# DESIGN CHOICES FOR LEARNING EMBEDDINGS FROM AUXILIARY TASKS FOR DOMAIN GENERALIZATION IN ANOMALOUS SOUND DETECTION

*Kevin Wilkinghoff*

Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

## ABSTRACT

Emitted machine sounds can change drastically due to a change in settings of machines or varying noise conditions resulting in false alarms when monitoring machine conditions with a trained anomalous sound detection (ASD) system. In this work, a conceptually simple state-of-the-art ASD system based on embeddings learned through auxiliary tasks generalizing to multiple data domains is presented. In experiments conducted on the DCASE 2022 ASD dataset, particular design choices such as preventing trivial projections, combining multiple input representations and choosing a suitable backend are shown to significantly improve the ASD performance.

***Index Terms*—** anomalous sound detection, representation learning, domain generalization, machine listening

## 1. INTRODUCTION

Semi-supervised anomalous sound detection (ASD) for machine condition monitoring is the task of training an ASD system to distinguish normal from anomalous machine sounds using only normal training samples [1, 2]. To avoid the need to repeatedly collect data and retrain the ASD system in case acoustic conditions or machine attributes change, domain generalization (DG) [3] techniques can be applied. For DG, there are two data domains with somehow different acoustics: a source domain with many training samples and a target domain with only a few training samples. The goal is to obtain an ASD system that correctly detects anomalies regardless of whether sounds belong to the source or target domain.

The most popular state-of-the-art approach for ASD is to train a neural network to extract discriminative embeddings through auxiliary tasks, also called outlier exposed ASD [4], using angular margin losses such as ArcFace [5], AdaCos [6] or sub-cluster AdaCos [7]. Examples of auxiliary tasks are to discriminate between different machine types or among other acoustic characteristics such as different machine settings or noise conditions. The assumption is that the information needed for correctly classifying the sounds is also sufficient to detect anomalous sounds. In contrast to directly modeling the distribution of the data (inlier modeling), e.g. by using autoencoders, the model learns to ignore other sound events and thus to handle noise more effectively whereas a model trained on a single class cannot tell the difference between important and irrelevant components of the signal.

The main contributions of this work are the following: First and foremost, a state-of-the-art ASD system with strong domain generalization capabilities in machine condition monitoring is presented[1]. The system is conceptually simple since its architecture and all hyperparameter settings are the same for each machine type and no external data resources are used for training the system. Furthermore, in ablation studies several design choices are shown to have a significant impact on the performance in the source and target domain. These design choices are 1) preventing trivial projections to hyperspheres, 2) combining multiple input feature representations by jointly training sub-networks and 3) choosing a suitable backend for generalizing to multiple data domains.

## 2. SYSTEM DESCRIPTION

An overview of the ASD system is shown in Fig. 1. The system is an improved version of the one described in [8].

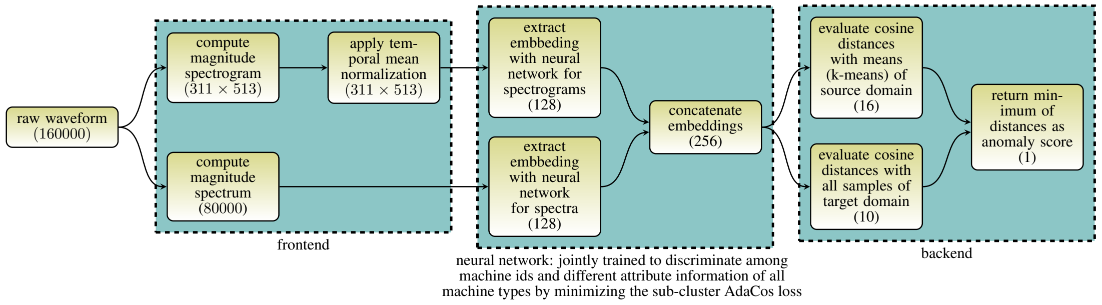
### 2.1. Frontend

The system uses two different feature representations derived from raw waveforms of 10 seconds length and a sampling rate of 16kHz as input. First, magnitude spectrograms with a Hanning-windowed DFT length of 1024 and a hop size of 512. Second, magnitude spectra of entire signals are used to have the highest possible frequency resolution for better capturing stationary sounds. To reduce acoustic differences between source and target domains, sample-wise temporal mean normalization similar to cepstral mean normalization [9] is applied to the magnitude spectrograms.

### 2.2. Neural network architecture

The neural network for extracting the embeddings consists of two different sub-networks for each input representation. To capture as much information as possible, the entire network

[1]An open-source implementation of the proposed system is available at: `https://github.com/wilkinghoff/icassp2023`



raw waveform (160000)
compute magnitude spectrogram (311 × 513)
apply temporal mean normalization (311 × 513)
compute magnitude spectrum (80000)
frontend
extract embedding with neural network for spectrograms (128)
extract embedding with neural network for spectra (128)
concatenate embeddings (256)
neural network: jointly trained to discriminate among machine ids and different attribute information of all machine types by minimizing the sub-cluster AdaCos loss
evaluate cosine distances with means (k-means) of source domain (16)
evaluate cosine distances with all samples of target domain (10)
return minimum of distances as anomaly score (1)
backend

**Fig. 1**. Structure of the proposed anomalous sound detection system. Representation size in each step is given in brackets.

**Table 1**. Modified ResNet architecture for spectrograms.

| layer name | structure | output size |
|---|---|---|
| input | BN (temporal axis) | $311 \times 513$ |
| 2D convolution | $7 \times 7$, stride= 2 | $156 \times 257 \times 16$ |
| residual block | $\begin{pmatrix} 3\times3 \\ 3\times3 \end{pmatrix} \times 2$, stride= 1 | $77 \times 128 \times 16$ |
| residual block | $\begin{pmatrix} 3\times3 \\ 3\times3 \end{pmatrix} \times 2$, stride= 1 | $39 \times 64 \times 32$ |
| residual block | $\begin{pmatrix} 3\times3 \\ 3\times3 \end{pmatrix} \times 2$, stride= 1 | $20 \times 32 \times 64$ |
| residual block | $\begin{pmatrix} 3\times3 \\ 3\times3 \end{pmatrix} \times 2$, stride= 1 | $10 \times 16 \times 128$ |
| max pooling | $10 \times 1$, stride= 1 | $1 \times 16 \times 128$ |
| flatten | BN | 2056 |
| dense (embedding) | linear | 128 |

**Table 2**. Network architecture for spectra.

| layer name | structure | output size |
|---|---|---|
| input | - | 80000 |
| 1D convolution | 256, stride= 64 | $1250 \times 128$ |
| 1D convolution | 64, stride= 16 | $40 \times 128$ |
| 1D convolution | 16, stride= 4 | $10 \times 128$ |
| flatten | - | 1280 |
| dense | BN, ReLU | 128 |
| dense | BN, ReLU | 128 |
| dense | BN, ReLU | 128 |
| dense | BN, ReLU | 128 |
| dense (embedding) | linear | 128 |

is trained to discriminate between all machine types, sections and different attribute information about the machines resulting in a total of 342 classes by minimizing the sub-cluster AdaCos loss [7] with 16 sub-clusters for each class. In contrast to [10], where multiple networks and loss functions have been used for different classification tasks, this training strategy is much simpler. The sub-network used for the spectrograms is based on a modified ResNet architecture [11] as also used in [7,8,10] and is described in Tab. 1. For the spectra, the sub-network consists of three one-dimensional convolutions and five dense layers as shown in Tab. 2. Note that improving the results for the auxiliary task does not necessarily imply a better ASD performance and thus adjusting the number of layers and their parameter size is not critical for improving the ASD capabilities. However, both sub-network architectures are carefully designed to avoid learning trivial mappings to hyperspheres for specific classes as done in networks for deep one-class classification [12]. This means that 1) no bounded non-linearities, 2) no bias terms and 3) no trainable hypersphere centers are used. Instead, for 1) we only use rectified linear units as non-linearities and for 3) we randomly initialize the cluster centers of the sub-cluster AdaCos loss without adapting them during training. A random initialization of the cluster centers is not a problem, since embeddings and cluster centers live in a relatively high-dimensional space and thus are very likely to be pairwise orthogonal.

The output of both sub-networks, which can both be interpreted as embeddings by themselves, are concatenated to obtain a single embedding for each file. This concatenation ensures that *both* networks capture all information needed to discriminate between the classes present in their respective feature representations. Therefore, the embeddings are more sensitive to anomalous sounds than when giving the network the freedom to utilize only a single feature representation (e.g. by taking the sum) because specific anomalies may be apparent in only one of the two input representations.

The entire network is implemented using Tensorflow [13] and is trained for 10 epochs with a batch size of 64 using Adam [14]. For data augmentation, only mixup [15] with a uniformly distributed mixing coefficient is used.

### 2.3. Backend

To obtain an anomaly score for a test sample, for each domain a different strategy is applied. For the source domain, k-means with 16 clusters is applied to the embeddings obtained with all source samples and the cosine distance between the test embedding and all these means is calculated. For the target domain, the cosine distance between the test embedding and the embeddings of all target samples is calculated. Finally, the minimum over all distances is returned as an anomaly score with higher scores indicating anomalies.

## 3. EXPERIMENTAL SETUP

For all experiments, the DCASE 2022 ASD dataset [2] was used. This dataset consists of sounds from the machine types “bearing”, “fan”, “gearbox”, “slider”, “valve” from MIMII DG [16], and “ToyCar”, “ToyTrain” from ToyADMOS2 [17]. For each machine type, there are 6 different subsets of the

**Table 3**. Comparison between using or not using trainable cluster centers and bias terms.

| | | trainable cluster centers | | non-trainable cluster centers | |
|---|---|---|---|---|---|
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| | | using bias terms | | | |
| dev | source | 81.65 ± 0.85 | 70.25 ± 1.31 | 82.75 ± 0.95 | 75.22 ± 0.78 |
| dev | target | 77.18 ± 0.88 | 62.05 ± 1.10 | 76.84 ± 1.39 | 61.66 ± 1.09 |
| dev | mixed | 77.73 ± 0.90 | 64.09 ± 1.40 | 79.79 ± 0.50 | 64.89 ± 0.41 |
| eval | source | 74.21 ± 1.14 | 62.84 ± 1.38 | 76.45 ± 0.71 | 65.12 ± 0.66 |
| eval | target | 71.13 ± 1.24 | 59.54 ± 0.86 | 69.19 ± 0.56 | 59.08 ± 1.15 |
| eval | mixed | 71.91 ± 0.98 | 58.84 ± 0.88 | 73.04 ± 0.58 | 59.18 ± 0.90 |
| | | not using bias terms | | | |
| dev | source | 82.88 ± 0.68 | 71.44 ± 0.96 | **84.19 ± 0.75** | **76.45 ± 0.90** |
| dev | target | 77.68 ± 1.11 | **62.82 ± 1.17** | **78.51 ± 0.90** | 62.54 ± 0.87 |
| dev | mixed | 78.15 ± 0.77 | 64.86 ± 0.52 | **81.36 ± 0.66** | **66.55 ± 0.86** |
| eval | source | 74.34 ± 0.96 | 63.49 ± 0.38 | **76.81 ± 0.79** | **65.84 ± 0.22** |
| eval | target | **71.30 ± 0.33** | **59.94 ± 0.81** | 69.80 ± 0.53 | 59.67 ± 1.14 |
| eval | mixed | 72.15 ± 0.50 | 58.90 ± 0.42 | **73.43 ± 0.54** | **59.78 ± 0.83** |

**Table 4**. Comparison between different input feature representations and ways of combining them.

| | | individual input feature representations | | | | | |
|---|---|---|---|---|---|---|---|
| | | magnitude spectrum | | log-mel magnitude spectrogram | | magnitude spectrogram | |
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| dev | source | 80.75 ± 0.92 | 71.09 ± 1.14 | 70.91 ± 2.10 | 63.21 ± 1.10 | 79.18 ± 1.07 | 70.38 ± 0.43 |
| dev | target | 73.95 ± 1.25 | 61.31 ± 1.40 | 65.78 ± 1.48 | 57.05 ± 0.76 | 76.59 ± 1.59 | 60.59 ± 1.50 |
| dev | mixed | 76.81 ± 0.94 | 63.09 ± 1.23 | 68.93 ± 1.52 | 57.79 ± 0.55 | 77.60 ± 1.02 | 62.31 ± 1.10 |
| eval | source | 68.42 ± 1.02 | 59.06 ± 0.89 | 67.59 ± 1.01 | 59.96 ± 0.65 | 74.65 ± 0.83 | 64.04 ± 1.23 |
| eval | target | 63.46 ± 1.39 | 56.90 ± 0.95 | 62.09 ± 0.61 | 56.60 ± 0.79 | 69.67 ± 1.14 | 58.95 ± 0.74 |
| eval | mixed | 66.00 ± 1.11 | 57.11 ± 0.58 | 65.04 ± 0.68 | 57.00 ± 0.50 | 72.10 ± 0.81 | 58.87 ± 0.33 |

| | | combining magnitude spectrum and magnitude spectrograms | | | | | |
|---|---|---|---|---|---|---|---|
| | | concatenate embeddings after training | | add embeddings while training | | concatenate embeddings while training | |
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| dev | source | **84.68 ± 0.86** | 74.51 ± 0.26 | 83.12 ± 1.18 | 73.25 ± 1.11 | 84.19 ± 0.75 | **76.45 ± 0.90** |
| dev | target | **78.78 ± 0.78** | **63.00 ± 0.36** | 77.96 ± 1.37 | 62.02 ± 1.11 | 78.51 ± 0.90 | 62.54 ± 0.87 |
| dev | mixed | 81.17 ± 0.66 | 65.23 ± 0.39 | 80.21 ± 0.73 | 64.40 ± 1.20 | **81.36 ± 0.66** | **66.55 ± 0.86** |
| eval | source | 75.27 ± 0.96 | 63.93 ± 0.63 | 75.54 ± 0.83 | 64.85 ± 0.73 | **76.81 ± 0.79** | **65.84 ± 0.22** |
| eval | target | 69.31 ± 0.90 | 59.45 ± 0.67 | 69.71 ± 0.39 | 59.08 ± 1.15 | **69.80 ± 0.53** | **59.67 ± 1.14** |
| eval | mixed | 72.18 ± 0.67 | 59.45 ± 0.46 | 72.48 ± 0.53 | 59.22 ± 0.66 | **73.43 ± 0.54** | **59.78 ± 0.83** |

dataset called *sections*, of which three belong to a development set and the other three belong to an evaluation set, corresponding to different types of domain shifts. For each section, there are 990 normal training samples belonging to the source domain, 10 normal training samples belonging to a target domain and 200 test samples each belonging to one of the domains. Furthermore, some attribute information defining states of the machines or different types of noise are given for training samples. All recordings include real factory noise, have a length of 10 seconds and a sampling rate of 16 kHz. For training the network, all machine types, sections and different attribute information about the machines belonging to the development and evaluation set (in each case both domains) are used as classes for the auxiliary task.

In each experiment, each system was trained five times and the arithmetic mean and standard deviation of the harmonic means of all AUCs and pAUCs obtained for all 42 machine ids are shown. Highest AUCs and pAUCs for each combination of dataset and domain are highlighted in bold letters.

## 4. EXPERIMENTAL RESULTS

### 4.1. Preventing trivial projections to hyperspheres

In Tab. 3, the performance obtained with trainable and non-trainable class centers, and when using or not using bias terms are compared. Although non-trainable class centers decrease the performance on the target domain, not using bias terms or trainable class centers improves the overall performance.

### 4.2. Input feature representations

Next, different input feature representations and their combinations are compared. The results are shown in Tab. 4 and show that magnitude spectrograms perform better than magnitude spectra and log-mel magnitude spectrograms with 128 mel bins. When combining a spectrogram with the full spectrum, the performance is even better since the model also has access to the whole frequency resolution instead of only a time-frequency representation. Moreover, the best way to combine these representations is by concatenating the corresponding embeddings while training.

**Table 5**. Effect of temporal normalization.

| | | without temporal normalization | | with temporal normalization | |
|---|---|---|---|---|---|
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| | | magnitude spectrogram | | | |
| dev | source | 79.93 ± 0.71 | 70.98 ± 1.38 | 79.18 ± 1.07 | 70.38 ± 0.43 |
| dev | target | 76.18 ± 0.85 | 60.35 ± 1.23 | 76.59 ± 1.59 | 60.59 ± 1.50 |
| dev | mixed | 77.69 ± 0.47 | 62.52 ± 1.09 | 77.60 ± 1.02 | 62.31 ± 1.10 |
| eval | source | 75.53 ± 1.19 | 64.31 ± 1.08 | 74.65 ± 0.83 | 64.04 ± 1.23 |
| eval | target | 69.52 ± 0.73 | **59.67 ± 0.71** | 69.67 ± 1.14 | 58.95 ± 0.74 |
| eval | mixed | 72.19 ± 0.77 | 59.39 ± 0.91 | 72.10 ± 0.81 | 58.87 ± 0.33 |
| | | magnitude spectrum + magnitude spectrogram | | | |
| dev | source | 83.18 ± 1.68 | 74.66 ± 0.71 | **84.19 ± 0.75** | **76.45 ± 0.90** |
| dev | target | 76.95 ± 0.97 | 62.26 ± 0.62 | **78.51 ± 0.90** | **62.54 ± 0.87** |
| dev | mixed | 80.04 ± 0.76 | 64.84 ± 0.51 | **81.36 ± 0.66** | **66.55 ± 0.86** |
| eval | source | 76.41 ± 0.48 | 65.39 ± 0.68 | **76.81 ± 0.79** | **65.84 ± 0.22** |
| eval | target | 68.89 ± 0.96 | 59.46 ± 0.68 | **69.80 ± 0.53** | **59.67 ± 1.14** |
| eval | mixed | 72.85 ± 0.61 | **59.91 ± 0.75** | **73.43 ± 0.54** | 59.78 ± 0.83 |

In Tab. 5, the effect of using temporal normalization for the magnitude spectrograms is investigated. It can be seen that temporal normalization improves the performance on the target domain as intended while slightly but not significantly degrading the performance on the source domain when using only magnitude spectrograms. But when combining spectra and spectrograms, temporal normalization also improves the performance on the source domain. The reason is that both input representations complement each other more effectively because stationary frequency information are removed from the spectrograms but are clearly contained in the spectra.

### 4.3. Backends

In Tab. 6, different backends, namely cosine distance and a GMM with 16 Gaussian components and a full covariance matrix regularized by adding $10^{-3}$ to the diagonal (for more details, see [8]), are compared. The following observations can be made: As expected, specialized models for individual domains perform better on the domain they are trained on and much worse on the other domain. For the source domain, the GMM model has a slightly higher performance than the cosine similarity, which is consistent with the findings in [7]. For the target domain, both backends are equivalent.

**Table 6**. Comparison between different backends.

| | | Gaussian mixture model (GMM) | | cosine distance | |
|---|---|---|---|---|---|
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| | | using scores from source domain model only | | | |
| dev | source | $82.97 \pm 0.97$ | $77.36 \pm 0.38$ | $83.10 \pm 1.02$ | $76.87 \pm 0.26$ |
| dev | target | $66.52 \pm 0.42$ | $59.63 \pm 0.70$ | $71.66 \pm 1.25$ | $61.45 \pm 0.83$ |
| dev | mixed | $71.48 \pm 0.26$ | $58.86 \pm 0.38$ | $76.72 \pm 0.78$ | $63.37 \pm 0.71$ |
| eval | source | $77.46 \pm 1.16$ | $66.73 \pm 0.56$ | $76.68 \pm 0.85$ | $66.25 \pm 0.51$ |
| eval | target | $44.23 \pm 3.67$ | $54.87 \pm 0.46$ | $57.90 \pm 1.17$ | $55.62 \pm 1.24$ |
| eval | mixed | $63.83 \pm 0.62$ | $55.74 \pm 0.33$ | $67.24 \pm 0.70$ | $56.83 \pm 0.92$ |
| | | using scores from target domain model only | | | |
| dev | source | $62.41 \pm 2.80$ | $60.54 \pm 1.44$ | $62.42 \pm 2.80$ | $60.55 \pm 1.44$ |
| dev | target | $\mathbf{79.93 \pm 0.92}$ | $62.19 \pm 1.05$ | $\mathbf{79.92 \pm 0.92}$ | $62.18 \pm 1.06$ |
| dev | mixed | $70.84 \pm 1.13$ | $58.36 \pm 1.38$ | $70.84 \pm 1.13$ | $58.36 \pm 1.38$ |
| eval | source | $52.82 \pm 3.40$ | $56.27 \pm 1.17$ | $52.80 \pm 3.41$ | $52.26 \pm 1.17$ |
| eval | target | $\mathbf{71.15 \pm 0.50}$ | $\mathbf{60.72 \pm 0.95}$ | $\mathbf{71.15 \pm 0.50}$ | $\mathbf{60.72 \pm 0.95}$ |
| eval | mixed | $62.55 \pm 0.79$ | $54.66 \pm 0.92$ | $62.55 \pm 0.79$ | $54.65 \pm 0.92$ |
| | | using sum of scores from both domain models | | | |
| dev | source | $75.94 \pm 2.70$ | $70.34 \pm 2.69$ | $79.44 \pm 1.95$ | $73.29 \pm 2.60$ |
| dev | target | $78.64 \pm 1.12$ | $63.28 \pm 0.94$ | $78.84 \pm 1.23$ | $62.67 \pm 0.98$ |
| dev | mixed | $77.10 \pm 1.59$ | $65.12 \pm 1.63$ | $77.83 \pm 1.56$ | $66.11 \pm 1.61$ |
| eval | source | $64.57 \pm 1.35$ | $60.96 \pm 1.21$ | $68.96 \pm 1.11$ | $63.09 \pm 0.87$ |
| eval | target | $66.69 \pm 0.54$ | $58.73 \pm 0.97$ | $67.84 \pm 0.48$ | $58.80 \pm 1.33$ |
| eval | mixed | $65.70 \pm 0.91$ | $57.26 \pm 0.77$ | $67.98 \pm 0.64$ | $58.12 \pm 0.71$ |
| | | using scores from joint model for both domains | | | |
| dev | source | $\mathbf{84.57 \pm 0.70}$ | $\mathbf{77.57 \pm 0.41}$ | $84.19 \pm 0.75$ | $76.45 \pm 0.90$ |
| dev | target | $77.26 \pm 1.02$ | $\mathbf{63.12 \pm 1.24}$ | $78.51 \pm 0.90$ | $62.54 \pm 0.87$ |
| dev | mixed | $80.06 \pm 0.35$ | $64.67 \pm 0.70$ | $\mathbf{81.36 \pm 0.66}$ | $\mathbf{66.55 \pm 0.86}$ |
| eval | source | $\mathbf{78.15 \pm 0.95}$ | $\mathbf{67.55 \pm 0.63}$ | $76.81 \pm 0.79$ | $65.84 \pm 0.22$ |
| eval | target | $65.17 \pm 0.48$ | $58.59 \pm 0.79$ | $69.80 \pm 0.53$ | $59.67 \pm 1.14$ |
| eval | mixed | $71.35 \pm 0.53$ | $59.32 \pm 0.83$ | $\mathbf{73.43 \pm 0.54}$ | $\mathbf{59.78 \pm 0.83}$ |

**Table 7**. Effect of the presented design choices.

| | | standard design choices | | presented design choices | |
|---|---|---|---|---|---|
| dataset | domain | AUC (%) | pAUC (%) | AUC (%) | pAUC (%) |
| dev | source | $71.65 \pm 1.07$ | $62.71 \pm 0.74$ | $\mathbf{84.19 \pm 0.75}$ | $\mathbf{76.45 \pm 0.90}$ |
| dev | target | $63.63 \pm 1.11$ | $55.28 \pm 0.84$ | $\mathbf{78.51 \pm 0.90}$ | $\mathbf{62.54 \pm 0.87}$ |
| dev | mixed | $67.72 \pm 0.79$ | $55.94 \pm 0.54$ | $\mathbf{81.36 \pm 0.66}$ | $\mathbf{66.55 \pm 0.86}$ |
| eval | source | $69.11 \pm 1.05$ | $58.46 \pm 0.42$ | $\mathbf{76.81 \pm 0.79}$ | $\mathbf{65.84 \pm 0.22}$ |
| eval | target | $61.33 \pm 0.70$ | $54.82 \pm 0.78$ | $\mathbf{69.80 \pm 0.53}$ | $\mathbf{59.67 \pm 1.14}$ |
| eval | mixed | $64.59 \pm 0.74$ | $55.22 \pm 0.66$ | $\mathbf{73.43 \pm 0.54}$ | $\mathbf{59.78 \pm 0.83}$ |

For domain generalization, it is necessary to use a single decision threshold for both domains and thus a single ASD score is required. It can be seen that training a joint model has a significantly higher performance than adding the ASD scores of individually trained models. However, for the target domain the joint model is worse than a specialized model. For the source domain, the joint GMM model has a better performance than using cosine distance again and interestingly even outperforms the specialized model. But when jointly evaluating both domains, using the cosine distance as a backend has a higher performance than using a GMM. The most probably reason is that in contrast to a GMM the cosine distance requires no training and thus the resulting ASD scores are scaled more consistently among both domains.

## 4.4. Putting it all together

To show the strong impact of the design choices on the ASD performance, we compared a system with standard design choices to the presented ones. The standard design choices are using log-mel magnitude spectrograms, trainable cluster centers, bias terms, and GMMs as backend and are for example used in the ASD system [10] ranked third in the DCASE challenge 2021. The results are shown in Tab. 7. It can be seen that the presented design choices led to a significant performance improvement for all domains and dataset splits.

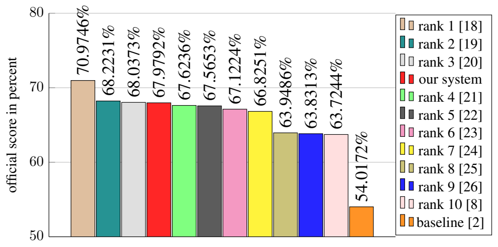


**Fig. 2**. Comparison between presented, baseline and 10 top-performing systems of the DCASE challenge 2022.

## 4.5. Comparison to other systems

Last, we compared the performance of our system (see Fig. 1) to the baseline and 10 top-performing systems of the DCASE challenge 2022. For these experiments, we trained the presented system five times and created an ensemble by adding the corresponding anomaly scores. The results can be found in Fig. 2. Our system would have obtained rank 4, performing close to the rank 3 system [20], and thus can be considered state-of-the-art. Note that the rank 1 system [18] used manually customized band-pass filters for each machine type without stating the details in the report and monitored the AUC score on the development set while training to find the best performing model parameters and thus also indirectly used anomalous samples for training. Both do not allow a fair comparison between performances and are a possible explanation for the large performance gap to all other systems.

## 5. CONCLUSIONS

In this work a conceptually simple ASD system based on learning embeddings through auxiliary tasks for domain generalization was presented. In various experiments conducted on the DCASE 2022 ASD dataset, it was shown that several design choices, namely preventing trivial projections, utilizing multiple input feature representations and choosing a suitable backend, significantly improve the performance and that the presented system achieves state-of-the-art performance. For future work, it is planned to carry out additional experiments to verify whether the presented design choices also improve the performances when using other ASD systems.

## 6. REFERENCES

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

[2] Kota Dohi et al., "Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques," in *DCASE*. 2022, pp. 26–30, Tampere University.

[3] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin, "Generalizing to unseen domains: A survey on domain generalization," in *IJCAI*. 2021, pp. 4627–4635, ijcai.org.

[4] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich, "Deep anomaly detection with outlier exposure," in *ICLR*. 2019, OpenReview.net.

[5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *CVPR*. 2019, pp. 4690–4699, IEEE.

[6] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *CVPR*. 2019, pp. 10823–10832, IEEE.

[7] Kevin Wilkinghoff, "Sub-cluster AdaCos: Learning representations for anomalous sound detection," in *IJCNN*. 2021, IEEE.

[8] Kevin Wilkinghoff, "An outlier exposed anomalous sound detection system for domain generalization in machine condition monitoring," Tech. Rep., DCASE Challenge, 2022.

[9] Aaron E. Rosenberg, Chin-Hui Lee, and Frank K. Soong, "Cepstral channel normalization techniques for HMM-based speaker verification," in *ICSLP*. 1994, ISCA.

[10] Kevin Wilkinghoff, "Combining multiple distributions based on sub-cluster AdaCos for anomalous sound detection under domain shifted conditions," in *DCASE*, 2021, pp. 55–59.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*. 2016, pp. 770–778, IEEE.

[12] Lukas Ruff et al., "Deep one-class classification," in *ICML*. 2018, vol. 80, pp. 4390–4399, PMLR.

[13] Martín Abadi et al., "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016, pp. 265–283.

[14] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[15] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[16] Kota Dohi et al., "MIMII DG: Sound dataset for malfunctioning industrial machine investigation and inspection for domain generalization task," in *DCASE*. 2022, pp. 31–35, Tampere University.

[17] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Masahiro Yasuda, and Shoichiro Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," in *DCASE*, 2021, pp. 1–5.

[18] Ying Zeng, Hongqing Liu, Lihua Xu, Yi Zhou, and Lu Gan, "Robust anomaly sound detection framework for machine condition monitoring," Tech. Rep., DCASE Challenge, 2022.

[19] Ibuki Kuroyanagi, Tomoki Hayashi, Kazuya Takeda, and Tomoki Toda, "Two-stage anomalous sound detection systems using domain generalization and specialization techniques," Tech. Rep., DCASE Challenge, 2022.

[20] Feiyang Xiao et al., "The dcase2022 challenge task 2 system: Anomalous sound detection with self-supervised attribute classification and gmm-based clustering," Tech. Rep., DCASE Challenge, 2022.

[21] Yufeng Deng, Jia Liu, and Wei-Qiang Zhang, "Aithu system for unsupervised anomalous detection of machine working status via sounding," Tech. Rep., DCASE Challenge, 2022.

[22] Satvik Venkatesh, Gordon Wichern, Aswin Subramanian, and Jonathan Le Roux, "Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection," Tech. Rep., DCASE Challenge, 2022.

[23] Yuming Wei, Jian Guan, Haiyan Lan, and Wenwu Wang, "Anomalous sound detection system with self-challenge and metric evaluation for dcase2022 challenge task 2," Tech. Rep., DCASE Challenge, 2022.

[24] Kazuki Morita, Tomohiko Yano, and Khai Tran, "Comparative experiments on spectrogram representation for anomalous sound detection," Tech. Rep., DCASE Challenge, 2022.

[25] Jisheng Bai, Yafei Jia, and Siwei Huang, "Jless submission to dcase2022 task2: Batch mixing strategy based method with anomaly detector for anomalous sound detection," Tech. Rep., DCASE Challenge, 2022.

[26] Sergey Verbitskiy, Milana Shkhanukova, and Viacheslav Vyshegorodtsev, "Unsupervised anomalous sound detection using multiple time-frequency representations," Tech. Rep., DCASE Challenge, 2022.