AN ENSEMBLE APPROACH TO ANOMALOUS SOUND DETECTION BASED ON CONFORMER-BASED AUTOENCODER AND BINARY CLASSIFIER INCORPORATED WITH METRIC LEARNING

Ibuki Kuroyanagi[1,2], *Tomoki Hayashi*[1,2], *Yusuke Adachi*[1,2], *Takenori Yoshimura*[1], *Kazuya Takeda*[2], *Tomoki Toda*[2]

[1] Human Dataware Lab. Co., Ltd., Nagoya, Japan
tomoki.hayashi@hdwlab.co.jp
[2] Nagoya University, Nagoya, Japan
kuroyanagi.ibuki@g.sp.m.is.nagoya-u.ac.jp

ABSTRACT

This paper presents an ensemble approach based on two unsupervised anomalous sound detection (ASD) methods for machine condition monitoring under domain-shifted conditions in DCASE 2021 challenge Task 2. The first ASD method is based on a conformer-based sequence-level autoencoder with section ID regression and a self-attention architecture. We utilize the data augmentation techniques such as SpecAugment to boost the performance and combine a simple scorer module for each section and each domain to address the domain shift problem. The second ASD method is based on a binary classification model using metric learning that uses task-irrelevant outliers as pseudo-anomalous data while controlling centroids of normal and outlier data in a feature space. As a countermeasure against the domain shift problem, we perform data augmentation based on Mixup with data from the target domain, resulting in a stable performance for each section. An ensemble approach is applied to each method, and the resulting two ensembled methods are further ensembled to maximize the ASD performance. The results of DCASE 2021 challenge Task 2 have demonstrated that our proposed method achieves a harmonic mean of 63.745% of area under the curve (AUC) and partial AUC ($p = 0.1$) over all machines, sections, and domains.

Index Terms— Anomalous sound detection, autoencoder, binary classification, metric learning

1. INTRODUCTION

Anomalous sound detection (ASD) is the task of detecting anomalous sounds caused by atypical events, such as the malfunction or breakdown of a machine. The detection of anomalous sounds can be used to improve the efficiency of maintenance work on manufacturing equipment and infrastructure, and to monitor equipment installed in locations which are difficult for people to enter. The use of ASD technology is expected to become widespread during the coming fourth industrial revolution, in applications such as factory automation utilizing artificial intelligence [1].

When training ASD models, it would be difficult to collect data representing every possible anomalous sound that could occur, because these sounds rarely occur during the normal operation of factory equipment, and the types of anomalous sounds which are possible are very diverse. Therefore, it is desirable to train ASD models without using anomalous data. In addition, real-world environments are often complicated and different conditions than those foreseen during the training of the ASD models may be encountered. Therefore, it is desirable to develop models that can detect anomalous sounds even when the normal state distribution is changed (i.e., after domain shift).

The two main approaches that have been proposed for performing ASD are generative methods and classification methods. Generative methods use only the normal data of a target sound to model its probability distribution and detect data that does not correspond to the model, categorized as anomalous. As a result of advances in deep learning technology, typical generative methods now involve the training of autoencoders (AE) to reconstruct normal data and calculate reconstruction error, or the use of autoregressive models with recursive neural networks to calculate the model likelihood, which is then used as an anomaly score [2, 3, 4, 5]. On the other hand, more recently developed classification methods distinguish between normal and outlier data by calculating anomaly scores based on distance from a decision boundary [6, 7, 8, 9], which have attracted much attention. Normal data from the operation of different machines is often used as outlier data during training. This method assumes that anomalous data is distributed outside the normal data, and that the outlier data is distributed even further outside the normal data. Based on this assumption, a binary classifier is trained using the normal data as positive examples, and the outlier data as pseudo-negative examples. Although generative and classification methods are both able to achieve good performance, they are unable to resolve the domain shift problem because the training and test data are recorded in the same environment.

Therefore, in this paper we propose an ASD method which is an ensemble of an autoencoder and a binary classification model, allowing it to function well even under domain shift conditions. The first component is a conformer-based, sequence-level autoencoder with section ID regression and a self-attention architecture [3]. We then utilize data augmentation techniques such as SpecAugment [10] to boost performance, and add a simple scorer module to each section and each domain to address the domain shift problem. The second component of our ASD method is a binary classification model employing metric learning, that uses task-irrelevant outliers as pseudo-anomalous data while controlling the centroids of normal and outlier data in a feature space [9]. As a countermeasure against the domain shift problem, we also perform data augmentation based on Mixup [11] using data from the target domain, resulting in stable performance for each data section. An ensemble approach is applied to each of the two components of our method, and the resulting ensemble methods are then ensembled to maximize ASD performance. We conduct our experimental evaluation using the DCASE 2021 Challenge Task 2 dataset.

2. BASELINE METHODS

This section provides an overview of methods that achieved high ASD performance when using the DCASE 2020 Task 2 data [12]. The Task 2 datasets contain recordings of six types of machines, and each set of audio data for each type of machine consists of seven or eight different machines of that type. ID information is provided to indicate which machine the audio data belonged to.

2.1. Conformer-based autoencoder [3]

The autoencoder method [3] assumes that an autoencoder would not be able to accurately reconstruct anomalous data, i.e., data other than normal data used to train the autoencoder. We assume that the input of an autoencoder at frame t is $\mathbf{x}_t$, and that the corresponding output is $\hat{\mathbf{x}}_t$. Reconstruction error e_t can be computed as follows:

$$e_t = \mathrm{abs}(\hat{\mathbf{x}}_t - \mathbf{x}_t), \tag{1}$$

where $\mathrm{abs}(\cdot)$ denotes an element-wise absolute operator. If $\mathbf{x}_t$ contains anomalous data, the norm of e_t should be large. Thus, anomaly detection can be performed by simple thresholding.

In this paper, we use a conformer [13] as an autoencoder. ID regression is used to accurately detect anomalous sounds combined with the sound of the target machine. We concatenate the integer machine ID to the input features, and the autoencoder then reconstructs the input acoustic features and the machine ID. The autoencoder tends to misidentify the machine ID when the audio clip includes anomalous sound, even if we provide the correct machine ID as an input. Therefore, we can detect whether the audio clip includes anomalous sound from the estimated machine ID. In addition, we modify $e_{1:T}$ based on the distribution of the reconstruction error to improve detection accuracy. Specifically, frame-level anomaly score, a_t, represent the negative likelihood of a Gaussian mixture model (GMM) consisting of K-mixture Gaussians for e_t:

$$a_t = -\sum_{k=1}^{K} w_k \mathcal{N}(e_t \,|\, \mu_k, \mathbf{\Sigma}_k), \tag{2}$$

where w_k, μ_k, and $\mathbf{\Sigma}_k$ are the weight, mean vector, and covariance matrix of the k th mixture component, respectively. For training the parameters, we use the reconstruction error calculated from the validation set, which is not used for training the autoencoder, but is randomly selected $10\,\%$ from the development data set. The frame-by-frame anomaly scores obtained by the GMM are aggregated into the final anomaly scores by removing some outlier data and using the softmax weighted average, since some of the lowest or highest negative likelihood may have adversely affected the anomaly scores. The aggregated anomaly score $\hat{a}$ is given by:

$$\hat{a} = \frac{1}{T'} \sum_{i=1}^{T'} a_i^{(\mathcal{M})} \frac{\exp(\alpha a_i^{(\mathcal{M})})}{\sum_{i=1}^{T'} \exp(\alpha a_i^{(\mathcal{M})})}, \tag{3}$$

where $a_i^{(\mathcal{M})}$ represents the frame-by-frame anomaly scores selected from a_i, T' is the number of selected frames, and α is a scalar hyperparameter.

2.2. Binary classifier with metric learning [9]

A binary classifier is trained using normal data as positive examples and with outlier data as pseudo-negative examples. We perform learning for each particular machine ID. The normal data for a particular target machine ID is used as the normal data, and the normal data of other machine IDs of the same machine type, as well as the normal data of all of the other machines in the same dataset, are used as outlier data.

Consider a set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ that has N samples of normal and outlier data. Normal and outlier data sets are assigned labels $y_i \in \{+1, -1\}$ $(i = 1, 2, ..., N)$ for each data sample. When performing ASD using a method based on binary classification, the network is trained to minimize the following binary cross-entropy (BCE) loss function:

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \Big\{ u(y_i) \log{(p_i)} + (1 - u(y_i)) \log(1 - p_i) \Big\}, \tag{4}$$

where p represents the posterior probabilities output of network ϕ_p (e.g., $p = \phi_p(\mathbf{x})$), which minimizes (4) when $\mathbf{x}$ is used as input, and where $u(y)$ is a binary function that takes 1 for $y > 0$ and 0 for $y \leq 0$. To further improve binary classification, we use the Deep Double-Centroids Semi-supervised Anomaly Detection (DDCSAD) loss function proposed in [9], which considers the centroids of both the normal and outlier data. The objective of the DDCSAD loss function is to minimize intra-class variance and maximize inter-class variance. The DDCSAD loss function is calculated as follows:

$$\mathcal{L}_{\mathrm{DDCSAD}} = \frac{1}{N} \sum_{i=1}^{N} \Big\{ \parallel \mathbf{z}_i - \mathbf{c}_p \parallel^{2y_i} + \parallel \mathbf{z}_i - \mathbf{c}_n \parallel^{-2y_i} \Big\}, \tag{5}$$

where $\mathbf{z}$ is the embedding vector output by encoder network ϕ_z (e.g., $\mathbf{z} = \phi_z(\mathbf{x})$), which minimizes (5) when $\mathbf{x}$ is used as the input, and where $\mathbf{c}_p \in \mathbb{R}^D$ and $\mathbf{c}_n \in \mathbb{R}^D$ represent the centroid of the normal and outlier data, respectively. Note that the initial values of centroids $\mathbf{c}_p$ and $\mathbf{c}_n$ are calculated using randomly initialized parameters, and are then recalculated at each epoch using the entire training data set. They are updated at each epoch by recalculating the centroids using the entire training data set. The following equation expresses the final loss function:

$$\mathcal{L} = \mathcal{L}_{\mathrm{BCE}} + \lambda \mathcal{L}_{\mathrm{DDCSAD}}, \tag{6}$$

where $\lambda > 0$ is a hyperparameter that controls the balance between the loss functions. Multi-task learning using both the cross-entropy of the posterior probability and the DDCSAD loss function increases accuracy when learning the decision boundaries, resulting in more accurate ASD.

During inference, posterior probability p, and distance $d = \parallel \mathbf{z} - \mathbf{c}_p \parallel^2$ between embedding vector $\mathbf{z}$ and centroid $\mathbf{c}_p$ of the normal class, are used to obtain the anomaly score. First, we compute distance d across the entire set of evaluation data, and then calculate standardized distance d' across the entire dataset. Finally, anomaly score s is calculated using the following equation:

$$s = \gamma \times (1 - p) + (1 - \gamma) \times d', \tag{7}$$

where, γ is a hyperparameter that determines the proportion of anomaly scores using posterior probability p.

3. PROPOSED METHODS

This section describes our proposed ASD methods working under domain shift conditions. We use section ID instead of machine ID due to the dataset change, but they have almost the same meaning.

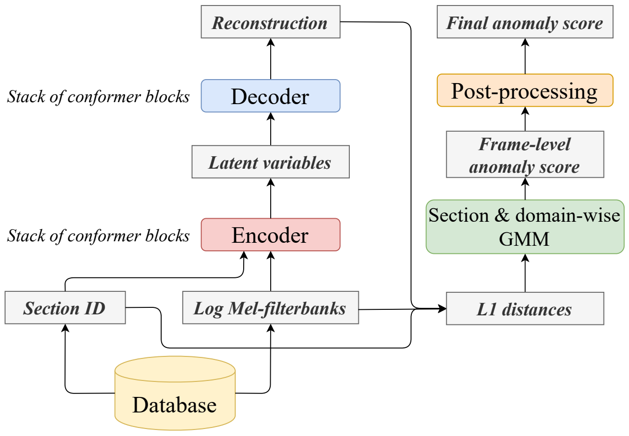

Figure 1: *Overview of proposed autoencoder method.*

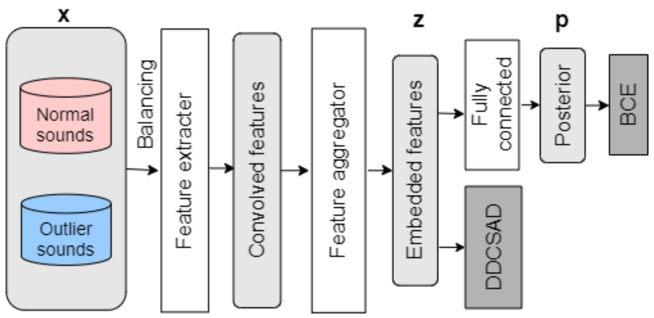

Figure 2: *Overview of proposed binary classification method.*

3.1. Ensembled, comformer-based autoencoder

The first component of our method is a sequence-level autoencoder with ID regression, which is based on the method described in Section 2.1. An overview is shown in Fig. 1. To boost the autoencoder's performance, we utilize SpecAugment [10] and dropout for the input feature sequence. Inspired by the interpolation deep neural network approach proposed in [14], we apply SpecAugment and dropout for the input feature sequence not only during training but also during inference, where we replicate the input sequence and apply different masking for each sequence. We then calculate the reconstruction error for each sequence and integrate the results using a pooling operation (e.g., average or max). This allows us to obtain the gain by using an ensemble model, even when using a single model. To address the domain shift problem, we build separate reconstruction error scoring modules for each section and each domain. This method is enabling us to capture differences in the anomaly score range between the sections and domains.

To further improve performance, we ensemble the model by selecting the N-best models when using the development data for each machine and each domain, and then integrate the outputs of these models to obtain the final score. We normalize the outputs and combine the normalized scores using the following four methods: *average*, *median*, *maximum*, and *ranking* which converts the scores into a rank and then calculates the average of the rankings.

3.2. Ensembled, binary classifier with metric learning

The second component of our proposed method is a binary classification model using metric learning, based on the method described in Section 2.2. An overview is shown in Fig. 2. We build the model for the source domain, and then perform fine-tuning for the target domain. When training for the source domain, only data from the source domain is used. On the other hand, when fine-tuning the model for the target domain, we create pseudo-target domain data using Mixup [11], using source and target domain data. It is expected that the use of Mixup will increase variation within each class and create data with an intermediate representation between the positive and negative examples.

The performance of the binary classification method is less stable than that of the autoencoder method, since the binary classification method builds a different model for each section of data. Therefore, we create many models in order to improve the performance of the model ensemble; for example, we change the method of pseudo-anomalous example selection, introduce additional data augmentation using methods such as Gaussian Noise and Volume control, change the architecture of the feature extraction module, and use an additional loss function (ArcFace [15]). ArcFace is a loss function used to achieve a clear geometric interpretation within a feature space, and the combination of ArcFace with DDCSAD results in further improvement in ASD performance. During pseudo-anomalous example selection, we also select samples from within the same dataset, resulting in more stable performance across the data sections. Finally, we use various different models as our feature extraction module, including ResNet34 [16], ResNeXt50 [17] and EfficientNet b3 [18] in PyTorch Image Models [19]. We average $N \times S$ models for each machine and each domain, where S represents the number of sections in the validation set.

3.3. Ensemble of the autoencoder and binary classifier models

We normalize the anomaly score of each of the two ensembled models to mean = 0 and variance = 1, and average the normalized scores.

4. EXPERIMENTAL EVALUATION

4.1. Experimental conditions

We conducted our experimental evaluation using the DCASE 2021 Challenge Task 2 dataset [20, 21]. The dataset consists of the normal and anomalous operating sounds of seven types of real machines: *ToyCar*, *ToyTrain*, *fan*, *gearbox*, *pump*, *slider*, and *valve*. Data for each type of machine includes six sections. Each section is further divided into two domains, containing source and target data, respectively. Each recording is a single-channel, 10 second segment of audio sampled at 16 kHz. The training data includes 1,000 samples in the source domain and only 3 samples in the target domain. The development and evaluation data consists of around 200 samples in each domain. The training data includes only normal sounds, but the development data includes both normal and anomalous sounds to allow the evaluation of anomaly detection performance. To evaluate the performance of our proposed method, we included the following models in our experiment for comparison:

Baseline (AE): The official autoencoder-based baseline method [22], trained with normal training data, which minimizes reconstruction error.

Baseline (MNV2): The official classification-based baseline method, which uses MobileNetV2 [22] trained using section ID classification.

Our baseline (AE): Our baseline, sequence-level autoencoder

Table 1: *Evaluation results. Values represent the harmonic mean of AUC* [%] *and pAUC (*$p = 0.1$*)* [%] *for each section of each domain. "All/har-mean" column values represent the harmonic mean of AUC and pAUC over all machines, sections and domains.*

		ToyCar		ToyTrain		fan		gearbox		pump		silder		valve		All
	Method	source	target	source	target	source	target	source	target	source	target	source	target	source	target	har-mean
dev	Baseline (AE)	59.44	54.74	64.31	51.99	59.14	56.72	56.42	61.04	63.85	53.01	67.09	55.71	52.43	51.45	57.28
	Baseline (MNV2)	57.19	55.89	58.81	50.77	63.31	61.58	65.54	60.72	62.20	57.36	65.43	52.17	53.99	55.17	58.22
	Our baseline (AE)	80.41	63.05	80.50	61.46	71.82	66.35	62.69	70.01	72.61	62.41	86.04	62.01	80.60	64.30	69.05
	Our baseline (BC)	57.91	58.68	76.23	49.04	67.36	59.36	**74.85**	74.59	72.12	59.86	80.64	57.24	86.18	70.10	69.08
	AE ens	**83.29**	68.70	**81.06**	**62.83**	74.37	69.75	64.14	72.32	74.60	**65.68**	86.12	65.41	82.94	67.81	71.67
	BC ens	60.93	64.55	76.16	55.40	**82.29**	66.62	74.49	70.58	75.20	60.70	**89.28**	57.69	92.70	79.35	73.29
	AE+BC ens (mix)	79.90	**70.08**	80.23	59.85	82.25	**71.58**	72.95	**76.25**	**77.29**	64.03	89.06	**68.49**	**93.03**	**80.15**	**76.59**
	AE+BC ens (max)	**83.29**	68.70	**81.16**	**62.83**	**82.29**	69.75	74.49	72.32	75.20	**65.68**	**89.28**	65.41	92.70	79.35	75.68
eval	Baseline (AE)	61.33	55.63	61.86	**63.26**	57.99	52.54	61.17	**60.58**	56.38	52.55	56.76	51.78	51.22	50.69	56.38
	Baseline (MNV2)	41.81	**57.59**	49.76	43.50	63.65	59.24	53.31	49.55	63.79	64.00	66.17	66.34	53.86	50.86	54.77
	AE ens	54.94	54.95	**65.84**	54.82	62.84	59.00	**66.18**	60.31	58.13	62.14	71.45	62.49	63.30	49.52	59.92
	BC ens	**64.39**	55.07	54.86	52.90	65.97	**63.70**	55.91	50.44	**81.40**	**79.86**	**84.22**	75.69	66.23	**58.49**	63.21
	AE+BC ens (mix)	60.83	56.30	64.64	54.84	**70.01**	63.12	60.93	56.21	70.46	64.84	82.26	**77.36**	**68.78**	55.10	**63.75**
	AE+BC ens (max)	54.94	54.95	**65.84**	54.82	65.97	59.00	55.91	60.31	**81.40**	62.14	**84.22**	62.49	66.23	**58.49**	62.26

model, trained for 50,000 steps using the Adam optimizer [23] with Warmup scheduler [24]. The batch size was set to 64 and the number of warmup steps was 8,000. The hyperparameters were optimized for each machine and each domain, including Mel-spectrogram extraction condition (e.g., shift size and Mel basis), model architecture (e.g., the number of blocks, units and kernel size) and post-processing. In SpecAugment, the number of time masks was set to 50, with a width range from one to five, while the number of frequency masks was set to five, with a width range from zero to ten. The dropout rate for the input sequence was set to 0.2.

Our baseline (BC): Our baseline, binary classification-based model was ResNet34 [16]. The size of the spectrogram was 256×256. The model was trained for 8,000 steps using the Adam optimizer, with a learning rate for the fully-connected layer of 1.0e-3, and a learning rate for the convolution layer of 5.0e-4. The OneCycleLR scheduler [25] was used, and the batch size was 64. The ratio of normal to outlier data was set to 1:1 in the mini-batch. When fine-tuning for the target domain, we trained the pre-trained model created using source domain data for 800 steps. Sampling was performed during fine-tuning so that the mini-batch always contained 16 samples of target domain or pseudo-target domain data, which was obtained by mixing up data from the target and source domains.

AE ens: An ensemble of the proposed autoencoder models. The value of N was selected from among 3, 5, 10 and 20, and the ensemble methods were optimized for each machine and each domain.

BC ens: An ensemble (average) of the proposed binary-classification models. The value of N was set to two.

AE+BC ens (mix): The average of AE ens and BC ens.

AE+BC ens (max): The ensemble of AE ens and BC ens. We took the maximum output value between AE ens and BC ens for each machine and each domain.

The hyperparameters and post-processing parameters of each model were optimized for each section and each domain.

4.2. Experimental results

Our experimental results are shown in Table 1. First, we focus on the results when using the development data. When comparing the performance of our baseline (AE) and AE ens, and our baseline (BC) and BC ens, we can see that performance of harmonic mean generally improved. When comparing the results when using our two baseline methods, we can see that AE outperformed BC for machine types *ToyCar* and *ToyTrain*, while BC outperformed AE for machine types *gearbox* and *valve*, and further improvements are yielded by ensembling these two methods, suggesting that each of them focuses on different features of each machine type.

Next, we focus on performance when using the evaluation data. BC ens outperformed AE ens for machine types *pump*, *slider* and *valve*, regardless of the domain. In these types of machines, where the sound generated is non-stationary (i.e., it includes a variety of intermittent sounds, such as clicks), the BC-based method was found to be superior. Unlike our results when using the development data, ASD performance for *ToyCar* when using AE ens decreased. These results suggest that model performance tends to be influenced not only by machine type, but also by the type of anomalous sound that is present. Finally, we found that all of the proposed methods outperformed all of the baseline methods with both datasets, and that the AE+BC ens (mix) model achieved the best ASD performance.

These results demonstrated that our proposed method performed well under domain shift conditions. Furthermore, we found that using an ensemble of the results from different ASD models focusing on different features contributes to score improvement, since the outputs of the models complement each other.

5. CONCLUSION

In this paper, we presented an ensemble ASD approach using both a conformer-based autoencoder and a binary classification model with metric learning. Our experimental evaluation showed that the proposed methods significantly outperformed the baseline methods by achieving higher ASD scores. We also demonstrated that by using an ensemble of completely different ASD methods, we were able to obtain better performance. These results suggest that different ASD methods focus on different audio data features to detect anomalous sounds, so it is important to ensemble models that can pick out different features. In future work, we will develop a method that can obtain the better ASD performance using fewer models.

6. ACKNOWLEDGMENT

This paper was partly supported by a project, JPNP20006, commissioned by NEDO.

7. REFERENCES

[1] B. Bayram, T. B. Duman, and G. Ince, “Real time detection of acoustic anomalies in industrial processes using sequential autoencoders,” *Expert Systems*, vol. 38, no. 1, p. e12564, 2021.

[2] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, “Anomalous Sound Detection Based on Interpolation Deep Neural Network,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 271–275.

[3] T. Hayashi, T. Yoshimura, and Y. Adachi, “Conformer-based id-aware autoencoder for unsupervised anomalous sound detection,” DCASE2020 Challenge, Tech. Rep., July 2020.

[4] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long Short Term Memory Networks for Anomaly Detection in Time Series,” in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[5] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara, “Latent Space Autoregression for Novelty Detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] P. Primus, V. Haunschmid, P. Praher, and G. Widmer, “Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 170–174.

[7] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, “Deep Semi-Supervised Anomaly Detection,” in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HkgH0TEYwH

[8] L. Ruff, R. A. Vandermeulen, B. J. Franks, K.-R. Müller, and M. Kloft, “Rethinking Assumptions in Deep Anomaly Detection,” in *ICML 2021 Workshop on Uncertainty & Robustness in Deep Learning*, 2021.

[9] I. Kuroyanagi, T. Hayashi, K. Takeda, and T. Toda, “Anomalous sound detection using a binary classification model and class centroids,” in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1995–1999.

[10] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *Proceedings of Interspeech 2019*, 2019, pp. 2613–2617.

[11] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[12] Y. Koizumi *et al.*, “Description and discussion on dcase2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 81–85.

[13] A. Gulati *et al.*, “Conformer: Convolution-augmented Transformer for Speech Recognition,” in *Proc. Interspeech 2020*, 2020, pp. 5036–5040. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-3015

[14] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, “Anomalous sound detection based on interpolation deep neural network,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 271–275.

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[18] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, “Self-training with noisy student improves imagenet classification,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[19] R. Wightman, “PyTorch Image Models,” https://github.com/rwightman/pytorch-image-models, 2019.

[20] R. Tanabe, H. Purohit, K. Dohi, T. Endo, Y. Nikaido, T. Nakamura, and Y. Kawaguchi, “MIMII DUE: Sound dataset for malfunctioning industrial machine investigation and inspection with domain shifts due to changes in operational and environmental conditions,” *arXiv preprint arXiv:2006.05822*, 2021.

[21] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, “ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions,” *arXiv preprint arXiv:2106.02369*, 2021.

[22] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, “Description and discussion on DCASE2021 Challenge Task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions,” *arXiv preprint arXiv:2106.04492*, 2021.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of 3rd International Conference on Learning Representations, ICLR 2015*, 2015, pp. 1–15.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of 2017 Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[25] L. N. Smith and N. Topin, “Super-convergence: very fast training of neural networks using large learning rates,” in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, T. Pham, Ed., vol. 11006, International Society for Optics and Photonics. SPIE, 2019, pp. 369 – 386. [Online]. Available: https://doi.org/10.1117/12.2520589