Distribution Mismatch Correction for Acoustic Scene Classification

Lukas Maier, Alexander Fuchs and Franz Pernkopf

August 24, 2023

Distribution Mismatch Correction for Acoustic Scene Classification

Lukas Maier, Alexander Fuchs, Franz Pernkopf

Christian Doppler Laboratory for Dependable Intelligent Systems in Harsh Environments, Graz University of Technology, Austria

Email: `contact@lukeiggy.com, fuchs@tugraz.at, pernkopf@tugraz.at`

Abstract

While deep learning methods have shown immense benefits for Acoustic Scene Classification (ASC) tasks in terms of performance, they also introduce new challenges as these methods are prone to suffer from large performance degradation for out of distribution data. To build robust ASC models that can achieve reliable performance across multiple recording devices, the architecture has to be able to quickly adapt to changing input and activation distributions. We present ASCMobConvNet, a CNN architecture based on Mobile Inverted Bottleneck Convolutions. In order to better adapt to domain shifts and the resulting change in activation distributions, it uses sub-spectral normalization layers in combination with residual normalization instead of batch normalization layers. Furthermore, the model corrects non-parametric mismatches in the activation distributions through the integration of Wasserstein distribution correction layers. Using our proposed architecture we are able to achieve an test accuracy of 68.10 % on the TAU Urban Acoustic Scenes 2020 Mobile development dataset. Using Wasserstein distribution correction layers we can further improve the accuracy by 0.68 %.

1 Introduction

As many interactive applications need to be optimized or adjusted for specific scenarios, information about the environment of the user or a device can provide valuable information and feedback to the application. Acoustic Scene Classification (ASC) aims to provide this information by recognizing the acoustic scene of a given audio environment. The analyzed audio samples are recorded within different acoustic scenes such as a public street, a restaurant or a bus driving in an urban environment. Acoustic scenes may vary significantly within each class and on the contrary they often share similar characteristics among different classes. This fact is challenging for ASC systems.

Another level of complexity is introduced via the diversity of devices that are used to record the audio samples. Each recording device has its own characteristic frequency response. This means that the spectral representation of an acoustic scene recorded by one device can differ significantly for other devices. Hence, the machine learning model used for the ASC task has to generalize with respect to the different recording devices.

Therefore, there have been multiple specialized Convolutional Neural Network (CNN) architectures proposed for ASC, often targeting a low computational complexity due to hardware limitations. CP_Resnet [1] provides a low-complexity solution by limiting the size of the receptive field. BC-ResNet-Mod [2] goes one step further and reduces the dimensionality by applying one-dimensional convolutional layers to a two-dimensional problem. Mini-SegNet [3], another low-complexity architecture which is based on [4], uses an encoder-decoder architecture. It is further reduced in size by pruning, quantization and knowledge distillation. Other specialized architectures [5, 6] often use MobileNet [7] or VGG [8] as their base architecture.

The generalization ability of these specialized architectures can be improved via a variety of other methods targeting all aspects of the modelling pipeline, i.e., data pre-processing, training regularization, and test-time adaption. One suitable data pre-processing method for ASC, used during the data generation phase, is spectrum correction [9, 10]. It applies a separate correction coefficient to each bin of a magnitude spectrogram in order to unify the input data distributions over multiple input devices. Another method is presented in [2], where a network inspired by the pix2pix framework [11] is used to translate spectrograms from one recording device to another simulated recording device. During training, improved generalization can be achieved with a wide variety of methods such as data augmentation, dropout, weight decay regularization, etc. [12, 13].

However, often input distribution shifts occur during the deployment of the model due to not anticipated influences such as noise and sensor, i.e. microphone drifts. These distribution shifts can cause great performance degradation for neural network based ASC systems [14]. Therefore, domain adaptation methods have been developed to mitigate these domain shifts [15, 16].

While most CNN-based models rely on Batch Normalization (BN) to normalize the activation statistics between layers, for ASC tasks it has been shown that Sub-Spectral Normalization (SSN) [17] can achieve improved performance. SSN layers are similar to BN layers. However, they split the frequency dimension into multiple groups and perform a separate normalization for each of these groups. This makes models using SSN better suited to ASC tasks. However, this requires to adapt most domain adaptation methods as these are usually designed for BN [15, 16].

Furthermore, the diversity of recording devices needs to be addressed. While for many tasks it is sufficient to normalize the input data to zero mean and unit variance using training set statistics, this static approach can fail if the domain shifts between training and test set are too large. This is the case for multi-device ASC tasks. To reduce the impact of these domain shifts, it is possible to apply Residual Normalization (RN) layers instead of dataset standardization [2]. These layers rely on a combination of Instance Normalization with a linear shortcut and are specialized for ASC processing [18]. Since the RN layers are inserted at multiple stages throughout a network, in contrast to the dataset standardization which is only applied to the input data, the use of RN is also an effective measure against domain shifts at activation level. While RN, due to its sample specific normalization, makes the model less sensitive to distribution shifts it can only account for parametric shifts within the activation distributions. Therefore, to further reduce the impact of domain shifts on the model, we need to investigate domain adaptation methods, that are (i) appli-

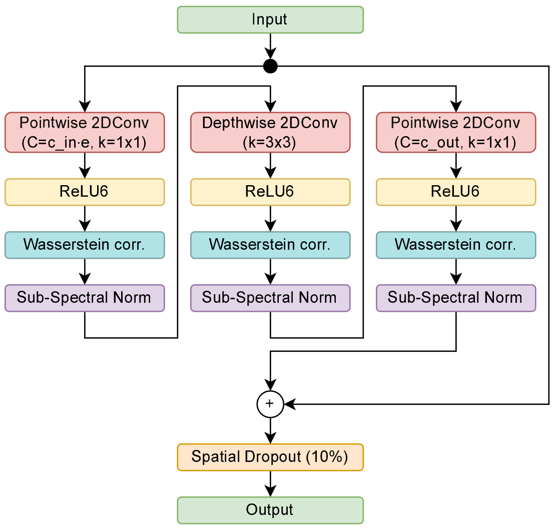

Figure 1: Structure of a 2D MobConv block.

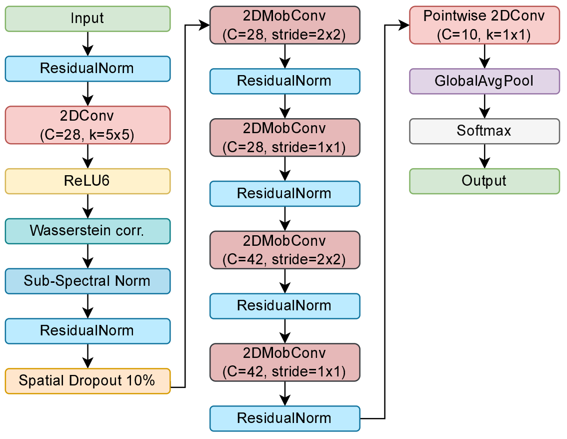

Figure 2: Structure of the ASCMobConvNet. It uses MobConvs and Wasserstein correction layers. Depending on the setting it uses SSN instead of BN layers and residual normalization over the frequency axis instead of zero-mean unit variance normalization. The RN layers are inserted at the input and between each convolutional stage.

cable to SSN and RN layers, and (ii) that can correct for non-parametric distribution shifts.

Our Wasserstein distance-based distribution correction layer [19] shows both properties, i.e. it is independent of the used normalization layer type and it can also perform non-parametric adaptions. This method adapts the activations in a layer during test-time to better match the distribution of activations observed during training using an energy-based minimization scheme. It can be added to the model after the model is trained. The Wasserstein correction is done for all devices in the dataset.

In this paper, we introduce ASCMobConvNet, a CNN architecture based on Mobile Inverted Bottleneck Convolutions [20]. In order to better adapt to domain shifts and the resulting change in activation distributions, it uses SNN layer in combination with RN layers instead of batch normalization layers and dataset standardization. Additionally, we apply Wasserstein correction layers to the optimized ASC model to improve performance and make the model even more robust. For our experiments, we use the TAU Urban Acoustic Scenes 2020 Mobile development dataset [21], which was published in the course of the annually hosted Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. It contains 64 h of audio data, recorded across ten different European cities. The audio samples were recorded with nine different recording devices. While the representation of these recording devices is balanced in the test set, it is greatly unbalanced in the training set, with some of the recording devices being only available in the test set. Therefore, this test set includes large domain shifts and an ASC model applied to this data must generalize well with regard to the different recording devices.

The remainder of this paper is organized as follows. We introduce the ASCMobConvNet in Section 2. In Section 3 we present the Wasserstein correction approach in detail. The experiments and results are summarized in Section 4. Finally, in Section 5 we conclude the paper.

2 Mobile Inverted Bottleneck Convolution Network

Our proposed CNN architecture called ASCMobConvNet is based on Mobile Inverted Bottleneck Convolutions (MobConvs) [20], which were introduced in the MobileNetV2 architecture and were adopted for the ASC task. The used 2D-MobConv blocks, as shown in Figure 1, consist of three main parts: (i) a point-wise convolution that is used to increase the channel count for the bottleneck, (ii) a depthwise convolution within the bottleneck, and (iii) another point-wise convolution to again reduce the number channels. After each convolution layer ASCMobConvNet uses a ReLU6 activation [7], an optional Wasserstein correction layer and a Sub-Spectral Normalization (SSN) layer [17]. Here, we use ReLU6 instead of ReLU because it has been shown to be more robust and suitable for low-complexity applications. Instead of the more commonly used Batch Normalization (BN) we use SSN, as it is frequency-aware and therefore has the ability to better learn characteristics in the audio data. SSN implements the same basic functionality as BN, but splits the frequency axis into multiple separate sub-bands. After the last SSN layer we add a residual connection and a spatial dropout layer [22]. The number of parameters of a MobConv block is mainly determined by the kernel size of the depthwise convolution, by the stride and by the expansion ratio, which is the factor by which the number of channels in the bottleneck is increased.

Regarding normalization, we replace the more traditional zero-mean unit variance data standardization, which is only applied to the input tensor, with RN layers performing normalization over the frequency axis of the input and activation tensors [2]. These layers are then applied at multiple stages throughout the network. RN for input activations $\mathbf{a}^{(m)} \in \mathbb{R}^{F \times T \times C}$ of sample m [1]

$$RN(a_{ftc}^{(m)}) = \lambda_{RN} \cdot a_{ftc}^{(m)} + \frac{a_{ftc}^{(m)} - \mu_f}{\sqrt{\sigma_f^2 + \epsilon}}, \tag{1}$$

[1] F, T and C denote the number of frequency bins, time bins and channels, respectively.

where λ_{RN} is a scaling parameter, ϵ is a small positive constant, $\mu_f = \frac{1}{TC}\sum_{t=1}^{T}\sum_{c=1}^{C} a_{ftc}$, and $\sigma_f^2 = \frac{1}{TC}\sum_{t=1}^{T}\sum_{c=1}^{C}(a_{ftc} - \mu_f)^2$. As shown in Figure 2, the corresponding Residual Normalization (RN) layers are inserted at the beginning of the network and between each convolutional stage, where each of the RN layers uses a residual factor of $\lambda_{RN} = 0.1$. The initial 2D-convolution layer in the input stage uses 28 channels, a 5×5-kernel and a stride of 1×1. It is followed by a ReLU6 activation layer, a Wasserstein correction layer, a SSN layer, and a RN layer. The stacking of normalization layers (SSN and RN) with different modes of normalization allows for improved robustness during test-time and increased convergence speed during training. Here, RN is independent of training set statistics and can adapt to changes in the activation distribution, while SSN allows for faster convergence during training. This combination has shown to be effective within our experiments. At the end of each MobConv block a spatial dropout layer is applied to regularize the training.

The backbone of the network consists of four MobConv blocks. The first two of these blocks use 28 channels, while the latter two use 42 channels. Here, the first and the third block use a stride of 2×2 to reduce the size of the feature map, while the second and the last block keep the feature map size using a stride of 1×1. All MobConv blocks use a 3×3-depth-wise kernel, an expansion ratio of 6 and a spatial dropout probability of 10 %.

The output stage of the network uses a point-wise convolution layer which reduces the channel count to the $C = 10$ classes in our dataset, resulting in an activation tensor $\mathbf{a} \in \mathbb{R}^{F \times T \times 10}$. A global average pooling layer then reduces the frequency and the time axes to a single bin creating the output logits $\mathbf{y} \in \mathbb{R}^{C}$. The last layer is a softmax activation computing the output probabilities for each of the 10 classes. Using the proposed configuration, ASCMobConvNet comprises only 75.9K trainable parameters, implementing a hardware-friendly architecture.

3 Non-parametric Wasserstein domain adaptation

Changes in the input data distribution also introduce an activation distribution mismatch between training and test set. The non-parametric Wasserstein distance-based correction layers reduce this mismatch for the test-time activations $\mathbf{a}^{(m)}$ of *each* audio sample m, improving the classification performance of CNN architectures. The advantage of using the one-dimensional Wasserstein distance between the target distribution $q(\mathbf{t})$ and the test-time distribution $p(\mathbf{a}^{(m)})$ is that it can be determined by a simple sort operation. Since we use only a single one-dimensional distribution per layer, we flatten the activations ($\mathbb{R}^{F \times T \times C}$) to a vector of length $N = F * T * C$, $\mathbf{a}^{(m)} \in \mathbb{R}^N$. This approach frames the distribution correction as a denoising problem using an energy minimization scheme. The energy $E(\tilde{\mathbf{a}}|\mathbf{a})$ is composed of two terms $\mathscr{R}(\tilde{\mathbf{a}})$ (corresponding to the prior) and $\mathscr{D}(\mathbf{a}|\tilde{\mathbf{a}})$ (corresponding to the likelihood) such that

$$E(\tilde{\mathbf{a}}|\mathbf{a}) = \lambda_1 \cdot \mathscr{D}(\mathbf{a}|\tilde{\mathbf{a}}) + \lambda_2 \cdot \mathscr{R}(\tilde{\mathbf{a}}), \tag{2}$$

using the weighting factors λ_1 and λ_2. Here $\mathscr{D}(\mathbf{a}|\tilde{\mathbf{a}})$ is the mean squared error between the corrected activations $\tilde{\mathbf{a}}$ and the original activations $\mathbf{a}$ which is minimized via

$$\frac{\mathrm{d}\mathscr{D}}{\mathrm{d}\mathbf{a}} = \mathbf{a} - \tilde{\mathbf{a}}. \tag{3}$$

$\mathscr{R}(\tilde{\mathbf{a}})$ is the one-dimensional Wasserstein distance to a predefined target vector $\mathbf{t} \in \mathbb{R}^N$. Here the target distribution $\mathbf{t}$ is chosen to be the Wasserstein barycenter of the training set distribution of size K, which is defined as:

$$t_{(i)} = \frac{1}{K}\sum_{m=1}^{K} a_{(i)}^{(m)}, \tag{4}$$

where $a_{(i)}^{(m)}$ are the sorted activation values of sample m,

$$[a_{(i)}^{(m)}], \mathbf{j} = \mathrm{sort}(\mathbf{a}^{(m)}). \tag{5}$$

Here $\mathbf{j}$ are the original indices of the activations, which are required to assign the correction to the corresponding activations. The updates to the activations $\mathbf{a}^{(m)}$ are then calculated by minimizing the Wasserstein 1-distance, which is defined as:

$$W_1\big(p(\mathbf{a}^{(m)}), q(\mathbf{t})\big) = \sum_{i=1}^{N} ||a_{(i)}^{(m)} - t_{(i)}||, \tag{6}$$

where $t_{(i)}$ are the sorted target values and $a_{(i)}^{(m)}$ are the sorted test-time activations. The update minimizing Eqn. (6) is defined via,

$$\boldsymbol{\Delta}^{(m)} = [t_{(i)}] - [a_{(i)}^{(m)}]. \tag{7}$$

The update tensor $\boldsymbol{\Delta}^{(m)}$ is used for updating the corresponding indices $\mathbf{j}$ within the activation tensor, resulting in the updated activations $\tilde{\mathbf{a}}$, i.e. the corrections are calculated in the space of sorted activations (both in $t_{(i)}$ and $a_{(i)}^{(m)}$), and are then added to the correct indices $\mathbf{j}$ within $\mathbf{a}^{(m)}$. Based on $\tilde{\mathbf{a}}$, the data term update is calculated according to Eqn. (3). The non-parametric Wasserstein correction layer (NP-C) then iteratively minimizes first $\mathscr{R}(\tilde{\mathbf{a}})$ and $\mathscr{D}(\mathbf{a}|\tilde{\mathbf{a}})$. Experiments have shown that one iteration is sufficient as the domain adaption is performed over all layers of the model. Since the correction layer does not need to be optimized, it can easily be added to pre-trained models, requiring only the collection of training dataset statistics. More details on non-parametric Wasserstein domain adaptation are provided in [19].

4 Experiments and Results

4.1 Data

We use the TAU Urban Acoustic Scenes 2020 Mobile development dataset [21]. It contains ten acoustic scenes, recorded across ten European cities. The samples were recorded with a total of nine different recording devices. Three of these are real and six are simulated. The dataset includes an official train-test split. In the test set the recording devices are evenly balanced, while in the training set the devices are greatly unbalanced, with device A contributing a portion of 62.5 %. Additionally, three of the simulated recording devices are only present in the test set. The training set contains 13 962 samples and the test set contains 2970 samples. Each audio sample is 10 s long,

ASCMobConvNet	Test accuracy
Baseline	$47.70 \pm 0.90\%$
BN	$66.07 \pm 0.86\%$
BN + RN	67.83 ± 0.86 %
SSN + RN	$68.10 \pm 0.86\%$
SSN + RN + NP-C	$68.78 \pm 0.85\%$

Table 1: Classification results for various normalizations and non-parametric wasserstein correction (NP-C).

uses a sampling rate of 44.1 kHz, and a bit width of 24. We further split the training set using a train-validation split of 85 % to 15 %.

We convert each audio sample to a log-melspectrogram with 256 bins. For the log-melspectrograms we use $f_{min} = 20Hz$ and $f_{max} = \frac{44.1kHz}{2}$. Additionally, during training we randomly roll the input feature tensors along the time dimension and apply the MixUp [23] data augmentation method. For the time-rolling operation we randomly draw a time within the interval $\{-5s, 5s\}$. For the MixUp augmentation method we set the parameter $\alpha_{MixUp} = 0.3$. Each augmentation method is applied on-the-fly with a probability of 75 %.

4.2 Setup

For training we use a batch size of 32, training the model for 120 epochs maximum. Here, an early stopping mechanism is employed that stops the training if the validation loss does not decrease for more than eight epochs. We use the Adam optimizer with a starting learning rate of 0.001. The learning rate is divided by a factor of 2 if the validation loss does not decrease over the course of five epochs. In between epochs the training set is shuffled.

We conduct an ablation study using four different settings. Here, only the fourth setting includes the non-parametric correction layers (NP-C). The baseline performance in our ablation study is derived from the baseline performance of the DCASE challenge 2022, task 1.A [24]. In the first setting we remove the SSN layers and replace them with plain BN instead, as well as deactivating the RN layers in favor of classic zero-mean unit variance standardization (BN). In the second setting we keep the BN layers but use the RN layers in the network instead of zero-mean unit variance standardization (BN+RN), and in the third setting we combine SSN layers with the RN layers (SSN+RN).

Lastly, in the fourth setting, we add the NP-C layer for test-time domain adaptation (SSN+RN+NP-C). For the Wasserstein correction parameters λ_1 and λ_2 we conduct a grid search. For each parameter set in the grid $\lambda_i = \{0.1, 0.2, ..., 0.9\}$ we take the model without Wasserstein correction layers, enable these layers and collect the activation distribution statistics $t_{(i)}$ from the training data. Afterwards, we evaluate the models using the corrected activation distribution for the test-samples.

4.3 Results and Discussion

In Table 1 the results for the model architectures are shown. Using RN over the frequency axis instead of zero-mean unit variance normalization improved the test accuracy by 1.76 %. Replacing the BN layers with SSN layers further improved the test accuracy by 0.27 %.

In Figure 3 we can see the absolute performance change

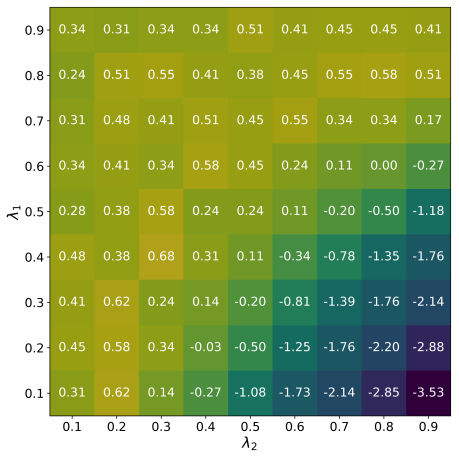

Figure 3: The improvement in test accuracy for each combination of λ_1 and λ_2.

of the Wasserstein correction for all λ_1, λ_2 combinations. The best performance is achieved with $\lambda_1 = 0.4$ and $\lambda_2 = 0.3$. It leads to an accuracy improvement of 0.68 %. This increases the test accuracy to 68.78 %. Setting λ_1 similar to λ_2 improves the accuracy. Low values of λ_1 combined with high values of λ_2 decrease the test accuracy. This underlines the assumption that putting too much emphasis on the prior and too little emphasis on the likelihood in Eqn. (2) distorts the spatial correlation between layers. The combination of low values of λ_1 and high value of λ_2, leads to a performance improvement.

Even though the Wasserstein correction improves the performance of our network its practical relevance l is limited. The Wasserstein correction layers add a massive amount of parameters to the network (i.e. the target distributions $\mathbf{t}$) without leading to a huge performance increase.

5 Conclusion

In this paper, we examined the Wasserstein correction algorithm for distribution shift correction in ASC. We specified a parameter efficient ASCMobConvNet model for the prediction of acoustic scenes using the TAU Urban Acoustic Scenes 2020 Mobile development dataset. Adding Wasserstein correction layers to our network improved the performance when a recording device mismatch across the train and the test datasets is present. We also observed that there is a sweet spot of hyperparameters for the Wasserstein correction, and that putting too much emphasis on the prior and too little emphasis on the likelihood of the correction algorithm degrades its performance. Using subspectral normalization layers and residual normalization layers instead of batch normalization and applying Wasserstein correction layers we were able to achieve an improvement in test accuracy of 2.71 % (absolute) for ASCMobConvNet, and a 21.08 %(absolute) improvement over the baseline model.

6 Acknowledgements

The financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

[1] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks," *CoRR*, vol. abs/2105.12395, 2021.

[2] B. Kim, S. Yang, J. Kim, and S. Chang, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," tech. rep., DCASE2021 Challenge, June 2021.

[3] Y.-F. Shao, X. Zhang, G.-G. Bing, K.-M. Zhao, J.-J. Xu, Y. Ma, and W.-Q. Zhang, "Mini-segnet for low-complexity acoustic scene classification," tech. rep., DCASE2022 Challenge, June 2022.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[5] R. Sugahara, R. Sato, M. Osawa, Y. Yuno, and C. Haruta, "Self-ensemble with multi-task learning for low-complexity acoustic scene classification," tech. rep., DCASE2022 Challenge, June 2022.

[6] H. Chen, Z. Liu, Z. Liu, P. Zhang, and Y. Yan, "Integrating the data augmentation scheme with various classifiers for acoustic scene modeling," tech. rep., DCASE2019 Challenge, June 2019.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[9] M. Kosmider, "Spectrum correction: Acoustic scene classification with mismatched recording devices," *CoRR*, vol. abs/2105.11856, 2021.

[10] T. Nguyen, F. Pernkopf, and M. Kosmider, "Acoustic scene classification for mismatched recording devices using heated-up softmax and spectrum correction," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 126–130, 2020.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[14] K. Drossos, P. Magron, and T. Virtanen, "Unsupervised adversarial domain adaptation based on the wasserstein distance for acoustic scene classification," 2019.

[15] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, "Improving robustness against common corruptions by covariate shift adaptation," *Advances in Neural Information Processing Systems*.

[16] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon, "Revisiting batch normalization for improving corruption robustness," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 494–503, 1 2021.

[17] S. Chang, H. Park, J. Cho, H. Park, S. Yun, and K. Hwang, "Subspectral normalization for neural audio data processing," *CoRR*, vol. abs/2103.13620, 2021.

[18] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[19] A. Fuchs, C. Knoll, and F. Pernkopf, "Distribution mismatch correction for improved robustness in deep neural networks," *CoRR*, vol. abs/2110.01955, 2021.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, June 2018.

[21] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, pp. 56–60, 2020.

[22] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks,"

[23] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017.

[24] I. Martin-Morato, T. Heittola, A. Mesaros, and T. Virtanen, "Low-complexity acoustic scene classification for multi-device audio: analysis of dcase 2021 challenge systems," 2021.