Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection

Kevin Wilkinghoff
Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE
Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

***Abstract*—When training a model for anomalous sound detection, one usually needs to estimate the underlying distribution of the normal data. By doing so, anomalous data has a lower probability in view of this distribution than normal data and thus can easily be detected. However, audio data is very high-dimensional making it difficult to have a good estimate of the true distribution. To have more accurate estimates, the dimension of the data can be reduced first. One way to do this is to train discriminative neural networks for extracting lower-dimensional representations of the data. Particularly, neural networks trained with angular margin losses as AdaCos have been shown to perform well for this task. In this work, a modified AdaCos loss called sub-cluster AdaCos specifically designed for detecting anomalous data is presented. In multiple experiments conducted on the DCASE 2020 dataset for "Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring", these design choices are empirically justified. As a result, a conceptually simple system for anomalous sound detection is presented that significantly outperforms all other published systems on this dataset.**

***Index Terms*—machine listening, anomaly detection, representation learning, angular margin loss**

I. Introduction

Anomalous sound detection has various applications such as detecting traffic accidents in road surveillance systems [1], [2], detecting terrorist attacks in subway stations [3] or machine condition monitoring [4]. Furthermore, all open-set classification problems include anomaly detection as a subtask since not all classes are known *a priori* when training the system. Hence, anomalous sound detection is of special interest for many machine listening applications. One can distinguish three major branches of anomaly detection: supervised, semi-supervised and unsupervised [5]. For supervised anomaly detection, two labeled datasets consisting of normal data and anomalous data are used. Although the space of anomalies is huge because it consists of everything that is not considered normal and thus can never be captured exhaustively, samples of anomalies can simplify the training process. This is especially true, when all expected anomalies sound roughly alike as for example when detecting accidents with traffic surveillance systems. In contrast to unsupervised anomaly detection where the training dataset can also include anomalous data and it is not known whether a data sample is normal or not, semi-supervised anomaly detection consists of a clean training dataset containing only normal data. Note that in most practical applications it is much easier to collect normal data than anomalous data. The main reasons are that anomalous events occur only rarely and can sound much more diverse than recordings of the normal condition. Using the traffic example again, large amounts of audio data belonging to regular road traffic can be collected much easier than recordings of accidents i.e. anomalous data. When considering terrorist attacks, this is even more evident. Therefore, not needing anomalous data for training a system is more suitable for realistic applications and thus a semi-supervised setting is considered in this work.

Usually, semi-supervised or unsupervised anomaly detection boils down to estimating the true underlying distribution of the known data without the aid of sample outliers. Afterwards, one can utilize this distribution to compute the log-probability of the test data or an approximation of it and decide whether this data is an inlier or an outlier using a threshold. However, raw data e.g. waveforms is mostly very high-dimensional making it challenging to estimate the corresponding distribution with limited training data resources. To circumvent this problem, one can train a model for extracting suitable, lower-dimensional representations of the data, which capture enough information such that representations belonging to outliers strongly deviate from inliers. One way to train such a model is to discriminate among all known classes in a supervised manner and utilize the output of an intermediate layer as a feature extractor for a lower-dimensional representation of the data. The assumption is that in order to be able to discriminate among the known classes, all representations need to capture enough information of the raw data and this information is also sufficient to detect outliers. In particular, angular margin losses such as ArcFace [6] and AdaCos [7] have been shown to work well for this task. The reason is that they enforce a low intra-class variability and a high inter-class variability by minimizing the cosine angle of a known class to a learned mean value and ensuring a margin for angles between different classes.

The aim of this work is to investigate specific design choices for an anomalous sound detection system based on an angular margin loss function. The contributions are the following: First, several changes for the AdaCos loss function are proposed leading to a novel loss function called sub-cluster AdaCos. The proposed changes are 1) taking into account the use of mixup to augment the samples, 2) utilizing sub-clusters to learn a less restrictive distribution than standard

AdaCos and 3) using Gaussian distributions or more generally Gaussian mixture models (GMMs) as a backend. In various experiments, it is shown that all of these changes lead to significant improvements in performance when detecting anomalous sounds. As a result, a conceptually simple anomalous sound detection system is presented that significantly outperforms all other published systems on the DCASE 2020 dataset for "Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring" [4].

II. Related Work

Recent work on machine listening is heavily promoted through the annual "Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop" and the associated challenges. Anomalous sound detection is not an exception. Of particular interest for this work is task 2 "Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring" [4] of DCASE 2020. The baseline system of this task consists of class-dependent autoencoders for encoding and decoding consecutive frames of log-Mel spectrograms and utilizing the reconstruction error as an anomaly score. The underlying assumption is that normal data, which has also been used for training the autoencoders, can be reconstructed much better than anomalous data. This fundamental approach of using autoencoders for detecting anomalous data has been extended by utilizing autoencoders conditioned on the machine ids i.e. the class labels [8], [9]. The main idea is that a single autoencoder is used instead of a separate one for each class and trained to have a low reconstruction error when being conditioned on the correct class and a high reconstruction error when being conditioned on another class.

A completely different approach is to train a neural network to discriminate among all known classes. By essentially treating the other classes as anomalous data, decision boundaries are learned for the normal data of each class. Several systems following this approach have been developed independently in the DCASE challenge 2020, most of which use neural networks with a suitable loss function that also reduces intra-class variability to extract lower-dimensional representations of the data. Inoue et al. [10] use center loss [11], Lopez et al. [12] an additive margin softmax layer [13] and Giri et al. [14] as well as Zhou [15] use ArcFace [6] for this purpose. After training such a discriminative network, these lower dimensional representations are utilized in different ways to obtain anomaly detection scores. In most cases, the direct output of the trained representation model or the cosine similarity are used. Another method is to train an additional backend model as for example probabilistic linear discriminant analysis (PLDA) [16] as done in [17].

III. Methodology

The purpose of this section is to first give a short review on angular margin losses, particularly AdaCos, and then propose a modified AdaCos loss that results in significantly better anomalous sound detection performance.

A. Standard AdaCos loss function

For many years the softmax function in combination with the categorical crossentropy as a loss function has been the standard output layer for classification tasks solved by neural networks. To avoid any confusion, within this work the term "class" corresponds to one of the known classes for which normal training data is available. This means that each machine id is treated as another class. But when training a neural network for the purpose of extracting lower-dimensional representations of the data, so-called embeddings, the softmax function only leads to representations that are linearly separable without explicitly reducing intra-class and increasing inter-class distance of samples. To address this issue, losses based on the Euclidean distance as for example triplet loss [18] and center loss [11] were proposed. Triplet loss uses an anchor input whose distance to a positive and a negative input sample belonging to the same and to another class is minimized and maximized, respectively. Center loss avoids constructing these triplets as input by minimizing the distance to learned center vectors for each class. Recently, angular margin loss functions such as ArcFace [6] have been shown to have better generalization capabilities than losses based on the Euclidean distance by enforcing a margin between angles of samples belonging to different classes. However, the performance in a particular task obtained with angular margin losses heavily relies on fine-tuning their hyperparameters, namely the scale parameter s and the angular margin parameter m. Therefore, AdaCos [7], which uses an adaptive scale parameter and does not depend on any manually set hyperparameters, was developed. Since both losses, ArcFace and AdaCos, lead to similar performance and tuning additional parameters is time consuming, this work focuses entirely on AdaCos.

Let us now formally introduce AdaCos. For AdaCos, the probability of sample $x_i \in \mathbb{R}^D$ belonging to class j of the $C \in \mathbb{N}$ classes is given by

$$P_{i,j} := \frac{\exp(\tilde{s} \cdot \cos\theta_{i,j})}{\sum_{k=1}^{C} \exp(\tilde{s} \cdot \cos\theta_{i,k})} \tag{1}$$

where $\theta_{i,j} \in [0,\pi]$ is defined through the cosine similarity $\cos\theta_{i,j} = \langle x_i, W_j\rangle / \|x_i\| \|W_j\|$ for a learned class center $W_j \in \mathbb{R}^D$. The dynamically adaptive scale parameter $\tilde{s}^{(t)}$ at training step $t \in \mathbb{N}_0$ is defined as

$$\tilde{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(C-1) & \text{if } t = 0 \\ \frac{\log B_{\text{avg}}^{(t)}}{\cos\left(\min(\frac{\pi}{4}, \theta_{\text{med}}^{(t)})\right)} & \text{else} \end{cases} \tag{2}$$

where $\theta_{\text{med}}^{(t)} \in [0,\pi]$ denotes the median of all angles θ_{i,y_i} belonging to a mini-batch of size $N \in \mathbb{N}$ with y_i being the class of x_i and

$$B_{\text{avg}}^{(t)} := \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{\substack{k=1 \\ k \neq y_i}}^{C} \exp\left(\tilde{s}^{(t-1)} \cdot \cos\theta_{i,k}\right) \tag{3}$$

with $\mathcal{N}^{(t)}$ denoting all indices of the samples belonging to the mini-batch of size $N \in \mathbb{N}$.

B. Sub-Cluster AdaCos loss function

The data augmentation technique "mixup" [19] is known to significantly improve the classification performance, since overfitting of a model to specific training data is prohibited. This is done by linearly interpolating between two samples $x_i, x_j \in \mathbb{R}^D$ contained in a mini-batch and their corresponding one-hot encoded class labels $\bar{y}_i, \bar{y}_j \in [0,1]^C$

$$\begin{aligned} x^{\text{mixed}} &:= \lambda x_i &&+ (1-\lambda)x_j \\ \bar{y}^{\text{mixed}} &:= \lambda \bar{y}_i &&+ (1-\lambda)\bar{y}_j \end{aligned} \tag{4}$$

where the mixing coefficient $\lambda \in [0,1]$ is drawn at random from a suitable distribution. But if one uses mixup, standard AdaCos is not well-defined since most mixed-up samples do not belong to a single class but multiple ones and thus do not have a well-defined class mean. Even when both mixed-up samples belong to the same class and thus have a well-defined class mean, these samples can be treated as anomalies when updating the AdaCos parameters to have a sharper boundary around the support of the distribution of the normal, non-mixed samples of this particular class. To incorporate these changes into the AdaCos function, only mixed-up samples are used for training with mixing coefficients drawn from the uniform distribution. Furthermore, $\hat{\theta}_{\text{med}}^{(t)}$ is the median of the mixed-up angles and $\hat{B}_{\text{avg}}^{(t)}$ is computed using all angles present in a mini-batch, not only the angles of the non-corresponding classes. The details will follow below.

When using AdaCos, only a single cluster is formed for each class and this enforces a single Gaussian distribution for the learned representations after projecting them to the unit sphere by normalizing their lengths. However, anomalous data is easier to detect when a more general, less restrictive distribution is learned for the representations. Gaussian mixture models can approximate any given smooth probability density function. Therefore, a canonical choice to relax the restriction on the distribution imposed by the AdaCos loss is to allow multiple sub-clusters. The same idea of automatically finding subclasses is also used in subclass discriminant analysis and has been shown to outperform other discriminant analysis approaches without subclasses [20], [21]. Note that Deng et al. proposed a very similar approach for the ArcFace loss, namely sub-center ArcFace that uses multiple learned sub-centers for each class instead of only one to efficiently handle label noise present in the training data [22]. This is done by first training the model and then dropping all samples belonging to small sub-clusters, so called non-dominant sub-centers. However, there is a subtle but important difference between sub-center ArcFace and sub-cluster AdaCos. For sub-center ArcFace, only the closest sub-center is considered by taking the maximum cosine similarity. Here, all softmax scores of all sub-clusters of any given class are later summed up to encourage the usage of multiple sub-clusters and thus modeling a more complex distribution.

To avoid numerical issues of large arguments inside the exponential function, which frequently arose in the conducted experiments due to the changes from above, a re-scaling trick that is also used in many implementations of the softmax function is applied. More concretely, the maximum value of the logits is determined

$$f_{\max}^{(t)} := \max_{i \in \mathcal{N}^{(t)}} \max_{j=1}^{CS} \left(\tilde{s}^{(t-1)} \cdot \cos\theta_{i,j} \right) \tag{5}$$

where $S \in \mathbb{N}$ denotes the number of sub-clusters and inserted appropriately into the formulas for $\hat{B}_{\text{avg}}^{(t)}$ and $\hat{s}^{(t)}$.

Using the same notation as before, Sub-Cluster AdaCos with all proposed changes is now defined as follows. The modified adaptive scale parameter $\hat{s}^{(t)}$ is defined as

$$\hat{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(CS - 1) & \text{if } t = 0 \\ \frac{f_{\max}^{(t)} + \log \hat{B}_{\text{avg}}^{(t)}}{\cos\left(\min(\frac{\pi}{4}, \hat{\theta}_{\text{med}}^{(t)})\right)} & \text{else} \end{cases} \tag{6}$$

where

$$\hat{B}_{\text{avg}}^{(t)} := \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{k=1}^{CS} \exp\left(\hat{s}^{(t-1)} \cdot \cos\theta_{i,k} - f_{\max}^{(t)} \right) \tag{7}$$

and $\hat{\theta}_{\text{med}}^{(t)}$ is the median of the mixed-up angles $\theta_{k,y_k}^{\text{mixed}}$. For a mixed-up sample $x_k^{\text{mixed}} = \lambda x_i + (1-\lambda)x_j$ of the mini-batch with λ fixed, this means

$$\theta_{k,y_k}^{\text{mixed}} := \lambda\theta_{i,y_i} + (1-\lambda)\theta_{j,y_j}. \tag{8}$$

Finally, the probabilities of sample x_i belonging to class j are given by

$$\hat{P}_{i,j} := \sum_{l \in \mathcal{M}^{(j)}} \frac{\exp(\hat{s} \cdot \cos\theta_{i,l})}{\sum_{k=1}^{CS} \exp(\hat{s} \cdot \cos\theta_{i,k})} \tag{9}$$

where $\mathcal{M}^{(j)}$ denotes all sub-clusters belonging to class j.

C. Outlier detection backend

When training a neural network to extract representations for anomaly detection, there are multiple ways of how to obtain scores and reach a final decision. Now, by using an angular margin loss, the most natural choice to calculate a score besides using the model itself, is to use the cosine similarity of a sample to the center of the corresponding class. A Gaussian distribution can also be seen as an alternative version of the cosine similarity, since the representations are trained such that they are scattered around a learned mean value after projecting them onto the unit sphere by normalizing their lengths, and the unit sphere is locally Euclidean as a manifold. Moreover, Gaussian distributions are more general than the cosine similarity as a scoring method because the cosine similarity is equivalent to using a Gaussian distribution with a spherical covariance matrix i.e. a diagonal matrix with all entries being equal. This is not a problem when doing closed-set classification because only the closest class mean is important. But for anomaly detection, this assumption does not need to be true as illustrated in Fig. 1. Therefore, a Gaussian distribution with full covariance matrix can more accurately estimate the distribution of the normal samples, in case it slightly deviates from a spherical distribution. This makes the detection of anomalies slightly more robust and thus a full covariance matrix is more suitable for the task. When using sub-cluster AdaCos, one can use a GMM with multiple modes

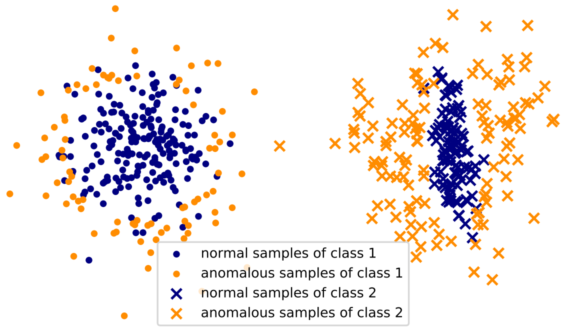

Fig. 1. Scatter plot of normal and anomalous data belonging to two different classes after projecting 3 dimensional representations onto the unit sphere and locally embedding the unit sphere into the 2 dimensional space. Both classes can be easily separated by measuring the distance to the mean. For class 1, anomalies can also be detected reasonably well, but for class 2 only measuring the distance to the mean does not work well because the data is not distributed spherically. In this case, a Gaussian with full covariance matrix would perform much better. Note that this plot is exaggerated for illustration purposes because an angular margin loss ensures that the distribution for each class is roughly spherical. Still, the distribution can slightly deviate from that making a full covariance matrix more suitable, especially in higher-dimensional spaces.

equal to the number of sub-clusters and initialize the means of the modes as the learned sub-cluster centers. After training, the highest log-probability among all the modes can be utilized as an anomaly detection score.

IV. Experimental Results

A. Dataset

For all experiments in this work the dataset belonging to task 2 "Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring" [4] of the DCASE challenge 2020 is used. As the name already implies, the task is to tell whether a recording is normal i.e. belongs to a fully functioning machine or not. The dataset consists of Wav files, each of length 10s and a sampling rate of 16kHz. Each file belongs to one of six different machine types, namely "fan", "pump", "slider", "valve" from MIMII [23] and "ToyCar" and "ToyConveyor" from ToyADMOS [24]. For each of the 6 machine types there are 7 different machine ids except for "ToyConveyor", which has recordings from 6 different machines. When training and testing it is known to which of the total 41 machines a given audio file belongs to. Furthermore, the dataset is divided into a training set, only consisting of around 1000 normal samples of all machine ids, and a development set and an evaluation set, both consisting of a few hundred normal and anomalous samples from mutually exclusive sets of 3 or 4 machine ids for each machine type. It is not allowed to use any of the files from the development or evaluation set for training. Hence, only normal data is available to train the system and it is known that the training dataset consists of normal samples only. Therefore, despite its name, the anomaly detection task is actually semi-supervised instead of unsupervised. The evaluation metrics for the dataset

TABLE I
Modified ResNet architecture used for all experiments.

layer name	structure	output size
input	-	313×128
2D convolution	7×7, stride= 2	$157 \times 64 \times 16$
residual block	$\begin{pmatrix} 3\times 3 \\ 3\times 3 \end{pmatrix} \times 2$, stride= 1	$78 \times 31 \times 16$
residual block	$\begin{pmatrix} 3\times 3 \\ 3\times 3 \end{pmatrix} \times 2$, stride= 1	$39 \times 16 \times 32$
residual block	$\begin{pmatrix} 3\times 3 \\ 3\times 3 \end{pmatrix} \times 2$, stride= 1	$20 \times 8 \times 64$
residual block	$\begin{pmatrix} 3\times 3 \\ 3\times 3 \end{pmatrix} \times 2$, stride= 1	$10 \times 4 \times 128$
max pooling	10×1, stride= 1	4×128
flatten	-	512
dense (representation)	linear	128
AdaCos	-	41

are the area under the receiver operating characteristic (ROC) curve (AUC) and the partial AUC (pAUC) with $p = 0.1$. The metric pAUC is the AUC computed under a low false-positive-rate range, namely $[0, p]$, which is used because a high true-positive-rate is desirable under this conditions in practical applications to avoid frequent false alarms. For more details about the dataset, the reader is referred to [4].

B. Input features and neural network architecture

Using the dataset described above a neural network architecture for extracting lower-dimensional representations of the data as well as their input features need to be defined. Doing so is the purpose of this section. First, the raw waveforms are converted into log-Mel spectrograms to initially reduce their dimension. More concretely, log-Mel spectrograms with 128 Mel bins, a window size of 1024 and a hop size of 512 are computed, which results in a time dimension of 313. Before inserting them as features into the neural network, all log-Mel spectrograms are normalized by subtracting the temporal mean and dividing by the temporal standard deviation of all files belonging to the training dataset. It has also been experimented with using openL3 embeddings [25] as intermediate representations instead of directly using the log-Mel spectrograms as done in [17], [26], but this degraded the performance.

The network architecture used in this work is a modified version of the ResNet architecture [27] with only a few layers and extracts 128-dimensional representations of the data. In each residual block, the leaky ReLU activation function [28] and batch normalization [29] are used. A detailed description of the model can be found in Tab I. It is worth noting that increasing the number of sub-clusters S used for AdaCos also increases the number of parameters significantly. A model without sub-clusters has $772,192$ trainable parameters and when using 64 sub-clusters, which results in the largest model used in the conducted experiments, this number increases to $1,102,816$. Still, the number of parameters is relatively small. When training the model, the training data belonging to all 41 machine ids is used. Using L^2-regularization applied to the

TABLE II
MEAN AUCS AND PAUCS OBTAINED WITH DIFFERENT BACKENDS ON THE DEVELOPMENT AND EVALUATION SET.

backend	development set		evaluation set	
	AUC	pAUC	AUC	pAUC
representation model output	87.20%	81.70%	89.55%	83.79%
cosine similarity to mean	88.71%	82.12%	91.13%	84.40%
cosine similarity to top 10	88.69%	82.12%	91.10%	84.38%
two-covariance PLDA	88.25%	82.18%	90.90%	84.32%
Gaussian (spherical covariance)	88.69%	82.12%	91.11%	84.38%
Gaussian (diagonal covariance)	88.71%	82.16%	91.12%	84.39%
Gaussian (full covariance)	**89.13%**	**82.59%**	**91.43%**	**84.47%**

weights, the model is trained for 400 epochs with a batch-size of 64 to discriminate among these classes by minimizing different versions of the AdaCos loss with Adam [30] and is implemented in Tensorflow [31]. Unless stated otherwise, "mixup" [19] with a mixing coefficient drawn from a uniform distribution and no other data augmentation technique is used.

C. Comparison of different backends

First, the performance obtained with different backends will be compared. For that purpose, a neural network with the standard AdaCos loss and using mix-up is trained to extract the representations. Using the same extracted embeddings, multiple backends are evaluated: the output of the model itself, the cosine similarity to the mean of each machine id, the mean of the cosine similarities to the 10 closest representations from the training set belonging to the same machine id, the log-likelihood ratios of a two-covariance PLDA model as implemented in [32] and Gaussian distributions, each trained for a single machine id, with a spherical, diagonal or full covariance matrix as implemented in scikit-learn [33]. The results can be found in Tab. II.

There are four observations to be made. First, the direct angular softmax output of the model performs significantly worse than all other scoring techniques. Second, PLDA performs better than the direct output but still worse than the remaining backends. Third, as expected, the cosine similarity based backends and the Gaussians with spherical or diagonal covariance matrix all lead to very similar results supporting the claim from before that they are equivalent. And fourth, a Gaussian with a full covariance matrix outperforms all other backends. Hence, in all remaining experiments only Gaussians with full covariance matrix will be used as backends.

D. Interplay of mixup and AdaCos

Next, it is investigated whether including mixup for training the model and the proposed changes of the AdaCos loss function to properly work with mixup improve the performance. The results are depicted in Tab. III. One can see that the performance decreases significantly, when not using mixup, even when the standard AdaCos loss is used as it is. Furthermore, the proposed changes to the AdaCos loss lead to significant improvements in terms of AUC and pAUC on the development set. For the evaluation set, AUC slightly increases and pAUC slightly decreases. A possible explanation for this

TABLE III
MEAN AUCS AND PAUCS OBTAINED WITH MIXUP AND THE MODIFIED ADACOS LOSS, BUT WITHOUT USING SUB-CLUSTERS, ON THE DEVELOPMENT AND EVALUATION SET.

mixup	modified AdaCos	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
		86.96%	80.68%	89.63%	82.47%
	✗	diverges	diverges	diverges	diverges
✗		89.13%	82.59%	91.43%	**84.47%**
✗	✗	**91.60%**	**85.01%**	**91.64%**	83.93%

TABLE IV
MEAN AUCS AND PAUCS OBTAINED WITH THE MODIFIED SUB-CLUSTER ADACOS LOSS ON THE DEVELOPMENT AND EVALUATION SET.

number of sub-clusters	backend	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
1	Gaussian	91.60%	85.01%	91.64%	83.93%
2	Gaussian	90.97%	82.54%	92.08%	85.08%
4	Gaussian	91.54%	83.53%	92.62%	84.31%
8	Gaussian	91.61%	85.24%	92.99%	85.74%
16	Gaussian	91.85%	85.61%	93.98%	88.27%
32	Gaussian	92.22%	85.69%	94.56%	87.51%
64	Gaussian	91.39%	83.58%	93.85%	85.43%
1	GMM	91.60%	85.01%	91.64%	83.93%
2	GMM	91.07%	82.70%	92.20%	85.60%
4	GMM	91.67%	83.70%	92.64%	84.35%
8	GMM	91.85%	85.46%	93.13%	86.07%
16	GMM	92.10%	85.84%	94.08%	**88.59%**
32	GMM	**92.57%**	**86.37%**	**94.69%**	87.90%
64	GMM	92.03%	84.06%	94.16%	86.19%

behavior is the randomness involved in training a neural network. Overall, the modifications still seem to improve the performance. When using the modified AdaCos loss and no mixup, the loss diverges because $\hat{s}^{(t)}$ grows exponentially. A proof can be found in the appendix.

E. Using sub-clusters for AdaCos

Now, further experiments are conducted to show that using sub-clusters inside the AdaCos loss improves the outlier detection performance. For this purpose, the neural network is trained with an increasing number of sub-clusters and evaluated with a single Gaussian and a GMM with modes equal to the number of sub-clusters as backends. The corresponding AUCs and pAUCs obtained on the development and evaluation set are shown in Tab. IV.

One can draw two main conclusions from the experimental results. First, using sub-clusters is highly beneficial, especially to increase the performance on the evaluation set. Note that by doing so, it is also possible to exclude potential outliers from the training data by removing small sub-clusters as done in [22]. Thus, this procedure is also well-suited for truly unsupervised anomaly detection problems instead of semi-supervised ones. A second conclusion is that using a GMM instead of a single Gaussian always improves the results because it can be fitted more accurately to the individual sub-clusters of the distribution.

TABLE V
MEAN AUCS AND PAUCS PER MACHINE TYPE OBTAINED WITH DIFFERENT REPRESENTATIONS ON THE DEVELOPMENT AND EVALUATION SET. WHEN USING THE COMBINED REPRESENTATIONS, ONLY THE LEARNED REPRESENTATIONS ARE USED, EXCEPT FOR MACHINE TYPE TOYCONVEYOR WHERE THE MEAN IS USED INSTEAD.

representation	machine type	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
mean	fan	80.73%	66.16%	95.32%	80.62%
max	fan	64.59%	51.48%	78.98%	57.70%
learned	fan	**87.61%**	**77.93%**	**97.60%**	**93.24%**
mean	pump	82.99%	68.50%	88.24%	70.36%
max	pump	70.13%	59.17%	68.96%	55.08%
learned	pump	**94.71%**	**88.91%**	**96.76%**	**88.30%**
mean	slider	87.46%	63.95%	72.16%	53.08%
max	slider	93.69%	76.69%	90.55%	70.65%
learned	slider	**99.55%**	**97.63%**	**97.61%**	**89.46%**
mean	valve	55.59%	50.23%	54.86%	52.09%
max	valve	98.54%	93.08%	96.35%	88.18%
learned	valve	**98.63%**	**94.62%**	**98.81%**	**95.80%**
mean	ToyCar	94.10%	80.94%	91.54%	76.87%
max	ToyCar	68.36%	53.85%	70.31%	54.90%
learned	ToyCar	**96.37%**	**91.64%**	**95.99%**	**91.93%**
mean	ToyConveyor	**85.78%**	**67.76%**	**91.74%**	**78.13%**
max	ToyConveyor	57.51%	50.39%	65.40%	53.06%
learned	ToyConveyor	73.89%	61.22%	81.37%	68.64%
mean	all	80.91%	66.19%	82.31%	68.52%
max	all	76.25%	64.71%	78.42%	63.26%
learned	all	92.57%	86.37%	94.69%	87.90%
combined	all	**94.21%**	**87.13%**	**96.42%**	**89.24%**

F. *Utilizing simple representations*

Instead of learning representations of the data by training a neural network, one can also use simple representations directly derived from the data. These representations have the advantage that extracting them does not require any training and they possibly contain useful information about the data that is not needed to discriminate among the classes and thus not contained in the trained representations. On the other hand, it is by no means ensured that these representations contain any useful information for detecting anomalous data at all. In this work, the mean and maximum of the log-Mel frequency bins over time are investigated as alternative representations. For this purpose, their performance is evaluated by estimating their distribution with a single Gaussian component each and using the resulting log-probabilities to detect outliers. The results can be found in Tab. V.

First, it can be seen that for most machine types, except "ToyConveyor", the trained representations perform best. But for some cases, the simple representations work surprisingly well. Examples are the maximum values for machine type "valve" and mean value for "ToyConveyor", which even outperforms the learned representation. The reason is that the performance of the learned representation is much worse for "ToyConveyor" than for all other machine types. This behavior can be found for many of the submitted systems e.g. [12], [15] and the exact reasons are still unclear. Koizumi et al. [4] speculate that the normal samples belonging to different machine ids of "ToyConveyor" are very similar and thus discriminative approaches have difficulties in finding decision boundaries. The other way around, i.e. using the mean value for "valve" and the maximum values for "ToyConveyor" leads to very poor performance, close to random guessing, in both cases. This is the reason why one cannot simply concatenate all three representations to obtain a single representation for all machine types because it would significantly degrade the performance. The best overall performance is achieved, by using the mean representations for "ToyConveyor" and the learned representations for the other machine types.

G. *Comparing the performance to other published systems*

Last but not least, the presented approach is compared to the five highest-ranked systems submitted to task 2 of the DCASE challenge 2020. To our best knowledge, no more recent work has been published and thus these systems represent the state-of-the-art. The results can be found in Fig. 2 and Fig. 3. First and foremost, the presented system significantly outperforms every other published system, both in terms of AUC and pAUC. This can easily be seen by comparing its performance to the one of the winning system of the challenge [34]. The proposed approach has a higher score for every machine type, except "slider" where the performance is the same. Furthermore, for most systems there is at least one machine type where the performance drops significantly, compared to the other systems, whereas the presented approach performs reasonably well for all machine types.

Note, that all systems but the one submitted by Primus [36] consist of an ensemble of multiple, very different models whereas the presented system consists of just a single model. Thus, for completeness an ensemble consisting of multiple versions of the proposed approach, each trained with another number of sub-clusters, ranging from 2^0 to 2^6, is also included. The ensemble is realized by summing the log-probabilities of all GMMs belonging to the subsystems. As expected, this ensemble significantly outperforms the system based on a single model and reaches a mean AUC of 97% and a mean pAUC of 91.24%.

V. CONCLUSIONS AND FUTURE WORK

In this work, multiple changes to the standard AdaCos loss specifically aimed at learning lower-dimensional representations for anomalous sound detection are proposed. In experiments conducted on the DCASE 2020 dataset for "Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring", it is shown that Gaussians or more generally GMMs outperform other widely used backends, namely the direct output of the trained model, cosine similarity and PLDA. Furthermore, using mixup in combination with modified parameter computations for AdaCos further improves the obtained results. By also using multiple learned sub-clusters instead of a single one for each class, less restrictive distributions than a single Gaussian for the representations of the data are learned. As a result, an even higher anomalous sound detection performance is achieved. In an additional experiment, the learned representation is compared to simple

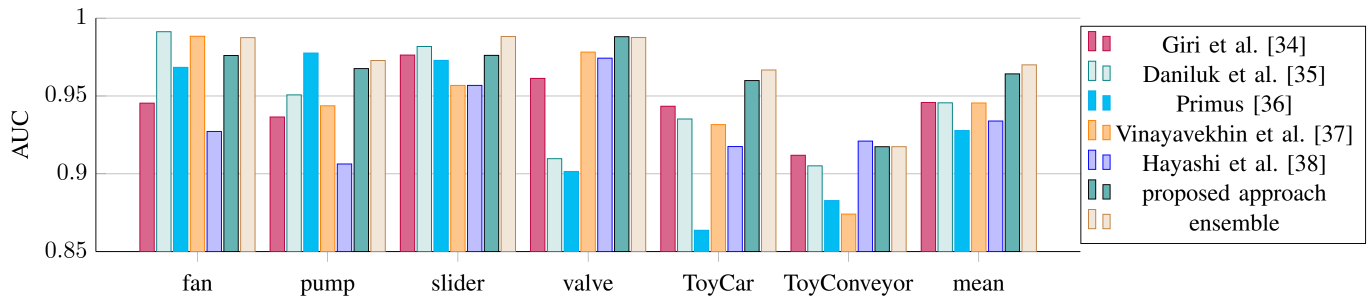

Fig. 2. Comparison of the AUCs obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE 2020 challenge task 2, the proposed approach and an ensemble. The ensemble consists of the sum of all log-probabilities given by GMMs belonging to trained models of the proposed approach with a different number of sub-clusters, ranging from 2^0 to 2^6.

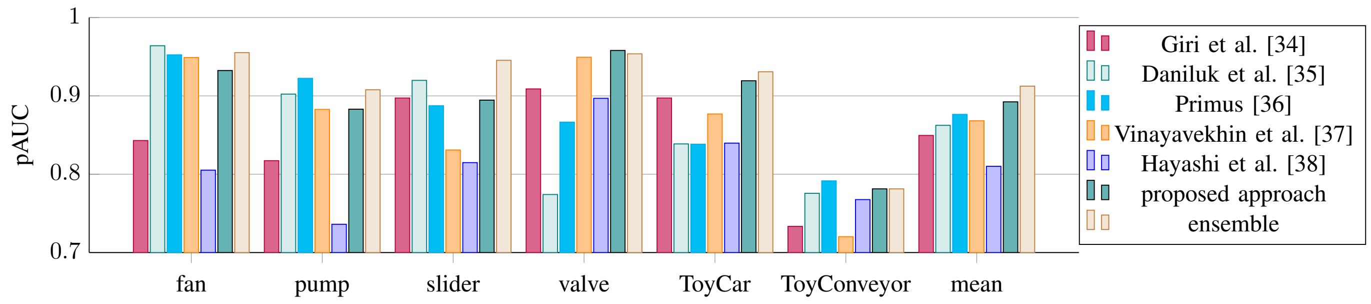

Fig. 3. Comparison of the pAUCs obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE 2020 challenge task 2, the proposed approach and an ensemble. The ensemble consists of the sum of all log-probabilities given by GMMs belonging to trained models of the proposed approach with a different number of sub-clusters, ranging from 2^0 to 2^6.

representations, namely the temporal mean and maximum of the log-Mel spectograms, and is shown to outperform them except for the machine type "ToyConveyor" where using the mean leads to the best results. Last but not least, the presented approach is shown to significantly outperform all other published systems on this dataset even when not ensembling multiple subsystems.

There are still some open questions to be answered. For the machine type "ToyConveyor", the performance of the learned representations is worse than simply taking the temporal mean of the log-Mel spectrogram. This shows that there is still room for improving the training process of the learned representations. One way to accomplish this could be using a self-supervised learning paradigm instead of training discriminatively among the known classes as done in [14]. Additionally, future work should also be aimed at clarifying why the performance for this particular class is worse to gain insights that may also be helpful for anomalous sound detection in general. Another way to further improve the performance of the presented model is to use more sophisticated approaches to mix samples than plain mixup. A collection of ways to mix samples can be found in [39].

References

[1] Pasquale Foggia, Nicolai Petkov, Alessia Saggese, Nicola Strisciuglio, and Mario Vento, "Audio surveillance of roads: A system for detecting anomalous sounds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 279–288, 2016.

[2] Yanxiong Li, Xianku Li, Yuhan Zhang, Mingle Liu, and Wucheng Wang, "Anomalous sound detection using deep audio representation and a BLSTM network for audio surveillance of roads," *IEEE Access*, vol. 6, pp. 58043–58055, 2018.

[3] Tomoki Hayashi, Tatsuya Komatsu, Reishi Kondo, Tomoki Toda, and Kazuya Takeda, "Anomalous sound event detection based on wavenet," in *26th European Signal Processing Conference (EUSIPCO)*. 2018, pp. 2494–2498, IEEE.

[4] Yuma Koizumi, Yohei Kawaguchi, Keisuke Imoto, Toshiki Nakamura, Yuki Nikaido, Ryo Tanabe, Harsh Purohit, Kaori Suefusa, Takashi Endo, Masahiro Yasuda, and Noboru Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.

[5] Charu Aggarwal, *Outlier Analysis*, Springer, 2nd edition, 2017.

[6] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4690–4699, IEEE.

[7] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10823–10832, IEEE.

[8] Sławomir Kapka, "ID-conditioned auto-encoder for unsupervised anomaly detection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 71–75.

[9] Koichi Miyazaki, Tatsuya Komatsu, Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, and Kazuya Takeda, "Conformer-based sound event detection with semi-supervised learning and data augmentation," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 100–104.

[10] Tadanobu Inoue, Phongtharin Vinayavekhin, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *Detection and Classification*

of Acoustic Scenes and Events Workshop (DCASE), 2020, pp. 66–70.

[11] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, "A discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 499–515.

[12] Jose A. Lopez, Hong Lu, Paulo Lopez-Meyer, Lama Nachman, Georg Stemmer, and Jonathan Huang, "A speaker recognition approach to anomaly detection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 96–99.

[13] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[14] Ritwik Giri, Srikanth V. Tenneti, Fangzhou Cheng, Karim Helwani, Umut Isik, and Arvindh Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 46–50.

[15] Qiping Zhou, "ArcFace based sound mobilenets for DCASE 2020 task 2," Tech. Rep., DCASE2020 Challenge, 2020.

[16] Simon J.D. Prince and James H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *11th International Conference on Computer Vision. ICCV*. IEEE, 2007, pp. 1–8.

[17] Kevin Wilkinghoff, "Using look, listen, and learn embeddings for detecting anomalous sounds in machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 215–219.

[18] Kilian Q. Weinberger and Lawrence K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Journal of machine learning research*, vol. 10, no. 2, 2009.

[19] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[20] Manli Zhu and Aleix M. Martinez, "Subclass discriminant analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1274–1286, 2006.

[21] Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj, "Robust fast subclass discriminant analysis," in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 1397–1401.

[22] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou, "Sub-center ArcFace: Boosting face recognition by large-scale noisy web faces," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 741–757.

[23] Harsh Purohit, Ryo Tanabe, Takeshi Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2019, pp. 209–213, New York University.

[24] Yuma Koizumi, Shoichiro Saito, Hisashi Uematsu, Noboru Harada, and Keisuke Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 313–317.

[25] Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[26] Sascha Grollmisch, David Johnson, Jakob Abeßer, and Hanna Lukashevich, "IAEO3 - combining OpenL3 embeddings and interpolation autoencoder for anomalous sound detection," Tech. Rep., DCASE2020 Challenge, 2020.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778, IEEE.

[28] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *30th International Conference on Machine Learning (ICML)*, 2013.

[29] Sergey Ioffe and Christian Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning (ICML)*, 2015, vol. 37, pp. 448–456.

[30] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[32] Aleksandr Sizov, Kong Aik Lee, and Tomi Kinnunen, "Unifying probabilistic linear discriminant analysis variants in biometric authentication," in *Proc. S+SSPR*. Springer, 2014, pp. 464–475, Software available at https://sites.google.com/site/fastplda/.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] Ritwik Giri, Srikanth V. Tenneti, Karim Helwani, Fangzhou Cheng, Umut Isik, and Arvindh Krishnaswamy, "Unsupervised anomalous sound detection using self-supervised classification and group masked autoencoder for density estimation," Tech. Rep., DCASE2020 Challenge, 2020.

[35] Pawel Daniluk, Marcin Gozdziewski, Slawomir Kapka, and Michal Kosmider, "Ensemble of auto-encoder based systems for anomaly detection," Tech. Rep., DCASE2020 Challenge, 2020.

[36] Paul Primus, "Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers," Tech. Rep., DCASE2020 Challenge, 2020.

[37] Phongtharin Vinayavekhin, Tadanobu Inoue, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," Tech. Rep., DCASE2020 Challenge, 2020.

[38] Tomoki Hayashi, Takenori Yoshimura, and Yusuke Adachi, "Conformer-based id-aware autoencoder for unsupervised anomalous sound detection," Tech. Rep., DCASE2020 Challenge, 2020.

[39] Cecilia Summers and Michael J. Dinneen, "Improved mixed-example data augmentation," in *Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 1262–1270, IEEE.

Appendix

Remark. *The adaptive scale parameter $\hat{s}^{(t)}$ of the modified AdaCos loss grows exponentially when not using mixup.*

Proof. After a few training iterations without mixup i.e. for $t > t_0 \in \mathbb{N}$, most training samples will have a very small angle to their associated class, i.e. $\theta_{i,y_i} \approx 0$. Therefore, $\cos\theta_{i,y_i} \approx 1$ and thus also $\cos\hat{\theta}^{(t)}_{\text{med}} \approx 1$. Furthermore, as empirically shown in [7], on average $\theta_{i,k} < \frac{\pi}{2}$ and thus $\cos\theta_{i,k} > 0$ for most $k \neq y_i$. Hence, by using the fact that the logarithm is a concave function and applying Jensen's inequality we obtain

$$\begin{aligned}\hat{s}^{(t)} &= \frac{f^{(t)}_{\max} + \log \hat{B}^{(t)}_{\text{avg}}}{\cos\left(\min(\frac{\pi}{4}, \hat{\theta}^{(t)}_{\text{med}})\right)} \\ &\approx \log\left(\frac{1}{N}\sum_{i\in\mathcal{N}^{(t)}}\sum_{k=1}^{CS}\exp\left(\hat{s}^{(t-1)}\cdot\cos\theta_{i,k}\right)\right) \\ &\geq \frac{1}{N}\sum_{i\in\mathcal{N}^{(t)}}\sum_{k=1}^{CS}\hat{s}^{(t-1)}\cdot\cos\theta_{i,k} \\ &\approx \hat{s}^{(t-1)}\underbrace{\left(1+\frac{1}{N}\sum_{i\in\mathcal{N}^{(t)}}\sum_{\substack{k=1\\k\neq y_i}}^{CS}\cos\theta_{i,k}\right)}_{>0}\end{aligned} \tag{10}$$

showing that $\hat{s}^{(t)}$ grows exponentially when not using mixup. Note that this inequality does not hold if only mixed-up samples are used for training. The reason is that most samples belong to multiple classes and thus do not have an angle of approximately 0 to their corresponding class mean because AdaCos increases the margin between classes.