UNSUPERVISED ADVERSARIAL DOMAIN ADAPTATION FOR ACOUSTIC SCENE CLASSIFICATION

Shayan Gharib[1*], *Konstantinos Drossos*[1*], *Emre Çakir*[1], *Dmitriy Serdyuk*[2], *and Tuomas Virtanen*[1]

[1]Audio Research Group, Lab. of Signal Processing,
Tampere University of Technology, Tampere, Finland
[2]Montreal Institute for Learning Algorithms, Montreal, Canada

ABSTRACT

A general problem in acoustic scene classification task is the mismatched conditions between training and testing data, which significantly reduces the performance of the developed methods on classification accuracy. As a countermeasure, we present the first method of unsupervised adversarial domain adaptation for acoustic scene classification. We employ a model pre-trained on data from one set of conditions and by using data from other set of conditions, we adapt the model in order that its output cannot be used for classifying the set of conditions that input data belong to. We use a freely available dataset from the DCASE 2018 challenge Task 1, subtask B, that contains data from mismatched recording devices. We consider the scenario where the annotations are available for the data recorded from one device, but not for the rest. Our results show that with our model agnostic method we can achieve $\sim 10\%$ increase at the accuracy on an unseen and unlabeled dataset, while keeping almost the same performance on the labeled dataset.

Index Terms— Adversarial domain adaptation, acoustic scene classification

1. INTRODUCTION

The task of acoustic scene classification is to assign to a sound segment the acoustic scene that it belongs to, e.g. office, park, tram, etc. Recently proposed methods for acoustic scene classification (ASC) are based on deep neural networks (DNNs)[1, 2]. They usually employ convolutional neural networks (CNNs) to extract discriminative features from the used data, then using these features as an input to a classifier for classifying the acoustic scene [3, 4, 5, 6]. In a realistic scenario, a method for ASC will be used to classify data emerging from a variety of different domains (i.e. acoustic conditions, acoustic channels) from the data used for optimizing that particular method. The mismatched domains introduce the dataset bias (or domain shift) phenomenon [7, 8, 9], which results in a degradation of the performance of the method.

A typical countermeasure to this phenomenon is the fine tuning of the method using annotated data from different acoustic conditions. For example, one can retrain a method given a newly collected dataset. But, the annotation of audio data is a tedious process and it is more likely for one to have audio data but not having their annotations. To leverage knowledge from new and unlabeled data, one can use domain adaptation processes. Domain adaptation is a subspace alignment problem, where the goal is the alignment of the latent representations of the data coming from different domains [8, 10, 11, 12]. The impact of the domain adaptation is greater when none or few annotations (labels) exist for data from the different domains. These processes are referred as unsupervised and semi-supervised, respectively, domain adaptation.

Before the emergence of adversarial training, different approaches had been employed to cope with the problem of covariate sift across domains, e.g. kernel mean matching (KMM) [13, 14] and autoencoder scheme based approaches [15, 16, 17]. One of the first works in adversarial domain adaptation with deep neural networks is [18] where the classification and the alignment of the latent representations can occur at the same time. The alignment is performed by using the reverse gradient of the domain classification to optimize the parameters that produce the latent representation for the classification. A similar concept has been adopted in many subsequent works, e.g. [19]. Later, an adversarial domain adaptation approach is presented in [9], where the training procedure of classification and adaptation are not happening simultaneously. The first step obtains a non-adapted model and, in a second step, this model is adapted. This increases the performance of adaptation, compared to the previous existing methods. Another method used in [20] implements classification and reconstruction by employing three different feature extractors and one shared encoder. One of the feature extractors is shared between the domains, while the other two are domain exclusive. The classifier predicts the labels based on the shared features between domains. In addition, there is an adversarial objective function for shared features to help the adaptation by increasing the similarity of extracted features across domains. Another recent work [21] presents two models for source and target while regularizing their parameters by sharing a loss between each layer, targeting to mitigate the existing disparity between source and target distributions. The above methods evaluate the domain adaptation in the context of natural language processing, sentiment classification, and image classification. There are no previous studies in the context of acoustic scene classification.

Driven by the above, in this paper we present the first approach for unsupervised domain adaptation for acoustic scene classification. We investigate the unsupervised domain adaptation scenario, i.e. the acoustic scene labels of the new data are not known during the adaptation part. We use the data from the DCASE 2018 Task 1, subtask B, which consist of recordings from mismatched recording devices [22]. We consider the difference in the acoustic channel, imposed by the different recording devices, as the domains. To mitigate this difference, we introduce a model agnostic process where we encourage the model to match the distributions of the learned representations of the data coming from the annotated (source domain) and the non-annotated (target domain) sets. The contributions of this paper are the following:

1. We follow a recently proposed general framework for adversarial domain adaptation [9] and we alter it by introducing

[*]Equally contributing authors

extra learning signals during the adaptation process;

2. We present the first application of deep neural network based, unsupervised domain adaptation for acoustic scene classification showing the effect of leveraging unlabeled data for acoustic scene classification, through unsupervised domain adaptation.

The rest of the paper is organized as follows. In Section 2 we explain the proposed method, and in Section 3 we present the evaluation procedure that we followed, including the presentation of the dataset used and the models implemented, and the details of the training and testing procedures. The obtained results are reported and discussed in Section 4, followed by the conclusions and proposals for future work in Section 5.

2. PROPOSED DOMAIN ADAPTATION METHOD

The data from the source domain are classified according to a specific set of labeled acoustic scenes. For example, in our study, these acoustic scenes are *airport*, *bus*, *airport*, *metro*, *metro station*, *park*, *public square*, *shopping mall*, *street pedestrian*, *street traffic*, and *tram*. The goal is to assign the target domain data to this same set of labels. The source and target domains data are time-frequency representation of audio (e.g. log mel-band energies). We follow the general framework for adversarial domain adaptation in [9] and choose not to tie the parameters of the source and adapted (target) models. Our models are neural networks that are used to extract the discriminative latent representation of the input data. This representation is used for the label classification by the classifier. The source model is the model optimized with the source data and the target model is the one adapted to the target data. Our presented method is independent of the architecture of the utilized model and concerns the adaptation of a model optimized on the source domain, to the target domain. For this reason, in this section, we present the method for the domain adaptation, and in Section 3 we present the specific models employed.

Having annotated (i.e. with reference labels) data from the source domain, $\mathbf{X}^S = \{\mathbf{X}_1^S, \mathbf{X}_2^S, \ldots, \mathbf{X}_{N_S}^S\}$, and the non-annotated target domain data, $\mathbf{X}^T = \{\mathbf{X}_1^T, \mathbf{X}_2^T, \ldots, \mathbf{X}_{N_T}^T\}$, the goal is to regularize a model M to produce feature mappings of the source domain, $M(\mathbf{X}^S)$, and of the target domain, $M(\mathbf{X}^T)$, that exhibit the same distribution. Then a classifier, trained on $M(\mathbf{X}^S)$, can be used in order to classify $M(\mathbf{X}^T)$. For this process, we employ three steps. At the first step, we pre-train the model M and the classifier C using dataset $\mathbf{X}^S$. Then, at the second step, we use adversarial training (as in generative adversarial network (GAN) [23]) to match the distributions of $M(\mathbf{X}^S)$ and $M(\mathbf{X}^T)$. Finally, at the third step, we test the performance of the classifier on the $M(\mathbf{X}^T)$. All the three steps of the process are schematically illustrated at Figure 1.

We differ from the original proposal of the general framework for adversarial domain adaptation in [9] by utilizing the label classifier C also during the adaptation step. Also, we differ from proposals with gradient reversing, e.g. [8], because the classifier C is not the domain classifier but the label one. We experimentally found that, for our task, the original setup (i.e. without C in the adaptation step) cannot work. In this setup, the adapted model was exhibiting worse label classification performance to both the target and the source domains, compared to the non-adapted one. Observing the adaptation process, we hypothesized that the learning signals used in [9] were not able to drive the model M to produce feature mappings that can be used for later target domain classification from C. Thus, we utilize the C in order to provide an additional learning signal during the adaptation process. This results in more stable domain adaptation process and the adapted model exhibits increased performance at the target domain, compared to the non-adapted one.

We start by having the data from the source domain, $\mathbf{X}^S$, and their corresponding one-hot-encoded labels for the acoustic scene, $\mathbf{Y}^S = \{\mathbf{y}_1^S, \mathbf{y}_2^S, \ldots, \mathbf{y}_{N_S}^S\}$. The first goal is to obtain a model and a classifier that are able to classify the source data (i.e. label and not domain classification). To this end, we utilize the $\mathbf{X}^S$ and $\mathbf{y}^S$ to train our source domain model, M_S, and pretrain the classifier C, by minimizing the loss

$$\mathcal{L}_S = -\sum_{n=1}^{N_S} \mathbf{y}_n^S \log(C(M_S(\mathbf{X}_n^S))), \tag{1}$$

At the second step, we target to obtain a model that can produce mappings of the data from the source and target domains, that they are as close as possible in terms of their distribution. Since we do not have the labels of the target domain, we can only leverage knowledge from the source data and their labels, and from the data of the target domain. Adopting the approach in [9], we use the adversarial training to match the distributions of $M(\mathbf{X}^S)$ and $M(\mathbf{X}^T)$. Specifically, we use an additional, target domain model, M_T, having the same architecture and amount of parameters as M_S. We do not use any constraints between M_S and M_T (e.g. parameters sharing/coupling between M_S and M_T), but we initialize the parameters of M_T with the ones from M_S. Additionally, we use a domain discriminator D that will be optimized to identify if its input is coming from the distribution of the source or the target domain (hence, its output is an indication if its input was or not from the source domain).

We jointly optimize the M_T and D in order to enforce the distribution of the $M_T(\mathbf{X}^T)$ to be as close as possible to the distribution of the $M_S(\mathbf{X}^S)$. In the GAN terminology, one can think the M_T as the generator, the M_S as the real examples, and the D as the discriminator. The output of the generator and real examples are given as an input to the discriminator, and the latter is optimized to identify which is real and which is coming from the generator. At the same time, the generator is optimized to fool the discriminator in believing that the output of the generator is also a real example. In our method, $M_S(\mathbf{X}^S)$ (real examples) and $M_T(\mathbf{X}^T)$ (generator output) are given as an input to the discriminator D. The latter is optimized to identify if its input is $M_S(\mathbf{X}^S)$ or $M_T(\mathbf{X}^T)$. At the same time, we optimize M_T in order to fool D that $M_T(\mathbf{X}^T)$ is $M_S(\mathbf{X}_S)$. We minimize the losses

$$\mathcal{L}_D = -\sum_{n=1}^{N_S} (\log D(M_S(\mathbf{X}_n^S) + \log(1 - D(M_T(\mathbf{X}_n^T))) \text{ and} \tag{2}$$

$$\mathcal{L}_{M_T} = -\sum_{n=1}^{N_S} (\log D(M_T(\mathbf{X}_n^T) + \mathbf{y}_{S_n} \log(C(M_T(\mathbf{X}_n^S))). \tag{3}$$

$\mathcal{L}_D$ is minimized w.r.t D and the $\mathcal{L}_{M_T}$ w.r.t. M_T. In the case where $N_T < N_S$ or $N_T > N_S$, then $\mathbf{X}^T$ will be either oversampled or undersampled, respectively. The minimization of $\mathcal{L}_D$ and $\mathcal{L}_{M_T}$ can be performed jointly or in an alternating way, e.g. do an update of D towards minimizing $\mathcal{L}_D$, then update M_T towards minimizing $\mathcal{L}_{M_T}$, and repeat until some criterion is met. The total loss, e.g. $\mathcal{L}_D + \mathcal{L}_{M_T}$, is a typical minimax objective for adversarial training as it has been used in [9, 12].The actual implementation of the minimization process is tied to the employed models of M_S, M_T,

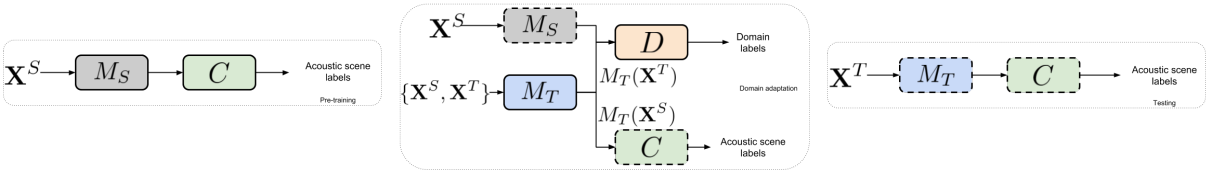


Figure 1: Illustration of the three steps of the domain adaptation method: pre-training; adversarial domain adaptation; and testing. Solid lines indicate the models that are optimized in the corresponding steps, and dashed lines indicate the models that are not optimized.

D, and C, and the dynamics of the training process. Our followed procedure is presented in Section 3. Finally, we use M_T with C in order to classify the $\mathbf{X}^T$.

3. EVALUATION

To assess the performance of our method we focus on the task of acoustic scene classification. We employ a freely available dataset that provides audio data recorded with mismatched recording devices. For model M, we employ two different models; one that achieved the first place in an acoustic scene classification contest[1] and a second that is the published baseline model for the Task 1, subtask B, of the DCASE challenge 2018 [22]. For the rest of the paper, we will refer to the former model as the Kaggle model and to the latter as the DCASE model. All hyper-parameters reported in this section are the same as in the proposed models. All models are implemented using the freely available PyTorch framework[2] and our code can be found online.[3]

3.1. Dataset and data preprocessing

The dataset used for the development and evaluation of our method is the one provided as the development dataset of Task 1, subtask B, of the DCASE 2018 challenge [22]. The dataset is collected with three different recording devices. The main recording device which is referred to as device A consists of a binaural microphone and a recorder using 48 kHz sampling rate and 24 bit resolution. The data from this device were re-sampled and averaged into a single channel to match the characteristic of data recorded by device B and C. The rest of data have been recorded using customer devices such as smart phones and cameras which are referred to as device B and C. This dataset contains a total of 28 hours of audio out of which 24 hours are from device A, 2 hours from device B, and 2 hours from device C. The proposed evaluation setup by the organizers of the DCASE 2018 challenge, 30% of audio files of device A, and 25% of device B and C are dedicated to the validation set.

During the development of our method, the annotations of evaluation/test data of the Task 1, subtask B, were not publicly available. Therefore, we use the original (i.e. proposed by the evaluation setup of the DCASE Task 1, subtask B) validation data as our test data (referred to as test data for the rest of this paper), a randomly selected 10% of the original training data as our validation (referred to as validation data from now on), and the rest of the original training data as our actually training data (referred to as training data from now on). This means that we use 5510 files from device A, 486 files from device B, and 486 files from device C as training data. 612, 54, and 54 files from device A, B, and C, respectively, are used as our validation set. We test our method on 2518, 180, and 180 files, from devices A, B, and C respectively which is equivalent to original validation set of the Task 1, subtask B.

From the available files, we extracted 64 log Mel-band energies, using a 2048 samples ($\sim$ 46 ms) Hamming window and 50% overlap. The extracted features from the data recorded from device A is our source domain data. The rest (B and C) are our target domain data. We use the Librosa package for feature extraction.[4] Since the amount of data for the target domain (i.e. the data from the B and C devices) are less than the source domain data (i.e. the data from device A), we oversampled the data from target domain to have the same amount of training data as the number of samples from device A, approximately 5.6 times more than the original size.

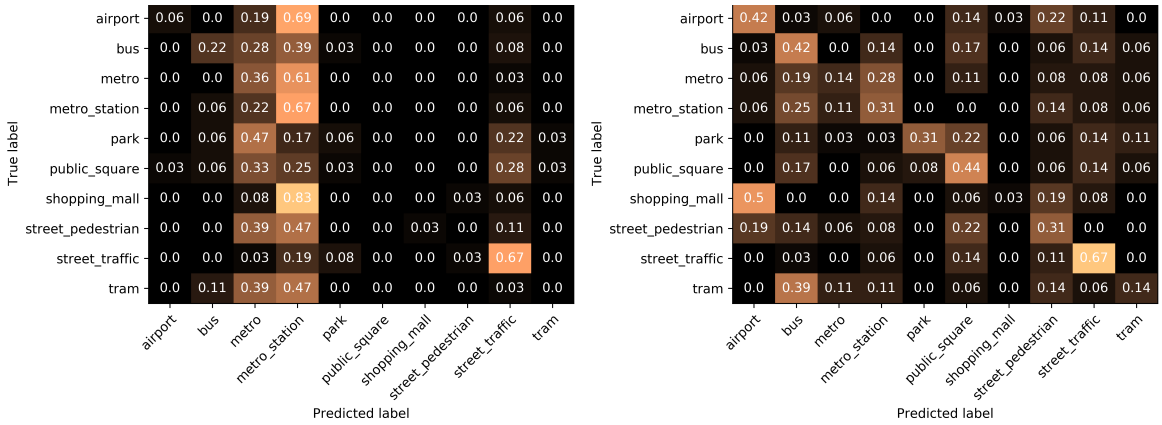
3.2. Models used

Since our proposed method is independent of the employed model, we evaluate it on two different and published models. The first is the Kaggle model and the second is DCASE baseline model. The Kaggle model is mainly a convolutional neural network (CNN) which has 5 convolutional layers, with kernel sizes of {(11, 11), (5, 5), (3, 3), (3, 3), (3,3)} and amount of channels/filters of {48, 128, 192, 192, 128}. The first two convolutional layers use strides of (2,3) and the rest (1,1). The first two convolutional layers together with the last one are followed by rectified linear unit (ReLU) non-linearity, max pooling layer, and batch normalization. The rest convolutional layers are followed only by the ReLU non-linearity. DCASE model consists of two convolutional layers with 32 and 64 filters, respectively. Both layers have a (7, 7) kernel size followed by batch normalization, ReLU non-linearity, and a max pooling operation. The kernels of the pooling operations are {(5, 5), (4, 100)}.

We use 64 log mel-band energies, but for the development of the DCASE model, 40 mel band energies were used. Therefore, we had to slightly alter the DCASE model in order to utilize our data. Specifically, we altered the kernel of the first pooling operation and the padding of the second convolutional layer. That is, we used kernel size of (8,4) for the pooling operation and we specified the padding for the second convolutional layer at (3,0). Because the Kaggle and the DCASE models had a different dimensionality of their outputs, we used two different discriminators.

As the discriminator D, when employing the Kaggle model, we use three convolutional layers all with a kernel size of (3,3) and {64, 32, 16} as the number of channels/filters. All layers are followed by the ReLU non-linearity and batch normalization. The output of the third convolutional layer is flattened and given as an input to a linear layer, which outputs the prediction for samples as source or target. As a discriminator for the DCASE model we used one linear layer. The input to our label classifier C is the output of the last layer of the model M which is turned to a vector (i.e. flattened) and is given as an input to three for the Kaggle and two for the DCASE model linear layers followed by 25%, for the Kaggle model, and 30%, for

[1] https://www.kaggle.com/c/acoustic-scene-2018
[2] https://pytorch.org/
[3] https://github.com/shayangharib/AUDASC
[4] https://librosa.github.io/librosa/



(a) Confusion matrix for the non-adapted Kaggle model (b) Confusion matrix for the adapted Kaggle model

Figure 2: Confusion matrix of the **non-adapted**, (a), and **adapted**, (b), **Kaggle model** for the **target domain**. The values are normalized according to the amount of examples in each class. Brighter color indicates higher value.

the DCASE model, dropout. The non-linearity of all except the last layer of the classifier is the ReLU. Lastly, the output of our classifier is followed by a softmax non-linearity.

3.3. Training and testing procedure

For the pre-training step, we use a minibatch size of 38 samples, all selected from source domain. During the domain adaptation process we used a minibatch size of 16 samples, out of which 10 were selected from the source domain and 6 from the target domain (more specifically 3 from device B and 3 from device C). For the pre-training and the domain adaptation process, Adam was selected as the optimizer with learning rate of $1\mathrm{e}{-4}$ and other values according to the ones presented in the original paper [24]. We updated the parameters of the M_S and M_T after each iteration but (according to experimental observations) we updated the parameters of the discriminator D after 10 iterations. We stopped the optimization procedure in pre-training and domain adaptation processes after 350 and 300 epochs respectively.

4. RESULTS AND DISCUSSION

We report the obtained accuracy of the label classification when using the non-adapted models (i.e. in the pre-training step) and when using the models after the domain adaptation process (i.e. adapted models), using the data from source and target domains. Table 1 presents the obtained accuracy on the source and target domains when using the Kaggle model and when using the DCASE model, respectively. Additionally, we present the confusion matrices of the label classification for the target domain of the non-adapted and adapted Kaggle model in Figure 2, where can be seen that the adapted model manages to increase significantly the correctly classified examples from all labels. This is easily visualized by the diagonal of the confusion matrices, where in Figures 2a and 2b there is a considerable difference. Additionally, our proposed method manages to increase the performance of the classification for different models. That is, no matter the architecture of the model M, by following our proposed method there is an increase on the target domain without significant decrease in the performance for the source domain. In fact, we managed to increase also the performance on the source domain for the adapted model. This is apparent in Table 1, where the obtained accuracy for the adapted models and the target domain is greater, compared to the non-adapted. Furthermore, from the same table can be seen that the reduction in the accuracy at the source domain is around 0.5% for the DCASE model. The accuracy is marginally (i.e. $\sim 0.1\%$) greater for the source domain and the adapted Kaggle model (i.e. 65.25% to 65.37%). We attribute this small increase to the usage of the label classifier C during the domain adaptation process.

Table 1: Obtained accuracy for the non-adapted and adapted Kaggle and DCASE models.

	Kaggle model		DCASE model	
	Non adapted	**Adapted**	**Non adapted**	**Adapted**
Source	65.25%	65.37%	61.71%	61.23%
Target	20.28%	31.67%	19.17%	25.28%

5. CONCLUSIONS AND FUTURE WORK

We presented the first unsupervised adversarial domain adaptation for acoustic scene classification, which is also independent of the actual models used. The goal of our method is the adaptation of a pre-trained model on a source dataset, to a new and unseen target dataset. In a GAN-like setting, the adapting model tries to fool a discriminator that its output comes from the source dataset, while the non-adapted model informs the discriminator about the data that really coming from the source dataset.

We managed to increase the performance of the used models to the unseen dataset by approx. 10%. This indicates that the domain adaption approaches can provide an appealing solution for the problem of mismatched training and testing data, regarding the acoustic scene classification. As future directions we suggest the adoption of different GAN losses and the usage of domain adaptation for sound event detection.

6. ACKNOWLEDGMENT

S. Gharib, E. Çakir, K. Drossos, and T. Virtanen wish to acknowledge the CSC-IT Center for Science, Finland, for computational resources. Part of the computations leading to these results was performed on a TITAN-X GPU donated by NVIDIA to K. Drossos. Part of the research leading to these results has received funding from the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

7. REFERENCES

[1] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "Dcase 2016 acoustic scene classification using convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE2016), Budapest, Hungary*, 2016.

[2] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *the Detection and Classification of Acoustic Scenes and Events (DCASE)*, pp. 1–5, 2017.

[3] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," DCASE2017 Challenge, Tech. Rep., September 2017.

[4] Z. Weiping, Y. Jiantao, X. Xiaotao, L. Xiangtao, and P. Shaohu, "Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion," DCASE2017 Challenge, Tech. Rep., September 2017.

[5] B. Lehner, H. Eghbal-Zadeh, M. Dorfer, F. Korzeniowski, K. Koutini, and G. Widmer, "Classifying short acoustic scenes with I-vectors and CNNs: Challenges and optimisations for the 2017 DCASE ASC task," DCASE2017 Challenge, Tech. Rep., September 2017.

[6] S. Park, S. Mun, Y. Lee, and H. Ko, "Acoustic scene classification based on convolutional neural network using double image features," DCASE2017 Challenge, Tech. Rep., September 2017.

[7] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate shift and local learning by distribution matching*. Cambridge, MA, USA: MIT Press, 2009, pp. 131–160.

[8] F. Alam, S. Joty, and M. Imran, "Domain adaptation with adversarial training and graph embeddings," in *32nd International Conference on Machine Learning (ICML 2015)*, vol. 37, Jul. 2015, pp. 1180–1189.

[9] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2962–2971.

[10] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *32nd AAAI Conference on Artificial Intelligence*, Feb. 2018.

[11] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *2013 IEEE International Conference on Computer Vision*, Dec. 2013, pp. 2960–2967.

[12] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Jul. 2015.

[13] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," 2009.

[14] Y.-Q. Miao, R. Araujo, and M. S. Kamel, "Cross-domain facial expression recognition using supervised kernel mean matching," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2. IEEE, 2012, pp. 326–332.

[15] J. Deng, Z. Zhang, F. Eyben, and B. Schuller, "Autoencoder-based unsupervised domain adaptation for speech emotion recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1068–1072, 2014.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[17] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," *arXiv preprint arXiv:1206.4683*, 2012.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[19] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2018.

[20] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 343–351.

[21] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[22] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," *ArXiv e-prints*, July 2018.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.