ASDKit: A Toolkit for Comprehensive Evaluation of Anomalous Sound Detection Methods

Takuya Fujimura[1], *Kevin Wilkinghoff* [2,3], *Keisuke Imoto*[4], *Tomoki Toda*[1]

[1]Nagoya University, Japan, [2]Aalborg University, Denmark, [3]Pioneer Centre for AI, Denmark, [4]Kyoto University, Japan

***Abstract*—In this paper, we introduce ASDKit, a toolkit for anomalous sound detection (ASD) task. Our aim is to facilitate ASD research by providing an open-source framework that collects and carefully evaluates various ASD methods. First, ASDKit provides training and evaluation scripts for a wide range of ASD methods, all handled within a unified framework. For instance, it includes the autoencoder-based official DCASE baseline, representative discriminative methods, and self-supervised learning-based methods. Second, it supports comprehensive evaluation on the DCASE 2020–2024 datasets, enabling careful assessment of ASD performance, which is highly sensitive to factors such as datasets and random seeds. In our experiments, we re-evaluate various ASD methods using ASDKit and identify consistently effective techniques across multiple datasets and trials. We also demonstrate that ASDKit reproduces the state-of-the-art-level performance on the considered datasets.**

***Index Terms*—anomalous sound detection, open-source toolkit**

1. INTRODUCTION

Anomalous sound detection (ASD) is a technique for machine condition monitoring, where ASD systems aim to detect mechanical failures from their operational machine sounds [1]–[5]. In this task, since it is infeasible to exhaustively collect rare and diverse anomalous sounds, the system is developed with only normal machine sounds. Therefore, ASD systems calculate anomaly scores based on the deviation from the normal sound distribution and assign anomalies to sounds with high anomaly scores.

The ASD field has advanced through the development of various methods. One major approach is based on generative modeling [1], [6], [7], where it directly models the distribution of audio features belonging to normal machine sounds and computes anomaly scores based on the negative likelihood of a given observation. For example, an autoencoder (AE)-based method [1] trains an AE network to reconstruct audio features and computes anomaly scores based on the reconstruction error for a given observation. Other variants of AEs only reconstruct the center frame of consecutive spectrogram frames [6] or mask the input to obtain an autoregressive model [8]. Another recent approach first projects audio signals into a lower-dimensional feature space and then computing anomaly scores based on the distance of an observation from the normal training data samples in that space [9]–[12]. For example, state-of-the-art (SOTA) discriminative methods [9], [12]–[16] train a discriminative feature extractor to classify meta-information labels associated with normal training data, and then compute anomaly scores in the discriminative feature space. Possible choices of meta information include the machine type, machine IDs and specific operating conditions of machines. Alternatively, self-supervised learning (SSL) feature spaces of models trained on (large) external datasets are directly utilized for ASD [11], [17].

Although the proposal of various ASD methods has undoubtedly advanced the field, we argue that further development is hindered by the lack of comprehensive and careful evaluation. A representative effort for evaluating ASD methods is the DCASE challenge [1]–[5], which plays an important role in enabling fair evaluations using unrevealed test data. However, challenge evaluations are conducted on a single dataset and a single trial, which is insufficient for thorough assessment, as we have observed that ASD performance is highly sensitive to datasets and random seeds in our preliminary experiments. Furthermore, it is difficult to precisely identify the effectiveness of individual components, as participants develop methods on their own frameworks and often employ ensemble to boost performance.

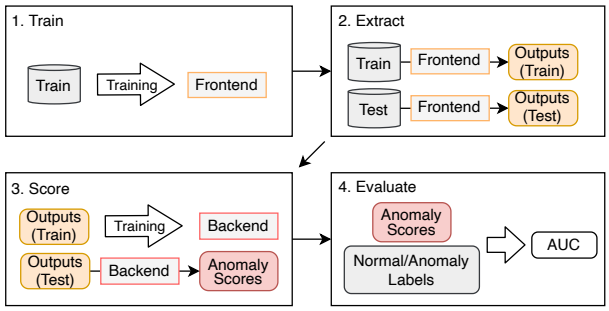

Fig. 1: Unified framework of ASDKit. (1) Training the frontend with the training dataset, (2) extracting and storing outputs from both the training and test datasets using the frontend, (3) computing anomaly scores for the test data using the backend trained on the training data, and (4) computing evaluation scores based on the anomaly scores and the ground-truth normal and anomalous labels.

To address these limitations, we introduce ASDKit[1], an open-source toolkit for the ASD task. The main features of ASDKit are summarized as follows: (1) ASDKit collects various ASD methods into a single open-source repository, enabling easy access and comparison. (2) ASDKit provides *recipes* that complete the entire training and evaluation process shown in Fig. 1. (3) ASDKit supports comprehensive evaluation on the DCASE 2020–2024 datasets [1]–[5] with multiple trials, enabling careful assessment. To demonstrate ASDKit's capabilities, we re-evaluate various ASD methods across multiple datasets and trials. By analyzing the results, we identify consistently effective techniques and provide new insights. Furthermore, we demonstrate that ASDKit can reproduce the SOTA-level performance on the considered datasets.

2. ASDKIT

2.1. Supported experimental conditions

ASDKit supports the DCASE 2020–2024 conditions [1]–[5], which are summarized in Table 1. Across these conditions, the machine types, the provided meta-information, and the evaluation scores differ. ASDKit provides download scripts and evaluation tools for these datasets. The download scripts automatically download the datasets and add ground-truth normal and anomalous labels to the test sets, for which the labels are concealed during the challenge and released by the organizers afterward. The evaluation tool computes the official evaluation scores according to the respective DCASE year. For all conditions, the datasets are divided into official *dev* and *eval* subsets, and evaluation scores are aggregated separately for each subset. The evaluation score is based on a combination of several types of area under the receiver operating characteristic (ROC) curve (AUC).

[1]https://github.com/TakuyaFujimura/dcase-asd-toolkit

Table 1: Supported experimental conditions in ASDKit. MT and Sec denote machine type and section, respectively. The "#ID/Sec" column shows the number of machine IDs or sections within each machine type. The "Dev/Eval" column shows the information used to define the dev and eval subsets. The "Aggregation" column shows how the evaluation results for each machine type and ID/Sec are aggregated to the machine-type level or the dev- and eval-subset level. $\mathrm{Amean}()$ and $\mathrm{Hmean}()$ are arithmetic mean and harmonic mean operations, respectively. *: the attribute information is not available for some machine types.

Year	#MT	#ID/Sec	Domain	Meta Information	Dev/Eval	Evaluation score for each MT and ID/Sec	Aggregation
2020 [1]	6	6 or 7	Source	MT, ID	ID	$\mathrm{Amean}(\mathrm{s\_auc}, \mathrm{s\_pauc})$	$\mathrm{Amean}()$
2021 [2]	7	6	Source, Target	MT, Sec, Attribute, Domain	Sec	$\mathrm{Hmean}(\mathrm{s\_auc}, \mathrm{s\_pauc}, \mathrm{t\_auc}, \mathrm{t\_pauc})$	$\mathrm{Hmean}()$
2022 [3]	7	6	Source, Target	MT, Sec, Attribute, Domain	Sec	$\mathrm{Hmean}(\mathrm{smix\_auc}, \mathrm{tmix\_auc}, \mathrm{mix\_pauc})$	$\mathrm{Hmean}()$
2023 [4]	14	1	Source, Target	MT, Sec, Attribute, Domain	MT	$\mathrm{Hmean}(\mathrm{smix\_auc}, \mathrm{tmix\_auc}, \mathrm{mix\_pauc})$	$\mathrm{Hmean}()$
2024 [5]	16	1	Source, Target	MT, Sec, Attribute*, Domain	MT	$\mathrm{Hmean}(\mathrm{smix\_auc}, \mathrm{tmix\_auc}, \mathrm{mix\_pauc})$	$\mathrm{Hmean}()$

Table 2: ASD methods supported by ASDKit.

Approach	Recipe name
AE	*ae*
Discriminative	*dis_spec_adacos_fixed*, *dis_beats_scac_trainable*, etc.
Raw feature	*raw_spec*, *raw_beats*, *raw_eat*

In DCASE 2020, the machine ID, which identifies each individual machine of the same machine type, is provided as meta information. The evaluation score is calculated as the arithmetic mean of the AUC and the partial AUC (pAUC) with $p = 0.1$. This score is computed for each machine ID within each machine type and then aggregated into dev- and eval-subset-level scores using the arithmetic mean. The dev and eval subsets are defined based on the machine ID [1].

DCASE 2021 newly introduces a domain shift problem, where recording environments or machine operations differ between the *source* and *target* domains [18]. While abundant training data is available in the source domain, only a few samples are available in the target domain. As meta information, *attributes* which denotes the recording and machine operation conditions, and the domain information are provided. The evaluation score is computed as the harmonic mean of $\mathrm{s\_auc}$, $\mathrm{s\_pauc}$, $\mathrm{t\_auc}$, and $\mathrm{t\_pauc}$. $\mathrm{s\_auc}$ and $\mathrm{s\_pauc}$ are the AUC and pAUC for the source domain, respectively, and $\mathrm{t\_auc}$ and $\mathrm{t\_pauc}$ are the AUC and pAUC for the target domain, respectively. Therefore, ASD systems are expected to perform well in both domains. Also, in DCASE 2021, the machine ID is replaced with the *section*. The section defines subsets within one machine type, and it serves a similar role to the machine ID; however, in some machine types, different machine IDs can appear in the same section, and the same machine ID can appear in multiple sections [2]. The evaluation scores for each section within each machine type, are aggregated to dev- and eval-subset-level scores using the harmonic mean.

DCASE 2022 inherits the domain shift problem from the DCASE 2021 condition, but employs a different evaluation score [3]. The score is calculated as the harmonic mean of $\mathrm{smix\_auc}$, $\mathrm{tmix\_auc}$, and $\mathrm{mix\_pauc}$. $\mathrm{smix\_auc}$ is the AUC calculated using normal and anomalous samples from the source domain together with anomalous samples from the target domain. Similarly, $\mathrm{tmix\_auc}$ is the AUC calculated using normal and anomalous samples from the target domain together with anomalous samples from the source domain. $\mathrm{mix\_pauc}$ is the pAUC calculated using normal and anomalous samples from both domains. These evaluation scores are calculated by jointly using samples from both domains to assess whether the anomaly score distributions are well aligned between domains.

In DCASE 2023 and 2024, only one section is provided, but a larger number of machine types are included. The dev and eval subsets are defined based on the machine types [4], [5]. In DCASE 2024, attribute information is not available for some machine types [5]. The evaluation score is the same as in DCASE 2022.

2.2. Supported methods

ASDKit handles various ASD methods in a unified framework shown in Fig. 1. The supported methods are summarized in Table 2. All ASD methods consist of *frontend* and *backend* modules, where the frontend extracts some features from audio signals, and the backend computes anomaly scores after processing these features. Training and evaluation are performed in four steps: (1) training the frontend with the training dataset, (2) extracting features from both the training and test datasets, (3) training the backend with the features extracted from the training dataset and computing anomaly scores for the test data, and (4) computing evaluation scores based on the anomaly scores and ground-truth normal and anomalous labels. In the following, we explain the ASD methods supported by ASDKit and their processing pipelines within this four-step framework.

2.2.1. AE: This is the AE-based official DCASE baseline method [1]. The frontend consists of a ten-layer multilayer perceptron (MLP). The input features are five adjacent frames of the log Mel power spectrogram, and the frontend is trained to minimize their reconstruction loss. AE directly computes anomaly scores based on the reconstruction error of encountered test samples in the second step, and in the third step, the backend simply copies these anomaly scores. The AE is independently trained for each machine type, repeating steps 1–4 for each machine type.

2.2.2. Discriminative methods: ASDKit provides various options for discriminative methods. For the frontend architecture, it supports multi-branch CNNs [9], [19] and SSL models [12], [20], [21]. The multi-branch CNN receives multiple input features, such as the spectrum and spectrograms. The frontend independently processes these features and obtains a final output by concatenating the outputs from each branch. Each branch consists of 1D or 2D convolutional layers followed by an MLP. For SSL models, ASDKit supports BEATs [20] and EAT [21], which are widely used in ASD tasks [11], [12], [22]–[24]. Based on previous works [12], we also introduce additional low-rank adaptation (LoRA) [25] parameters to fine-tune these models. These frontends are trained using classification of meta-information labels.

For the discriminative loss function, ASDKit supports ArcFace [26], AdaCos [27], Sub-cluster AdaCos (SCAC) [28], and AdaProj [29], where these angular margin loss functions are known to be effective for ASD tasks [30]. ASDKit also supports the option to choose between fixed and trainable class centers for the angular margin loss functions, as this choice affects performance; fixed class centers have been shown to achieve better results in previous work [9].

For data augmentation, ASDKit supports Mixup [31] and SpecAugment [32], which are widely used in ASD tasks [9]. Additionally, ASDKit supports techniques to boost ASD performance, such as FeatEx [10] and its parameter-efficient variant, subspace loss [19]. FeatEx and subspace loss introduce additional losses using subspace features in the multi-branch CNN, where subspace features refer to the features extracted from each branch before concatenation.

The backend consists of k-nearest neighbor (kNN), where anomaly scores are calculated as the average distance to the k nearest neighbors in the training (reference) dataset from a given observation in the discriminative feature space. ASDKit also supports variants that

Table 3: Setups of the discriminative methods. MB indicates multi branch CNN. Parentheses in the "Training strategy" column denote whether the class centers in the loss function are fixed or trainable.

Recipe Name	Frontend	Training strategy	DataAug
dis_spec_ adacos_fixed_wo_mixup	MB	AdaCos (fixed)	No
dis_spec_adacos_fixed	MB	AdaCos (fixed)	Mixup
dis_spec_scac_fixed	MB	SCAC (fixed)	Mixup
dis_spec_scac_trainable	MB	SCAC (trainable)	Mixup
dis_spec_subspaceloss	MB	Subspace loss	Mixup
dis_spec_featex	MB	FeatEx	Mixup
dis_multispec_scac_trainable	MB	SCAC (trainable)	Mixup
dis_beats_scac_trainable	BEATs w/ LoRA	SCAC (trainable)	Mixup
dis_eat_scac_trainable	EAT w/ LoRA	SCAC (trainable)	Mixup

incorporate kmeans clustering [9], SMOTE oversampling [23], [33], and anomaly score rescaling [34]. Kmeans clustering and SMOTE are used to balance the number of training samples between the source and target domains. Kmeans clustering reduces the number of samples in the source domain, and the resulting centroids are used as reference samples in the subsequent kNN-based anomaly score calculation. SMOTE is applied to oversample the training samples in the target domain. The anomaly score rescaling technique [34] rescales anomaly scores based on the local density of training samples in the feature space, addressing the tendency for low-density target domains to exhibit higher anomaly scores.

The discriminative methods train a common frontend using data from all machine types, and then independently train the backend for each machine type by repeating steps 2–4 for each type.

2.2.3. Raw feature-based methods: ASDKit supports raw feature-based ASD methods [11], [35], where the frontend extracts raw features without training with meta-information label classification. For example, as raw features, [35] uses the time-averaged spectrogram, and [11] uses BEATs features without fine-tuning. For the backend, the same kNN-based anomaly score calculation as in the discriminative methods is employed. This method repeats steps 2–4 for each machine type, skipping the frontend training in the first step.

3. EXPERIMENTAL EVALUATION

We re-evaluated various ASD methods using ASDKit to demonstrate its capabilities.

3.1. Setups

3.1.1. Datasets and metrics: We used the DCASE 2020–2024 datasets [1]–[5] and the official evaluation scores described in Sec. 2.1. We computed the arithmetic mean and standard deviation across four independent trials, where each trial's official score was calculated as the harmonic or arithmetic mean, as defined in Table 1.

3.1.2. Evaluated methods and their setups: In the following, we list the ASD methods evaluated in this paper and describe their setups. Each method is referred to by its recipe name provided in ASDKit.

ae is the AE-based method [1] described in Sec. 2.2.1. We trained the AE network for 50 epochs using the Adam optimizer [36] using the mean squared error as a loss function with a fixed learning rate of 0.001 and a batch size of 256. For the input features, we used five adjacent frames of the log Mel power spectrogram, with 128 Mel bins, a DFT size of 1024 samples, and a hop size of 512 samples.

The discriminative methods we evaluated are summarized in Table 3. Mixup [31] was always applied with a probability of 50%. *dis_multispec_scac_trainable* extracted 512-dimensional features from an amplitude spectrum and three spectrograms with different DFT sizes (256, 512, and 1024 samples), while the other multi-branch CNN methods extracted 256-dimensional features from an amplitude spectrum and an amplitude spectrogram with a DFT size of 1024 samples. The hop size was set to half the DFT size, and frequency bins in the range of 200 Hz to 8000 Hz were used. We trained the multi-branch CNNs for 16 epochs using AdamW optimizer [37] with a fixed learning rate of 0.001 and a batch size of 64. For *dis_beats_scac_trainable*, we used the *BEATs_iter3.pt* checkpoint and introduced LoRA parameters to the query and key projection layers within the Transformer. The 768-dimensional feature sequence from BEATs was aggregated using a statistics pooling layer [38] and then projected to a 256-dimensional feature using a linear layer. For *dis_eat_scac_trainable*, we used the *EAT-base_epoch10_pt.pt* checkpoint and introduced LoRA parameters to the query, key, and value projection layers. A 768-dimensional CLS feature from EAT was projected to a 256-dimensional feature using a linear layer. We fine-tuned the SSL-based models for 25 epochs using AdamW optimizer with a batch size of 8 and a LoRA rank of 64. The learning rate was linearly increased from 0 to 0.0001 over the first 5,000 steps.

For the raw feature-based methods described in Sec. 2.2.3, we evaluated *raw_spec*, *raw_beats*, and *raw_eat*. *raw_spec* used time-averaged Mel amplitude spectrograms with 128 Mel bins, a DFT size of 1024 samples, and a hop size of 512 samples. *raw_beats* averaged the 768-dimensional feature sequence from BEATs, while *raw_eat* directly used the 768-dimensional CLS feature from EAT.

For the discriminative methods and raw feature-based methods, we used a kNN-based backend with $k = 1$ described in Sec. 2.2.2. We also used kmeans clustering, SMOTE oversampling, and anomaly score rescaling techniques. For kmeans clustering, we set the number of clusters to 16. For SMOTE oversampling, we set the oversampling ratio to 20% and the number of neighbors to 2. For anomaly score rescaling [34], we used a kNN-based approach with the number of neighbors set to 4.

3.2. Results

Figure 2 shows the official evaluation scores of the considered ASD methods, where both the discriminative methods and raw feature-based methods employ kNN with SMOTE as the backend. The figure also includes the evaluation scores of the top-performing system and the official baselines reported by the DCASE organizers.[2] First, it is evident that evaluation on a single dataset and a single trial is insufficient. For example, the performance order of *dis_spec_adacos_fixed* and *dis_beats* is reversed between the DCASE 2023 dev and eval subsets. Also, *dis_spec_adacos_fixed* exhibits a standard deviation of 2.04 on the DCASE 2023 eval subset, with performance ranging from 53.07 to 57.98 across four trials, as observed in our experimental results.

We reaffirm that training techniques for discriminative methods significantly impact performance. For instance, the effectiveness of mixup is demonstrated by comparing *dis_spec_adacos_fixed_wo_mixup* and *dis_spec_adacos_fixed*; the effectiveness of SCAC, by comparing *dis_spec_adacos_fixed* and *dis_spec_scac_fixed*; and the effectiveness of multi-resolution spectrograms, by comparing

[2]The evaluation scores on the development set are not officially reported. In DCASE 2020, the official scores aggregated across all machine types were also not provided. Although several official baseline systems are provided, only the best-performing baseline systems are included in the figure. The best official baseline systems in DCASE 2021 [1] and 2024 [5] use the same method as our *ae* recipe, while the 2023 [4] baseline uses its variant [39]. The 2022 baseline employs a different discriminative method [3].

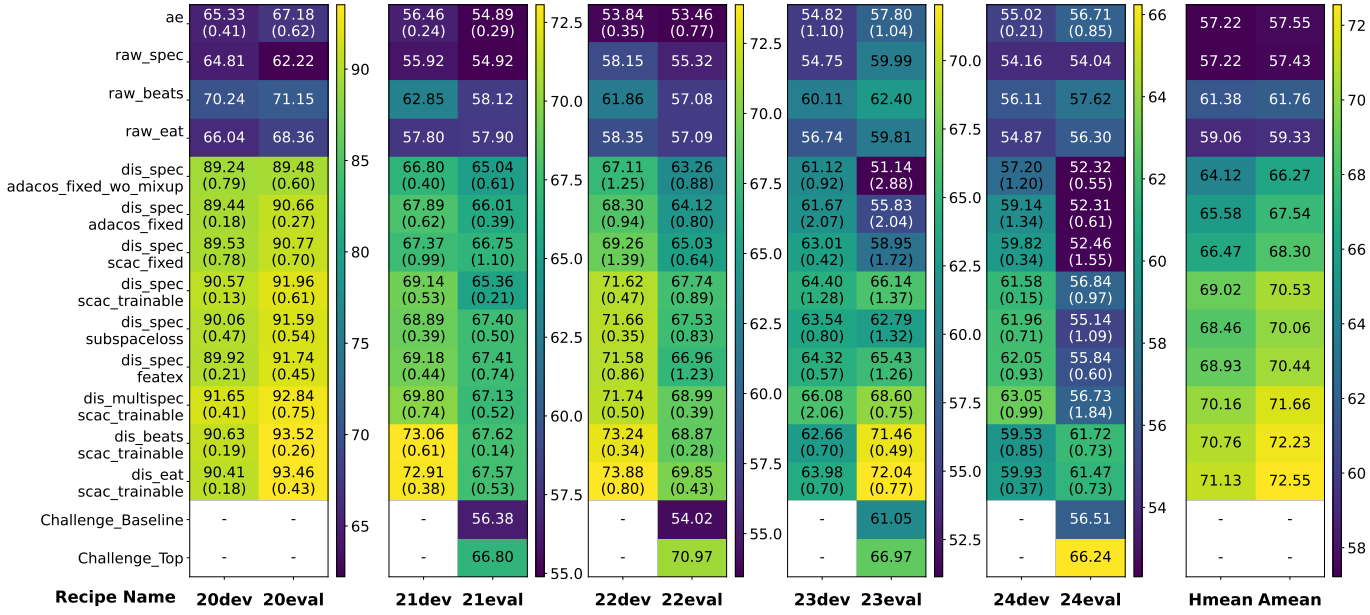

Fig. 2: Official evaluation scores. Values are presented as "arithmetic mean (standard deviation)" across four independent trials. Hmean and Amean denote the harmonic mean and arithmetic mean, respectively, computed over all dev and eval subsets across the datasets. Backend for the discriminative methods and raw feature-based methods is kNN with SMOTE. Raw feature-based methods do not involve any random processes; therefore, the standard deviation is not reported.

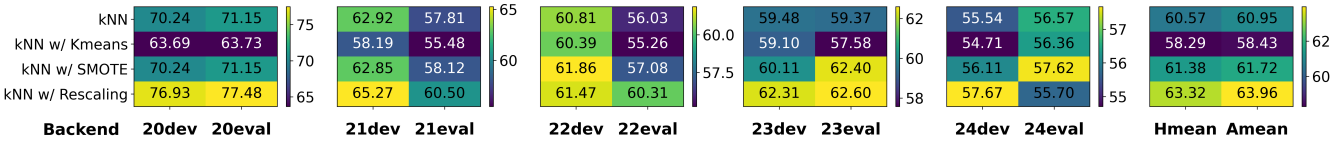

Fig. 3: Official evaluation scores of *raw_beats* with different backends.

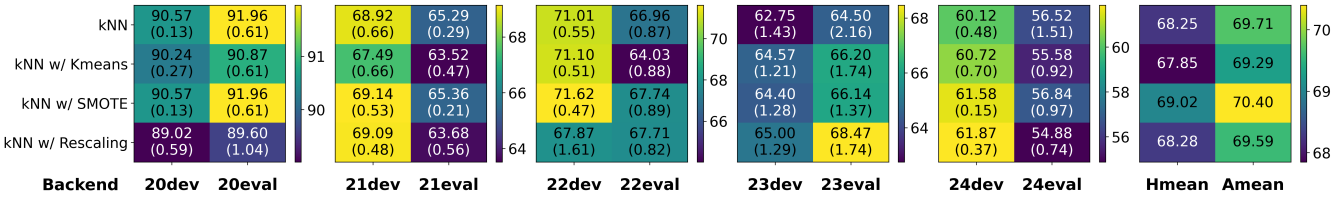

Fig. 4: Official evaluation scores of *dis_spec_scac_trainable* with different backends. Values are presented as "arithmetic mean (standard deviation)" across four independent trials.

dis_spec_scac_trainable and *dis_multispec_scac_trainable*. Additionally, fine-tuned SSL models (*dis_beats_scac_trainable* and *dis_eat_scac_trainable*) consistently achieve high performance.

As a new insight, we find that trainable class centers improve performance in most cases, as demonstrated by the comparison between *dis_spec_scac_fixed* and *dis_spec_scac_trainable*, while fixed class centers achieved better results in previous work [9]. Moreover, the FeatEx (*dis_spec_featex*) and subspace loss (*dis_spec_subspaceloss*) techniques achieve performance improvements similar to those obtained by using trainable class centers. Since both FeatEx and subspace loss employ trainable centers for additional loss terms, we conclude that their performance gains are mainly attributable to the use of trainable class centers.

Furthermore, Fig.2 demonstrates that ASDKit achieves SOTA-level performance. Note that the top-performing systems in DCASE 2021 [40], 2022 [41], and 2024 [42] employed ensembles, with the 2022 system additionally using machine-specific hyperparameter tuning, except for the 2023 [43] top system.

Figures 3 and 4 show the official evaluation scores of *raw_beats* and *dis_spec_scac_trainable* with different backends, respectively. The results show that kNN with SMOTE achieves high performance in most cases. The anomaly score rescaling technique significantly improves the performance of *raw_beats*, although our re-evaluation across multiple datasets reveals that its effect is unstable for the considered discriminative embedding model.

4. CONCLUSION

In this paper, we introduced ASDKit, an open-source toolkit for ASD research. ASDKit provides recipes for various ASD methods, including AE-based approaches, state-of-the-art (SOTA) discriminative methods, and raw feature-based methods. In addition, it supports comprehensive evaluation on the DCASE 2020–2024 datasets with multiple trials. Using ASDKit, we re-evaluated various ASD methods and identified the effectiveness of mixup, SCAC with trainable class centers, multi-resolution spectrograms, and fine-tuned SSL models. We will continuously update ASDKit with new ASD methods and welcome contributions from the community to facilitate further development of ASD research.

5. ACKNOWLEDGMENT

This work was partly supported by JSPS KAKENHI Grant Number JP25KJ1439.

REFERENCES

[1] Y. Koizumi, Y. Kawaguchi, K. Imoto, *et al.*, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Proc. DCASE*, 2020, pp. 81–85.

[2] Y. Kawaguchi, K. Imoto, Y. Koizumi, *et al.*, "Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous detection for machine condition monitoring under domain shifted conditions," in *Proc. DCASE*, 2021, pp. 186–190.

[3] K. Dohi, K. Imoto, N. Harada, *et al.*, "Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques," in *Proc. DCASE*, 2022, pp. 1–5.

[4] K. Dohi, K. Imoto, N. Harada, *et al.*, "Description and discussion on DCASE 2023 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring," in *Proc. DCASE*, 2023, pp. 31–35.

[5] T. Nishida, N. Harada, D. Niizumi, *et al.*, "Description and discussion on DCASE 2024 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring," in *Proc. DCASE*, 2024, pp. 111–115.

[6] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Anomalous sound detection based on interpolation deep neural network," in *Proc. ICASSP*, 2020, pp. 271–275.

[7] K. Dohi, T. Endo, H. Purohit, R. Tanabe, and Y. Kawaguchi, "Flow-based self-supervised density estimation for anomalous sound detection," in *Proc. ICASSP*, 2021, pp. 336–340.

[8] R. Giri, F. Cheng, K. Helwani, S. V. Tenneti, U. Isik, and A. Krishnaswamy, "Group masked autoencoder based density estimator for audio anomaly detection," in *Proc. DCASE*, 2020, pp. 51–55.

[9] K. Wilkinghoff, "Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection," in *Proc. ICASSP*, 2023, pp. 1–5.

[10] K. Wilkinghoff, "Self-supervised learning for anomalous sound detection," in *Proc. ICASSP*, 2024, pp. 276–280.

[11] P. Saengthong and T. Shinozaki, "Deep generic representations for domain-generalized anomalous sound detection," in *Proc. ICASSP*, 2025, pp. 1–5.

[12] X. Zheng, A. Jiang, B. Han, *et al.*, "Improving anomalous sound detection via low-rank adaptation fine-tuning of pre-trained audio models," in *Proc. SLT*, 2024, pp. 969–974.

[13] J. A. Lopez, H. Lu, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," in *Proc. DCASE*, 2020, pp. 96–99.

[14] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Proc. DCASE*, 2020, pp. 46–50.

[15] 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 *Proc. DCASE*, 2020, pp. 170–174.

[16] I. Kuroyanagi, T. Hayashi, K. Takeda, and T. Toda, "Serial-OE: Anomalous sound detection based on serial method with outlier exposure capable of using small amounts of anomalous data for training," *APSIPA Transactions on Signal and Information Processing*, vol. 14, no. 1, 2025.

[17] K. Wilkinghoff and F. Fritz, "On using pre-trained embeddings for detecting anomalous sounds with limited training data," in *Proc. EUSIPCO*, 2023, pp. 186–190.

[18] K. Wilkinghoff, T. Fujimura, K. Imoto, J. Le Roux, Z.-H. Tan, and T. Toda, "Handling domain shifts for anomalous sound detection: A review of DCASE-related work," *arXiv preprint arXiv:2503.10435*, 2025.

[19] T. Fujimura, I. Kuroyanagi, and T. Toda, "Improvements of discriminative feature space training for anomalous sound detection in unlabeled conditions," in *Proc. ICASSP*, 2025, pp. 1–5.

[20] S. Chen, Y. Wu, C. Wang, *et al.*, "BEATs: Audio pre-training with acoustic tokenizers," in *Proc. ICML*, 2023, pp. 5178–5193.

[21] W. Chen, Y. Liang, Z. Ma, Z. Zheng, and X. Chen, "EAT: Self-supervised pre-training with efficient audio transformer," in *Proc. IJCAI*, Main Track, 2024, pp. 3807–3815.

[22] A. Jiang, B. Han, Z. Lv, *et al.*, "Anopatch: Towards better consistency in machine anomalous sound detection," in *Proc. Interspeech*, 2024, pp. 107–111.

[23] A. Jiang, X. Zheng, B. Han, *et al.*, "Adaptive prototype learning for anomalous sound detection with partially known attributes," in *Proc. ICASSP*, 2025, pp. 1–5.

[24] J. Yin, Y. Gao, W. Zhang, T. Wang, and M. Zhang, "Diffusion augmentation sub-center modeling for unsupervised anomalous sound detection with partially attribute-unavailable conditions," in *Proc. ICASSP*, 2025, pp. 1–5.

[25] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "LoRA: Low-rank adaptation of large language models.," *ICLR*, vol. 1, no. 2, p. 3, 2022.

[26] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[27] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Proc. CVPR*, 2019, pp. 10 823–10 832.

[28] K. Wilkinghoff, "Sub-cluster AdaCos: Learning representations for anomalous sound detection," in *Proc. IJCNN*, 2021, pp. 1–8.

[29] K. Wilkinghoff, "AdaProj: Adaptively scaled angular margin subspace projections for anomalous sound detection with auxiliary classification tasks," in *Proc. DCASE*, 2024, pp. 186–190.

[30] K. Wilkinghoff and F. Kurth, "Why do angular margin losses work well for semi-supervised anomalous sound detection?" *IEEE/ACM TASLP*, 2023.

[31] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.

[32] D. S. Park, W. Chan, Y. Zhang, *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *JAIR*, vol. 16, pp. 321–357, 2002.

[34] K. Wilkinghoff, H. Yang, J. Ebbers, F. G. Germain, G. Wichern, and J. Le Roux, "Keeping the balance: Anomaly score calculation for domain generalization," in *Proc. ICASSP*, 2025, pp. 1–5.

[35] J. Guan, Y. Liu, Q. Zhu, T. Zheng, J. Han, and W. Wang, "Time-weighted frequency domain audio representation with GMM estimator for anomalous sound detection," in *Proc. ICASSP*, 2023, pp. 1–5.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019.

[38] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.

[39] N. Harada, D. Niizumi, Y. Ohishi, D. Takeuchi, and M. Yasuda, "First-shot anomaly sound detection for machine condition monitoring: A domain generalization baseline," in *Proc. EUSIPCO*, 2023, pp. 191–195.

[40] J. Lopez, G. Stemmer, and P. Lopez-Meyer, "Ensemble of complementary anomaly detectors under domain shifted conditions," DCASE2021 Challenge, Tech. Rep., 2021.

[41] Y. Zeng, H. Liu, L. Xu, Y. Zhou, and L. Gan, "Robust anomaly sound detection framework for machine condition monitoring," DCASE2022 Challenge, Tech. Rep., 2022.

[42] Z. Lv, A. Jiang, B. Han, *et al.*, "AITHU system for first-shot unsupervised anomalous sound detection," DCASE2024 Challenge, Tech. Rep., 2024.

[43] J. Jie, "Anomalous sound detection based on self-supervised learning," DCASE2023 Challenge, Tech. Rep., 2023.