) sleep stage classification is limited due to the domain shift issue. Therefore, many researches were directed to adopt transfer learning approaches to handle this issue.

B. Transfer Learning for Sleep Staging

Some works studied the problem of personalized sleep staging to improve the classification accuracy for individual subjects within the same dataset using transfer learning [16], [17]. For a dataset with two-night recordings for each subject, these works pretrained the model by excluding the two nights of the test subject. Next, the first night is applied for fine-tuning the model and the second night is used for evaluation. However, few works have been proposed for the cross-dataset scenario, i.e., training a model on subjects from one dataset and testing on different subjects from another dataset. Phan *et al.* [7] studied the data-variability issue with the availability of large source dataset, and different labeled but insufficient target dataset. They trained their model on the source dataset and fine-tuned it on the smaller target dataset. With a similar problem setting, Phan *et al.* [8] proposed to use deep transfer learning to overcome the problem of channel mismatch between the two domains.

These methods require large corpus source datasets to increase their generalization ability and a labeled target dataset to fine-tune their models. Unsupervised domain adaptation (UDA) approaches were proposed to address these issues by aligning the features from different domains. These approaches can be categorized as discrepancy-based approaches and adversarial-based approaches. The discrepancy-based approaches such as Maximum Mean Discrepancy (MMD) [18] and CORrelation ALignment (CORAL) [19], attempt to minimize the distance metric between the source and target distributions. One the other hand, adversarial-based approaches mimic the adversarial training proposed in the generative adversarial network (GAN) [20]. This approach is more popular in previous sleep staging works. For example, Zhao *et al.* [21] proposed using adversarial UDA with a domain discriminator and multiple classifiers fed from the different feature extractor layers. Nasiri *et al.* [6] used adversarial training along with local and global attention mechanisms to extract the transferable individual information. Yoo *et al.* [22] proposed using adversarial domain adaptation with three discriminators; one for global alignment and two for stage and subject discrimination.

Differently, we enhance the adversarial training process by preserving the domain-specific features through domain-specific attention. Besides, we consider the fine-grained domain classes with the iterative self-training strategy, which deploys the target pseudo labels to improve the classification performance in the unseen target domain.

III. Method

A. Preliminaries

In this work, we focus on the problem of unsupervised cross-domain adaptation for EEG-based sleep staging. In this setting, we have access to a labeled source dataset $X_s = \{(\mathbf{x}^i_s, y^i_s)\}_{i=1}^{n_s}$ of n_s labeled samples, and an unlabeled target dataset $X_t = \{(\mathbf{x}^j_t)\}_{j=1}^{n_t}$ of n_t samples. The source and target domains are sampled from source distribution $P_s(X_s)$ and target distribution $P_t(X_t)$ respectively, such that these distributions are different (i.e., $P_s \neq P_t$). Both domains share

the same label space $Y = \{1, 2, \dots K\}$, where K is the number of classes (i.e., sleep stages). The domain adaptation scenario aims to transfer the knowledge from a labeled source domain to a domain-shifted unlabeled target domain. In the context of EEG data, both $\mathbf{x}_s^i$ and $\mathbf{x}_t^i \in \mathbb{R}^{1\times T}$, where the number of electrodes/channels is 1 since we use single-channel EEG data, and T represents the number of timesteps in the 30-second EEG epochs.

B. Overview

As shown in Fig. 2, our proposed framework consists of three main components, namely domain-specific attention, adversarial training, and dual classifier-based iterative self-training. First, domain-specific attention plays an important role in refining the extracted features so that each domain preserves its key features. Second, the adversarial training step leverages a domain discriminator to align the source and target features. Particularly, the domain discriminator network is trained to distinguish between the source and target features while the feature extractor is trained to confuse the domain discriminator by generating domain invariant features. Finally, the iterative self-training strategy utilizes the target domain pseudo labels to adapt the classification decision boundaries according to the target domain classes. The dual classifiers are incorporated to improve the quality and robustness of the pseudo labels. Further details about each component will be provided in the following subsections.

C. Domain-specific Attention

Our proposed framework extracts domain invariant features by using a shared CNN-based feature extractor, i.e., $F_s(\cdot) = F_t(\cdot) = F(\cdot)$. Unlike the totally unshared architectures that require an extra pretraining step and are usually harder to converge, the shared feature extractor allows end-to-end training, besides being easier to converge. Therefore, most UDA algorithms adopted this shared design [23]. However, relying solely on this shared architecture may not be able to preserve the key features of each domain [24], [25]. The reason is that the high-dimensional features may contain information that distinguishes source and target domains, and are relevant for predicting the label [26]. Throughout the training, the model tries to remove these features to reduce the difference between source and target distributions and make the features domain-invariant, but this also affects the classification performance on each separate domain. As the classification mainly relies on the available source domain labels, the learned classifier will be more biased toward the source domain [27]. Hence, we propose an unshared attention module to learn both domain-invariant and domain-specific features jointly in our proposed framework.

For each position in the feature space, the attention module calculates the weighted sum of the features at all positions with a little computational cost. Thus, the features at each location have fine details that are coordinated with fine details in distant portions of the features. Formally, given an input source sample $\mathbf{x}_s \in \mathbb{R}^{1\times T}$ that is passed through the feature extractor to generate the source features, i.e., $F(\mathbf{x}_s) = (\mathbf{f}_{s1}, \dots, \mathbf{f}_{sl}) \in \mathbb{R}^{d\times l}$, where d is the number of CNN channels, and l is the length of the features. Inspired by [28], we deploy a convolutional attention mechanism as shown in Fig. 3. The attention operation starts by obtaining new representation for the features at each position by using two 1D-convolutions, i.e., H_1 and H_2. Specifically, given $\mathbf{f}_{si}, \mathbf{f}_{sj} \in \mathbb{R}^d$, which are the feature values at the positions i and j, they are transformed into $\mathcal{Z}_{si} = H_1(\mathbf{f}_{si})$ and $\mathcal{Z}_{sj} = H_2(\mathbf{f}_{sj})$. The attention scores are calculated as follows.

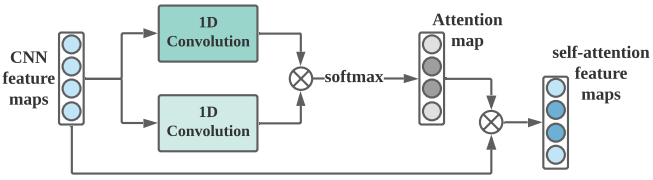


Fig. 3: Design of domain-specific attention module.

$$\mathcal{V}_{ji} = \frac{\exp(\mathcal{Z}_{si}^\top \mathcal{Z}_{sj})}{\sum_{k=1}^{l} \exp(\mathcal{Z}_{sk}^\top \mathcal{Z}_{sj})}. \tag{1}$$

Here, the attention score $\mathcal{V}_{ji}$ indicates the extent to which j^{th} position attends to the i^{th} position in the feature map. The output of the attention layer is $\mathcal{O}_s = (\mathbf{o}_{s1}, \dots, \mathbf{o}_{sj}, \dots \mathbf{o}_{sl}) \in \mathbb{R}^{d\times l}$, where

$$\mathbf{o}_{sj} = \sum_{i=1}^{l} \mathcal{V}_{ji} \mathbf{f}_{si}. \tag{2}$$

We denote the attention process in Equations 1 and 2 as $A(\cdot)$, such that $\mathcal{O}_s = A_s(F(\mathbf{x}_s))$. The same process applies to the target domain data flow to train A_t.

D. Adversarial Training

Given the learned source and target representations that preserve the domain-specific features, adversarial training is employed to align the source and target domains. Inspired by the generative adversarial network (GAN) [20], we aim to solve a minimax objective between the feature extractor and domain discriminator. Specifically, the domain discriminator is trained to classify between the source and target features, while the feature extractor tries to generate indistinguishable representations for both source and target domains. By doing so, the classifier trained on the source domain can generalize well on the target domain. However, with the minimax objective, the discriminator can saturate quickly, resulting in a gradient vanishing problem [29]. To address this issue, we train our model using a standard GAN loss with inverted labels [20]. Formally, the domain discriminator, D, classifies the input features to be either from the source or target domain. Thus, D can be optimized using a standard cross-entropy loss with the labels indicating the domain of the data point. The objective of this operation $\mathcal{L}_D$ can be defined as:

$$\begin{aligned}\min_D \mathcal{L}_{\mathrm{D}} = &- \mathbb{E}_{\mathbf{x}_s\sim P_s}[\log D(A_s(F(\mathbf{x}_s)))] \\ &- \mathbb{E}_{\mathbf{x}_t\sim P_t}[\log(1 - D(A_t(F(\mathbf{x}_t))))],\end{aligned} \tag{3}$$

where $\mathcal{L}_{\mathrm{D}}$ is used to optimize the domain discriminator separately so that it discriminates the source and target features. On the other hand, the feature extractor and the domain-specific attention are trained to confuse the discriminator by mapping the target features to be similar to the source ones. The objective function can be described as:

$$\min_{F,A_s,A_t} \mathcal{L}_{\mathrm{adv}} = -\mathbb{E}_{\mathbf{x}_s\sim P_s}[\log(1 - D(A_s(F(\mathbf{x}_s))))] - \mathbb{E}_{\mathbf{x}_t\sim P_t}[\log D(A_t(F(\mathbf{x}_t)))]. \quad (4)$$

Notably, only $\mathcal{L}_{\mathrm{adv}}$, which optimizes the feature extractor and the domain-specific attentions, is added to the overall objective function to ensure that the model is able to generate domain-invariant features.

E. Dual Classifier based Iterative Self-Training

With adversarial training, the distributions of source and target domains become globally aligned. However, the global alignment does not guarantee a good classification performance on the target domain, because of the difference in classification boundaries among source and target domains. Therefore, we propose a novel iterative self-training strategy to adjust the classification boundaries to fit the target domain and improve its classification performance.

Self-training converts the target domain predictions into pseudo labels and uses them to minimize the cross-entropy loss [30]. Given high-quality pseudo labels, they are treated as supervisory signals to adapt the decision boundaries of the classifier according to target domain classes. However, due to the domain shift, finding high-quality target domain pseudo labels can be a challenging problem, and the generated ones might be noisy and inefficient, especially at the beginning of the training. Nevertheless, we aim to *first* minimize the number of incorrect pseudo labels and *second* minimize the negative impact of these incorrect ones on the performance. To do so, we follow two main strategies.

First, we repeat the training of the model for r iterations, where the pseudo labels generated in the previous iteration are used in the next one. Since the model will be very uncertain about the pseudo labels in the first iteration, we ignore the target classification loss at this iteration. However, in the following iterations, we take it into consideration since the model becomes more confident about the pseudo labels after being trained to minimize the domain shift between source and target domains. Second, we use dual classifiers C_1 and C_2 setup, which has two main benefits. First, it helps the model to avoid the variance in the training data. Second, the average prediction vector of the two classifiers decreases the probability of low-confident predictions.

Notably, we design the two classifiers such that they share the same architecture, i.e., a single fully connected layer. This helps limiting the total number of parameters in the model and avoid pruning to overfitting. Consequently, we need to ensure that their predictions are diversified and they do not converge to become the one classifier throughout training. Thus, we add a regularization term $|\theta_{C_1}^{\intercal}\theta_{C_2}|$ on the weights of the two classifiers as inspired by [31], where θ_{C_1}, θ_{C_2} represent the weights of C_1 and C_2 respectively. This regularization term ensures the diversity of the two classifiers and helps them to produce different yet correct predictions. The final prediction vector is the averaged vector of the predictions of both classifiers.

Formally, in each iteration, we first calculate the average probability $\mathbf{p}_t$ of the two classifiers, and the corresponding target pseudo labels $\hat{y}_t$ as follows.

$$\mathbf{p}_t = \frac{1}{2}\left[C_1(A_t(F(\mathbf{x}_t)) + C_2(A_t(F(\mathbf{x}_t))\right], \quad (5)$$

$$\hat{y}_t = argmax(\mathbf{p}_t). \quad (6)$$

The target classification loss $\mathcal{L}_{\mathrm{cls}}^t$ based on the above pseudo labels is defined as follows.

$$\min_{F,A_t,C_1,C_2} \mathcal{L}_{\mathrm{cls}}^t = -\mathbb{E}_{\mathbf{x}_t\sim P_t} \sum_{k=1}^{K} \mathbb{1}_{[\hat{y}_t=k]} \log \mathbf{p}_t^k, \quad (7)$$

where $\mathbb{1}$ is the indicator function, which is set to be 1 when the condition is met, and set to 0 otherwise. The target classification loss $\mathcal{L}_{\mathrm{cls}}^t$ optimizes the feature extractor F, the target domain-specific attention A_t as well as the dual classifiers C_1 and C_2.

Similarly, the source classification loss $\mathcal{L}_{\mathrm{cls}}^s$, which depends on the source labels y_s, is formalized as follows.

$$\mathbf{p}_s = \frac{1}{2}\left[C_1(A_s(F(\mathbf{x}_s)) + C_2(A_s(F(\mathbf{x}_s))\right], \quad (8)$$

$$\min_{F,A_s,C_1,C_2} \mathcal{L}_{\mathrm{cls}}^s = -\mathbb{E}_{(\mathbf{x}_s,y_s)\sim P_s} \sum_{k=1}^{K} \mathbb{1}_{[y_s=k]} \log \mathbf{p}_s^k, \quad (9)$$

where the source classification loss $\mathcal{L}_{\mathrm{cls}}^s$ optimizes the feature extractor F, the source domain-specific attention A_s as well as the dual classifiers C_1 and C_2.

To sum up, we integrate the adversarial loss with the source and target classification losses and the regularization of the dual classifiers in one objective loss function as follows.

$$\mathcal{L}_{\mathrm{overall}} = \mathcal{L}_{\mathrm{adv}} + \mathcal{L}_{\mathrm{cls}}^s + \lambda_1 \mathcal{L}_{\mathrm{cls}}^t + \lambda_2 |\theta_{C_1}^{\intercal}\theta_{C_2}|. \quad (10)$$

Since the adversarial training and the source classification are two essential modules, we set their weights to one and tune the values of the two hyperparameters λ_1 and λ_2 to control their contributions. In overall, the three losses are integrated to guide the feature extractor to generate domain-invariant features, while allowing the domain-specific attentions to preserve the key features for each domain. Additionally, the dual classifiers are diversified using the regularization term. More details about the training can be found in Algorithm 1.

IV. Experiments

A. Datasets

We evaluate the proposed framework on three challenging datasets, namely Sleep-EDF[1] (**EDF** for short), SHHS-1 (**S1**) and SHHS-2[2] (**S2**). These three datasets represent distinct domains due to their differences in sampling rates and EEG channels.

[1]https://physionet.org/physiobank/database/sleep-edfx/

[2]https://sleepdata.org/datasets/shhs

Algorithm 1: Training procedure of our proposed ADAST framework.

Input: X_s, X_t, D, F, ψ_s, ψ_t, $\mathcal{C}_1$, $\mathcal{C}_2$;
Output: Trained F, ψ_t for the target domain;
1. Set $\lambda_1 = 0$;
2. **for** *i=1 to r* **do**
   3. Sample mini-batches from source domain $(\mathbf{x}_s, y_s) \sim P_s$ and target domain $\mathbf{x}_t \sim P_t$;
   4. Extract shared features using F, then unshared representations $\psi_s(F(\mathbf{x}_s))$, $\psi_t(F(\mathbf{x}_t))$;
   5. Update D by $\mathcal{L}_D$ (Eq. 3);
   6. Update F, ψ_s, ψ_t by $\mathcal{L}_{adv}$ (Eq. 4);
   7. Update $F, \psi_s, \mathcal{C}_1, \mathcal{C}_2$ by $\mathcal{L}^s_{cls}$ (Eq. 9);
   8. Penalize $\mathcal{C}_1$ and $\mathcal{C}_2$ weights similarity ;
   9. Generate pseudo labels $\hat{y}^i_t$ (Eq. 5,6);
   10. Update $F, \psi_t, \mathcal{C}_1, \mathcal{C}_2$ by $\mathcal{L}^t_{cls}$ (Eq. 7);
   11. Set a value to λ_1;
end

1) Sleep-EDF (EDF): EDF dataset was first published in 2013 with the polysomnography (PSG) readings of 20 healthy subjects (10 males and 10 females). It contains data from two studies: Sleep Cassette (SC* files) and Sleep Telemetry (ST* files). The Sleep Cassette study (1987-1991) addresses the age effects on sleep, while the Sleep Telemetry study (1994) addressed the temazepam effects on sleep. Each PSG recording consists of two EEG channels namely Fpz-Cz and Pz-Oz, with a sampling rate of 100 Hz. We adopted the EEG recordings from Fpz-Cz channel following previous studies [2], [4], [5].

2) SHHS-1 (S1): SHHS [32], [33] is a multi-center cohort study conducted to assess the cardiovascular and other consequences of sleep-disordered breathing. It tests the effect of some diseases such as stroke and hypertension on sleep-related breathing. The data of S1 was recorded for a total of 6,441 men and women aged 40 years and older in their first visit, which was by the end of 1995 and lasted for two years.

3) SHHS-2 (S2): The data of S2 represents the polysomnogram recordings of the second visit of 3,295 of the participants in S1. The outcome data was used to adjust the parent cohort.

Each PSG file in both S1 and S2 datasets contains data from two EEG channels namely C4-A1 and C3-A2, where we only adopt C4-A1 channel recordings for both datasets. We selected subjects from S1 and S2 datasets such that 1) they contain different patients, 2) subjects from S2 dataset have a sampling rate of 250 Hz, and 3) the subjects have Apnea Hypopnea Index (AHI) < 1 to eliminate the bias to sleep disorders and ensure a consistent clinical status of subjects [34]. Notably, we down-sampled the data from S1 and S2 datasets such that the sequence length is the same as the EDF dataset, i.e., 30 seconds $\times$ 100 Hz ($T = 3,000$).

We preprocessed the three datasets by 1) merging stages N3 and N4 into one stage (N3) according to AASM standard, and 2) including only 30 minutes of wake stage periods before and after the sleep [2]. Table I shows a brief summary of the above three datasets before down-sampling describing the number of subjects (#Subjects) in each cross-domain, the selected EEG channel, the sampling rate, and the number of training (#Train), validation (#Val), and testing (#Test) samples in each domain.

TABLE I: A brief description about the datasets.

Dataset	#Subjects	EEG Channel	Sampling Rate	#Train	#Val	#Test
EDF	20	Fpz-Cz	100	25,612	7,786	8,910
S1	42	C4-A1	125	24,515	7,948	9,067
S2	44	C4-A1	250	31,613	9,769	12,413

B. Feature Extractor

To extract the features from the EEG signals, we first preprocess the EEG signals to be split into 30-second segments (i.e., epochs). Each epoch is then passed through our feature extractor network to extract the features. We followed the design of the feature extractor proposed in [35] which consists of three blocks, such that each block consists of a 1D-convolution layer, a batch normalization layer, a non-linear ReLU activation and a MaxPooling layer, as shown in Fig. 2. The 1D-convolution layer in the first block has 32 filters, with a kernel size of 25 and a stride of 6. The 1D-convolution layer in the second and third layers have 64 and 128 filters respectively and both have a kernel size of 8 and a stride of 1. The features extracted from these three blocks are then sent to the self-attention mechanism.

C. Experimental Settings

To evaluate the performance of our model and baseline models, we employed the classification accuracy (ACC) and the macro-averaged F1-score (MF1). These two metrics are defined as follows:

$$ACC = \frac{\sum_{i=1}^{K} TP_i}{M}, \tag{11}$$

$$MF1 = \frac{1}{K}\sum_{i=1}^{K} \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i}, \tag{12}$$

where $Precision_i = \frac{TP_i}{TP_i+FP_i}$, and $Recall_i = \frac{TP_i}{TP_i+FN_i}$. TP_i, FP_i, TN_i, and FN_i denote the True Positives, False Positives, True Negatives, and False Negatives for the i-th class respectively, M is the total number of samples and K is the number of classes. All the experiments were repeated 5 times with different random seeds for model initialization, and then we reported the average performance (i.e., ACC and MF1) with standard deviation.

We performed *subject-wise* splits for the data from the three domains, i.e., we split them into 60%, 20%, and 20% for training, validation and testing, respectively, such that the data from one subject were assigned to either of the 3 splits. We used the training part of source and target domains while training our model. We used the validation part and test part of the target domain for validation and testing. Following [31], [36], we used the validation split of the target domain to select the best hyperparameters in our model. We tuned the parameters λ_1, λ_2 in the range $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1\}$, and set their values as $\lambda_1 = 0.01$ and $\lambda_2 = 0.001$. For

TABLE II: Comparison against various baselines. Best results are in bold, and the second best are underlined.

		Cross-Domain Accuracy						AVG
	Baselines	EDF→S1	EDF→S2	S1→EDF	S1→S2	S2→EDF	S2→S1	ACC
DT	DeepSleepNet [2]	49.19±3.09	48.48±2.74	76.37±0.11	61.53±0.82	62.45±0.26	**78.97±0.11**	62.83
	SleepEEGNet [10]	62.51±3.31	49.82±3.22	56.21±0.46	59.41±2.70	62.98±1.34	69.96±2.00	60.14
	AttnSleep [5]	64.44±3.37	57.43±5.45	75.41±0.65	72.08±0.23	66.59±0.83	77.52±0.29	68.91
	Source-Only *(Ours)*	57.76±2.40	52.05±4.47	75.05±1.43	63.84±3.83	59.65±2.90	73.94±1.48	63.72
DA	Deep CORAL [37]	63.92±2.35	53.24±4.19	75.95±4.53	61.49±2.43	68.90±3.69	75.55±1.64	66.51
	MDDA [38]	66.18±0.73	59.22±4.34	74.97±1.39	65.96±4.57	69.93±2.42	75.18±2.43	68.57
	DSAN [39]	65.45±0.61	**69.58±1.62**	**82.30±0.31**	67.19±2.11	70.89±1.01	75.80±1.62	71.87
	DANN [40]	65.93±0.63	58.67±3.44	77.40±0.54	64.14±0.48	67.53±2.60	73.72±0.87	67.90
	ADDA [29]	68.02±1.07	52.89±6.64	80.73±1.50	57.85±0.72	72.65±0.28	71.63±1.10	67.30
	CDAN [41]	64.15±2.41	64.09±1.27	78.02±0.84	66.06±2.29	72.13±2.13	76.42±1.88	70.14
	DIRT-T [36]	66.51±2.99	59.20±4.34	79.98±0.35	65.06±3.98	72.95±3.27	77.16±0.61	70.19
	ADAST *(Ours)*	**75.50±1.03**	67.56±2.37	75.94±1.25	**72.27±1.06**	**75.28±1.78**	77.80±0.25	**74.00**
		Cross-Domain F1-score						MF1
DT	DeepSleepNet [2]	42.43±2.85	39.93±2.37	64.58±0.44	55.02±0.61	53.22±0.22	**66.65±0.65**	53.63
	SleepEEGNet [10]	55.29±3.09	40.68±4.63	51.97±0.77	55.10±1.10	54.74±1.42	61.04±2.98	53.13
	AttnSleep [5]	54.77±2.36	50.06±3.29	63.03±0.66	**62.02±0.69**	56.41±0.82	63.71±0.12	58.33
	Source-Only *(Ours)*	46.04±1.86	42.22±4.07	63.07±1.34	54.96±3.12	49.10±2.55	62.71±3.00	53.02
DA	Deep CORAL [37]	55.94±2.21	42.75±5.82	62.36±3.99	50.68±2.34	57.35±2.81	61.84±1.41	55.16
	MDDA [38]	56.00±0.53	48.07±1.88	62.18±1.01	54.09±2.83	57.43±2.67	60.94±1.88	56.45
	DSAN [39]	55.67±0.43	**55.07±1.19**	67.78±0.28	55.55±1.16	58.79±1.04	62.20±0.93	59.18
	DANN [40]	54.79±0.54	48.49±2.57	63.86±0.69	53.48±0.36	57.14±2.08	60.17±0.65	56.32
	ADDA [29]	58.18±1.41	43.96±6.26	**68.43±0.88**	48.53±0.64	59.02±0.54	58.54±0.78	56.11
	CDAN [41]	52.61±2.31	52.42±0.33	64.31±0.93	54.73±1.45	59.70±2.18	62.76±1.79	57.75
	DIRT-T [36]	55.34±2.81	48.31±3.81	66.12±0.45	54.55±2.85	57.72±4.61	62.05±0.72	57.35
	ADAST *(Ours)*	**61.92±0.83**	53.80±2.11	63.33±1.02	58.69±0.60	**62.49±1.04**	63.10±0.06	**60.39**

iterative self-training, we set the maximum iterations r to 2, as the performance of the model was found to converge. We used Adam optimizer with a learning rate of 1e-3 that is decayed by 0.1 after 10 epochs, weight decay of 3e-4, $\beta_1 = 0.5$, $\beta_2 = 0.99$, and a batch size of 128. We trained the model for a predetermined number of epochs (15 epochs in our case) per iteration. All the experiments were performed with PyTorch 1.7 on a NVIDIA GeForce RTX 2080 Ti GPU. The source code and supplementary material are available at https://github.com/emadeldeen24/ADAST.

D. Baselines

To assess our proposed ADAST model, we compared it against various baselines. We first included the Direct Transfer (DT) results of three sleep staging methods. These methods are **DeepSleepNet** [2]; **SleepEEGNet** [10]; and **AttnSleep** [5] (refer to Section II-A for their details). In addition, we adopted seven state-of-the-art discrepancy- and adversarial-based domain adaptation (DA) baselines. In particular, Deep CORAL, MDDA and DSAN are discrepancy-based methods, while DANN, ADDA, CDAN, and DIRT-T are adversarial-based methods. These baselines are summarized as follows.

- **Deep CORAL** [37]: it extends CORAL [19] to learn a nonlinear transformation that aligns the correlations of layer activations in deep neural networks.
- **MDDA** [38]: it applies MMD and CORAL on multiple classification layers to minimize the discrepancy between the source and target domains.
- **DSAN** [39]: it incorporates a local MMD loss to align the same-class sub-domain distributions.
- **DANN** [40]: it jointly trains feature extractor and domain classifier by negating the gradient from the domain classifier with a gradient reversal layer (GRL).
- **ADDA** [29]: it performs a similar operation as DANN but by inverting the labels instead of using GRL.
- **CDAN** [41]: it minimizes the cross-covariance between feature representations and classifier predictions.
- **DIRT-T** [36]: it combines virtual adversarial domain adaptation with a teacher model to refine the decision boundary of the target domain.

Notably, we included the results of the **Source-Only** experiment, which refers to the DT results of our backbone network. In addition, we used our backbone feature extractor for all the seven DA baselines to ensure a fair comparison. We tuned the hyperparameters of the baselines to achieve their best performance.

E. Experimental Results

Table II shows the comparison results among various methods. Overall, the direct transfer usually achieves lower performance than domain adaptation. The results of DT experiments on [2], [5], [10] indicate that the domain shift problem causes a big performance drop and should be addressed separately. Therefore, it becomes important to use domain adaptation to address the domain shift problem for cross-dataset sleep stage classification, which is supported by the results of the other seven DA baselines.

It should be highlighted that the performance increase in the seven DA baselines should be compared to our Source-Only DT results, as they share the same backbone network.

TABLE III: Ablation study showing the different variants of our proposed ADAST framework. **ATT**: domain-specific ATTention, **DC**: Dual Classifiers, **ST**: Self Training. The first row indicates using only the domain discriminator along with a single classifier.

Component			Cross-Domain Accuracy						AVG
ATT	DC	ST	EDF→S1	EDF→S2	S1→EDF	S1→S2	S2→EDF	S2→S1	ACC
-	-	-	66.46±1.86	62.72±4.14	**79.10±1.23**	61.54±0.08	72.28±3.33	73.19±0.64	69.21
✓	-	-	71.31±1.78	**69.07±1.50**	76.83±1.83	63.98±3.11	74.59±0.93	75.19±1.63	71.50
✓	✓	-	72.53±0.88	66.50±2.45	78.22±0.48	68.54±5.29	**75.57±0.86**	76.98±0.54	73.05
✓	-	✓	70.76±2.22	68.79±0.20	77.61±0.79	68.58±5.14	74.05±1.73	75.82±0.67	72.60
✓	✓	✓	**75.50±1.03**	67.56±2.37	75.54±1.25	**72.27±1.06**	75.28±1.78	**77.80±0.25**	**74.00**

We noticed that the three methods considering the class-conditional distribution, i.e., CDAN, DIRT-T, and DSAN, outperform the ones globally aligning the source and target domains, i.e., DANN, Deep CORAL, ADDA, and MDDA. This indicates that considering class distribution, especially in the case of imbalanced sleep data, is important to achieve better classification performance on the target domain. Our proposed ADAST achieves superior performance over all the baselines in terms of both mean accuracy and F1-score in four out of six cross-domain scenarios for two reasons. First, our ADAST, similar to CDAN, DIRT-T, and DSAN, also considers the class-conditional distribution. In particular, ADAST explores the target domain classes using the proposed iterative self-training strategy with dual classifiers. Second, ADAST preserves domain-specific features using the unshared attention module, which improves the performance.

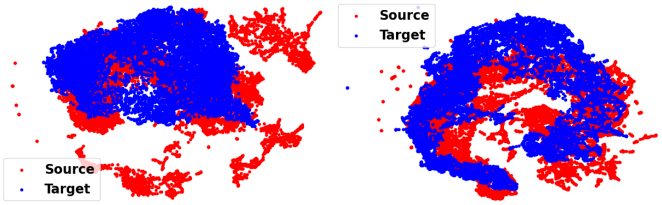
As shown in Table II, the performance of our model is less than most baselines in the scenario S1→EDF. Note that we used the same value of λ_1 (i.e., 0.01) for all the six scenarios, which might not be fair for some scenarios. We found that the quality of the pseudo labels is not good in this scenario S1→EDF, and thus we should use a smaller λ_1 to reduce the contribution of the target classification loss. By tuning λ_1 from 0.01 to 10^{-6}, the mean accuracy and MF1 of our ADAST in the scenario S1→EDF would increase from 75.94% and 63.33% to 78.50% and 64.73%, respectively. Please refer to Fig. S.1c in the supplementary material for more details.

We also observed interesting results while investigating different cross-dataset scenarios. Various methods usually achieve better performance in the cross-domain scenario S1→S2 than EDF→S2 (and similarly S2→S1 is better than EDF→S1). To explain this, as shown in Table I, S1 and S2 are closer to each other, as they have the same EEG channel. Meanwhile, EDF has a different EEG channel and sampling rate, and thus it is a distant domain from S1 and S2. These results indicate that distant domain adaptation is still very challenging. Finally, we observed that S1→EDF is easier than S2→EDF, probably because S1 and EDF have close sampling rates to each other.

F. Ablation Study

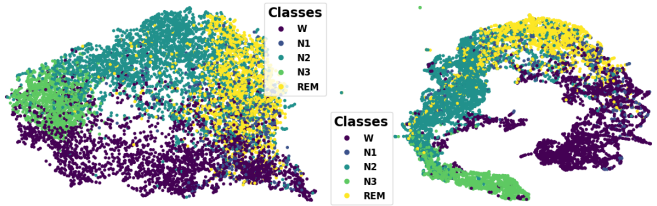
We assessed the contribution of each component in our ADAST framework, namely the unshared domain-specific attention module (**ATT**), the dual classifiers (**DC**) and self-training (**ST**). Particularly, we conducted an ablation study to show the results of different variants of ADAST in Table III.

The results emphasize three main conclusions. First, using the proposed domain-specific attention benefits the overall performance, as it helps to preserve the domain-specific features. Second, the self-training improves the classification performance by $\sim$ 1.1%. This improvement shows the benefit of incorporating the target domain class information in modifying the classification boundaries by using pseudo labels. Third, the addition of dual classifiers benefits the classification performance in overall as it avoids the variance in the training data. Moreover, combining it with the self-training in specific is helpful to further improve the performance by 2.5% through improving the quality of the pseudo labels.



(a) (b)

Fig. 4: UMAP feature space visualization showing the source and target domains alignment using (a) Source-Only, and (b) our ADAST, applied for the scenario S2→EDF.



(a) (b)

Fig. 5: UMAP feature space visualization showing the target domains classification performance after (a) Source-Only, and (b) our ADAST alignment, applied for the scenario S2→EDF.

G. Representation Visualization

In Section IV-E, the results illustrate the advantages of our proposed ADAST framework over the initial Source-Only performance. To make the comparison more intuitive, we visualized the feature representations that are learned during the training process using Uniform Manifold Approximation and Projection (UMAP) [42].

First, we investigated the alignment quality, where Fig. 4 visualizes the source and target alignment in the scenario S2→EDF. In particular, Fig. 4a shows the Source-Only align-

ment, and Fig. 4b shows our ADAST framework alignment. In these figures, the red dots represent the source domain, and the blue dots denote the target domain. We can observe that the Source-Only is not very efficient as many disjoint patches are not well-aligned with the target domain. However, our ADAST framework improves the alignment of the two domains to become arc-shaped, which increases the overlapped region and they become less discriminative.

Additionally, we investigated the target domain classification performance in the aforementioned scenario after the alignment in Fig. 5. In particular, Fig. 5a is the Source-Only performance, and Fig. 5b is the one after our alignment. We noticed that the Source-Only alignment generates a lot of overlapping samples from different classes, which degrades the target domain classification performance. On the other hand, our ADAST framework improves the discrimination between the classes and they become more distinct from each other. This is achieved with the aid of the iterative self-training strategy.

H. Domain-Specific vs Domain-Invariant Features

In this subsection, we show that domain-invariant features has been extracted in two different manners.

Quantitatively: To have a quantitative insight about extracting domain-specific information, we use the Kullback-Leibler (KL) divergence as a distribution similarity measure. In general, KL-divergence is a non-symmetric measure of the difference between two probability distributions $p(x)$ and $q(x)$. Specifically, the KL-divergence of $q(x)$ from $p(x)$, denoted as $D_{KL}(p(x), q(x))$, is a measure of the information lost when $q(x)$ is used to approximate $p(x)$ [43]. So, if we can approximate $p(x)$ with $q(x)$ without a big loss in information (achieve a low KL divergence value), then we can conclude more similarity between the two distributions, and vice versa.

To validate this idea, we first extract the reference features for each domain. To do so, we use the feature extractor network and the dual-classifiers to train the source domain data with the cross-entropy loss, and freeze the extracted features from the last epoch (i.e., R_s). Since we do not have access to the target domain labels, we use the pretrained model (on source domain data) to obtain the target domain features (i.e., R_t).

Next, we train the original model with source and target data to obtain the domain-invariant (DI) and domain-specific (DS) features before and after the domain-specific attention module respectively. Finally, we calculate the following KL-divergence on the source domain sides as: $D_{KL}(DI_s, R_s)$ and $D_{KL}(DS_s, R_s)$. Similarly, we calculate the following KL-divergence on the target domain sides as: $D_{KL}(DI_t, R_t)$ and $D_{KL}(DS_t, R_t)$.

We perform this experiment on the cross-domain scenario (EDF→S1), and provide the results in Table IV. Notably, the KL-divergence value is the minimum between the domain-specific features and their corresponding original domain features. This indicates a minimal information loss when trying to approximate the domain-specific features using the reference features, i.e., more similarity between them. Therefore, we conclude the efficacy of the domain-specific attention module in preserving the unique characteristics of each domain.

TABLE IV: KL-divergence between original features (R) and domain-invariant (DI) and domain-specific (DS) features applied for both source and target domains.

		Domain-Invariant (DI)	Domain-Specific (DS)
Source domain	R_s	0.7591	0.7346
Target domain	R_t	0.7849	0.6735

Visual Inspection To analyze the feature space of both domain-invariant and domain-specific features, we visualize their distribution in the scenario (EDF→S1) with UMAP, as shown in Fig. 6. We notice that the feature extractor can extract domain-invariant information by minimizing the distance between source and target distributions as shown in Fig. 6a. However, these domain-invariant features still misclassify some classes as seen in Fig. 6b. Meanwhile, the domain-specific attention helps to better align the domains (Fig. 6c), as well as improving the class-wise alignment as in Fig. 6d. In addition, the wake class (Magenta) is now closer to the Rapid Eye Movement (REM) class (Yellow), which is reasonable since both classes share related information. Similarly, the points of N3 class become closer to N2 class. This implies that our model well learns specific features that can be adapted to the new unseen domain.

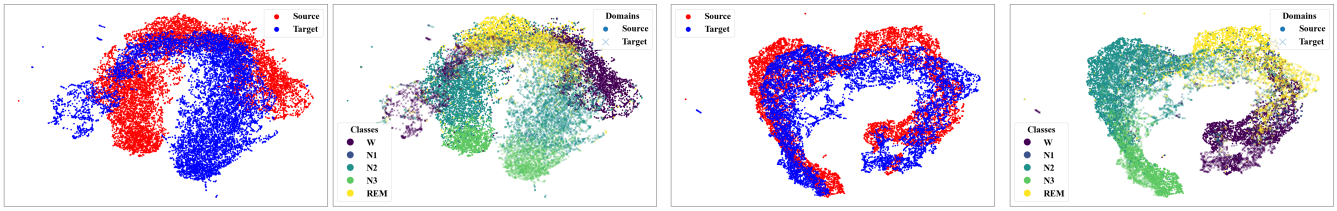
I. Sensitivity Analysis

Effect of target classification loss. Since the self-training process relies on target domain pseudo labels, it is not practical to assign a high weight to the target classification loss as the pseudo labels are expected to have some uncertainties. Therefore, we studied the effect of the different variants to the weight assigned to the target classification loss λ_1, as shown in Fig. 7.

Notably, when λ_1 is very small (i.e., λ_1 = 1e-6), it makes the self-training useless, and the performance becomes very close to the case without self-training. As we gradually increase λ_1 value, we notice improvement in the overall performance until we reach the optimal value of $\lambda_1 = 0.01$. Further increasing λ_1 deteriorates the performance as the model is highly penalized based on the pseudo labels which may contain false examples.

Effect of classifier weight constraint. Since the dual classifiers share the same architecture, it is important to keep their predictions relatively different but not with a big gap. The classifier weight constraint is the factor that keeps this distance with an acceptable margin, and hence, it becomes important to study the effect of this term and how its weight λ_2 should be selected. We analyzed the performance of our model with different λ_2 values, as illustrated in Fig. 7.

When λ_2 is very small, it makes the two classifiers perform very closely to each other, which has a similar performance with a single classifier. The performance is gradually improved when increasing λ_2, as the two classifiers tend to have different classification decisions. It can be found that the best performance is achieved with $\lambda_2 = 0.001$. However, as its value is increased beyond this threshold (i.e., 0.001), we notice that the



(a) (b) (c) (d)

Fig. 6: UMAP feature space visualization showing the domain-invariant alignment (a) domain-wise , and (b) class-wise, besides the domain-specific alignment (c) domain-wise, and (d) class-wise under the scenario of EDF→S1.

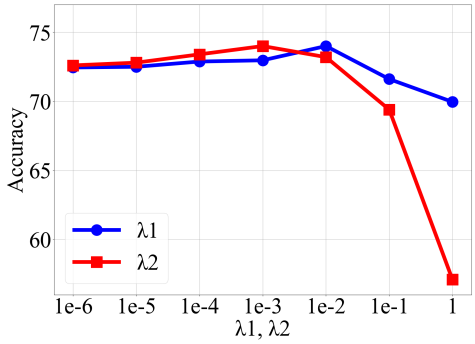


Fig. 7: Sensitivity analysis to the different variants of λ_1 and λ_2 in Eq.10.

overall performance degrades. This happens as the weights of the two classifiers became very dissimilar, moving them away from the correct predictions.

V. Conclusions

In this paper, we proposed a novel adversarial domain adaptation architecture for sleep stage classification using single-channel raw EEG signals. We tackle the problem of the domain shift that happens when training the model on one dataset (i.e., the source domain), and testing it on another out of distribution dataset (i.e., the target domain). We developed unshared attention mechanisms to preserve domain-specific features. We also proposed a dual classifier-based iterative self-training strategy, which helps the model to adapt the classification boundaries according to the target domain with robust pseudo labels. The experiments performed on six cross-domain scenarios generated from three public datasets prove that our model can achieve superior performance over state-of-the-art domain adaptation methods. Additionally, the ablation study shows that the dual classifier-based self-training is the main contributor to the improvement as it considers class-conditional distribution in the target domain.

References

[1] P. Memar and F. Faradji. A novel multi-class eeg-based sleep stage classification system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(1):84–95, 2018.

[2] A. Supratak, H. Dong, C. Wu, and Y. Guo. Deepsleepnet: a model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017.

[3] Panteleimon Chriskos, Christos A. Frantzidis, Polyxeni T. Gkivogkli, Panagiotis D. Bamidis, and Chrysoula Kourtidou-Papadeli. Automatic sleep staging employing convolutional neural networks and cortical connectivity images. *IEEE Transactions on Neural Networks and Learning Systems*, 31(1):113–123, 2020.

[4] Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y. Chen, and Maarten De Vos. Seqsleepnet: End-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. *international conference of the ieee engineering in medicine and biology society*, 27(3):400–410, 2019.

[5] Emadeldeen Eldele, Zhenghua Chen, Chengyu Liu, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:809–818, 2021.

[6] Samaneh Nasiri and Gari D. Clifford. Attentive adversarial network for large-scale sleep staging. *MLHC*, pages 457–478, 2020.

[7] Huy Phan, Oliver Y. Chén, Philipp Koch, Zongqing Lu, Ian Vince McLoughlin, Alfred Mertins, and Maarten De Vos. Towards more accurate automatic sleep staging via deep transfer learning. *IEEE Transactions on Biomedical Engineering*, pages 1–1, 2020.

[8] Huy Phan, Oliver Y. Chen, Philipp Koch, Alfred Mertins, and Maarten De Vos. Deep transfer learning for single-channel automatic sleep staging with channel mismatch. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[9] Stanislas Chambon, Mathieu N. Galtier, and Alexandre Gramfort. Domain adaptation with optimal transport improves eeg sleep stage classifiers. In *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, pages 1–4, 2018.

[10] Sajad Mousavi, Fatemeh Afghah, and U. Rajendra Acharya. Sleepeeg-net: Automated sleep stage scoring with sequence to sequence deep learning approach. *PLOS ONE*, 14(5):1–15, 2019.

[11] Fan Li, Rui Yan, Reza Mahini, Lai Wei, Zhiqiang Wang, Klaus Mathiak, Rong Liu, and Fengyu Cong. End-to-end sleep staging using convolutional neural network in raw single-channel eeg. *Biomedical Signal Processing and Control*, 63:102203, 2021.

[12] Wei Qu, Zhiyong Wang, Hong Hong, Zheru Chi, David Dagan Feng, Ron Grunstein, and Christopher Gordon. A residual based attention model for eeg based sleep staging. *IEEE Journal of Biomedical and Health Informatics*, 24(10):2833–2843, 2020.

[13] Huy Phan, Oliver Y. Chen, Minh C. Tran, Philipp Koch, Alfred Mertins, and Maarten De Vos. Xsleepnet: Multi-view sequential model for automatic sleep staging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[14] Ziyu Jia, Youfang Lin, Jing Wang, Ronghao Zhou, Xiaojun Ning, Yuanlai He, and Yaoshuai Zhao. Graphsleepnet: Adaptive spatial-temporal graph convolutional networks for sleep stage classification. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 2, pages 1324–1330, 2020.

[15] Wenpeng Neng, Jun Lu, and Lei Xu. Ccrrsleepnet: A hybrid relational inductive biases network for automatic sleep stage classification on raw single-channel eeg. *Brain Sciences*, 11(4):456, 2021.

[16] Kaare Mikkelsen and Maarten de Vos. Personalizing deep learning models for automatic sleep staging. *arXiv preprint arXiv:1801.02645*, 2018.

[17] Huy Phan, Kaare B. Mikkelsen, Oliver Y. Chén, Philipp Koch, Alfred Mertins, Preben Kidmose, and Maarten De Vos. Personalized automatic sleep staging with single-night data: a pilot study with kl-divergence regularization. *Physiological Measurement*, 41(6):64004, 2020.

[18] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2012.

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

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.

[21] Ranqi Zhao, Yi Xia, and Yongliang Zhang. Unsupervised sleep staging system based on domain adaptation. *Biomedical Signal Processing and Control*, 2021.

[22] Chaehwa Yoo, Hyang Woon Lee, and Jewon Kang. Transferring structured knowledge in unsupervised domain adaptation of a sleep staging network. *IEEE Journal of Biomedical and Health Informatics*, 2021.

[23] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. *ACM Trans. Intell. Syst. Technol.*, 11(5), 2020.

[24] Ha Bui, Toan Tran, Anh Tuan Tran, and Dinh Phung. Exploiting domain-specific features to enhance domain generalization. In *Advances in Neural Information Processing Systems*, 2021.

[25] Wenxuan Ma, Jinming Zhang, Shuang Li, Chi Harold Liu, Yulin Wang, and Wei Li. Exploiting both domain-specific and invariant knowledge via a win-win transformer for unsupervised domain adaptation. *arXiv preprint arXiv:2111.12941*, 2021.

[26] Fredrik D. Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 527–536, 2019.

[27] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1081–1090, 2019.

[28] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363, 2018.

[29] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, 2017.

[30] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018.

[31] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning*, pages 2988–2997, 2017.

[32] G. Zhang, L. Cui, R. Mueller, S. Tao, M. Kim, M. Rueschman, S. Mariani, D. Mobley, and S. Redline. The national sleep research resource: towards a sleep data commons. *Journal of the American Medical Informatics Association*, 25(10):1351–1358, 2018.

[33] S. F. Quan, B. V. Howard, C. Iber, J. P. Kiley, F. J. Nieto, G. T. O'Connor, D. M. Rapoport, S. Redline, J. Robbins, J. M. Samet, et al. The sleep heart health study: design, rationale, and methods. *Sleep*, 20(12):1077–1085, 1997.

[34] P. Fonseca, N. den Teuling, X. Long, and R. M. Aarts. Cardiorespiratory sleep stage detection using conditional random fields. *IEEE Journal of Biomedical and Health Informatics*, 21(4):956–966, 2017.

[35] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359, 2021.

[36] Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[37] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450, 2016.

[38] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, pages 81–94, 2020.

[39] Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10, 2020.

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):1–35, 2016.

[41] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, volume 31, pages 1640–1650, 2018.

[42] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[43] D Anderson and K Burnham. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, 63:10, 2004.