An Ensemble Approach to Unsupervised Anomalous Sound Detection

Jahangir Alam, Gilles Boulianne, Vishwa Gupta, Abderrahim Fathan

Computer Research Institute of Montreal, Canada

jahangir.alam,gilles.boulianne,vishwa.gupta,abderrahim.fathan@crim.ca

Abstract

The task of anomalous sound detection (ASD) is to determine whether an observed sound is anomalous or normal. Both supervised and unsupervised approach can be adopted for the ASD task. In supervised approach anomalous and normal data are used in training whereas in unsupervised approach only normal data are used for training. In this work, we provide an overview of the systems developed for the task 2 i.e., unsupervised detection of anomalous sounds for machine condition monitoring, of the DCASE 2020 challenge. We employ various handcrafted local representations from the short-time spectral analysis of sounds. We also use fisher vector encoding - a learned global representations obtained from local representations of sound. Autoencoder variants and copy detection approaches are applied on the top of local representations and a standard GMM classifier is used with fisher vector encodings for unsupervised detection of anomalous sounds.

Index Terms: unsupervised anomaly detection, DCASE 2020 Challenge, autoencoder, GMM, copy detection, fisher vector.

1. Introduction

Anomalous sounds are referred to as sound data that deviate significantly from the majority of normal sound data. The objective of anomalous sound detection is to distinguish anomalous sounds from the normal sounds. Detection of anomalous sound has received much attention recently as it has various important applications such as audio surveillance, animal husbandry, product inspection, predictive maintenance and monitoring of machine condition. Prompt detection of machine anomaly by observing its sounds can possibly prevent propagation of damage, decrease the number of defective products and can be useful for monitoring the machine condition for predictive maintenance.

Based on the availability of anomalous data in the training phase ASD task can be categorized as supervised ASD and unsupervised ASD. Compared to supervised ASD the unsupervised ASD is much more realistic as it does not use anomalous sound data in training time. This is because making and/or collecting an exhaustive set of anomalous sounds is impossible. Besides, in real-world factories, actual anomalous sounds occur rarely and are very diverse. Considering the above, this year, the DCASE (Detection and Classification of Acoustic Scenes and Events) challenge organizers introduced a very challenging task known as "unsupervised-ASD". This indicates that the participants have to detect anomalous sounds that are not seen in the training data.

Various supervised and unsupervised approaches have been investigated for the detection of anomalous sounds. Many earlier works focused on the extraction of discriminative sound features by effective capturing of artifacts present in the anomalous sounds. Different feature-extractor-optimization methods have been studied in the past. In earlier studies, various statistical models such as GMM (Gaussian Mixture Models), SVM (Support Vector Machine) were used for computing anomaly scores. Some recent studies used deep learning based techniques such as autoencoders, variational autoencoders, one class neural networks, and generative adversarial networks (GANs) for sound-based anomaly detection. In the case of autoencoders, it is trained to minimize the reconstruction error of the normal training data, and the anomaly score is calculated as the reconstruction error of the observed sound. Thus, the autoencoder yields small anomaly scores for normal sounds and large scores for unseen anomalous sounds as these are not used in training.

In this work, for the challenging task of unsupervised anomalous sound detection, we employ different local (i.e., frame level) and global (i.e., sound level) representations of sound and use autoencoder variants and standard GMM classifier for calculating anomaly scores from the observed sounds in the development and evaluation test sets. The following local representations are used by short-time spectral analysis of the sounds: Log Mel-frequency spectrum, Mel-frequency cepstral coefficients (MFCC), Linear frequency cepstral coefficients (LFCC), product spectral cepstral coefficients (PSCC), and modulation spectrum. Fisher vector encodings extracted on top of LFCC features are used as global representations. A standard GMM classifier is used with fisher vectors for computing anomaly scores. With local features, various deep learning-based approaches such as simple autoencoder (AE), variational autoencoder (VAE), conditional VAE, and LSTM autoencoder (LAE) are employed to calculate anomaly scores. We also adopt the copy detection algorithm proposed in [1] for unsupervised anomalous sound detection (UASD) task.

2. Dataset and Task

2.1. DCASE 2020 Challenge Task 2 Database

The database used for task 2 of DCASE 2020 challenge [2] was derived from the ToyADMOS [3] and the MIMII [4] corpora. The entire database, comprised of normal/anomalous operating sounds of six types of toy/real machines, is divided into development (train and test) and evaluation (train and test) subsets. Training sets contain only normal data and only the labels (normal versus anomalous) of development test set were released to the participants. Among the six toy/real machines, the ToyCar and Conveyor machines database were taken from ToyADMOS, and valve, pump, fan, and slider database came from MIMII Dataset. Each recording is a single-channel having duration of approximately 10-sec that includes both a target machine's operating sound and environmental noise. Each machine type (e.g., ToyCar) has several individual machines with separate identifiers known as machine IDs. Machine types and machine IDs for both development and evaluation sets were released to the participants. For more details about the DCASE 2020 challenge Task 2 data the readers are referred to [2, 3, 4]

2.2. Challenge Task

In this new task, no anomalous sounds data were made available for training the model. So, the main challenge of this task is to identify unknown anomalous sounds under the condition that only normal sound samples can be used in training.

3. Representations of Sound

3.1. Local Representations of Sound

By local representations of sound we mean acoustic features typically extracted at 10 ms (or several multiples of it) intervals and designed to detect artifacts in the anomalous sounds. In this work, the following local representations are used by short-time spectral analysis of the sounds: Log Mel-frequency spectrum, Mel-frequency cepstral coefficients (MFCC), Linear frequency cepstral coefficients (LFCC), product spectral cepstral coefficients (PSCC), and modulation spectrum [5, 6, 7, 8, 9, 10].

3.2. Global Representations of Sound

By global representations we mean utterance level embeddings extracted using the short-time features. Here, we use fisher vector encodings as the global representations of sounds. The Fisher vector (FV) encoding was originally introduced and is popularly used in computer vision applications, especially in large scale image retrieval [11, 12, 13]. The FV with cascaded non-linear normalization has been further applied to classify the eating condition of a speaker from her/his recording for the Interspeech 2015 computational paralinguistic challenge [14]. In [15], we used fisher vector encoding-based global representation for voice anti-spoofing task.

The main idea behind such encoding is to measure the amount of change induced by the utterance/video descriptors on a background probability model, which is typically a Gaussian Mixture Model (GMM). Fisher vector encodes the amount of change of model parameters to optimally fit the new-coming data. This requires the computation of the Fisher information matrix, which is the derivative of the log-likelihood with respect to model parameters. FV require a smaller number of components in a GMM when compared to i-vectors.

The FV encoding assumes that descriptors are generated by a GMM model with a diagonal covariance matrix. Initially, a K-Gaussians GMM is trained. The GMM model is parameterized as $\lambda = \{w_k, \mu_k, \sigma_k\}_{k=1}^{K}$, where w_k, μ_k, and σ_k represent the mixture weights, mean, and variance corresponding to the k-th Gaussian component, respectively. Once the model is trained, the FV representation of a set of local descriptors $\{x_1, x_2, ..., x_N\}$ will be given by [13]:

$$\mathbf{u}_k = \frac{1}{N\sqrt{w_k}} \sum_{i=1}^{N} q_{ki} \Big(\frac{x_i - \mu_k}{\sigma_k} \Big), \tag{1}$$

$$\mathbf{v}_k = \frac{1}{N\sqrt{2w_k}} \sum_{i=1}^{N} q_{ki} \left[\Big(\frac{x_i - \mu_k}{\sigma_k} \Big)^2 - 1 \right], \tag{2}$$

where q_{ki} corresponds to the soft assignment of descriptor x_i to the k-th Gaussian.

The **u** and **v** parts capture the 1st and 2nd order differences, respectively. With a d-dimensional local descriptor, the final representation of size $2dK$ is obtained by concatenating **u** and **v**, which will usually yield FV encoding of relatively high dimensionality. Using a $K = 32$ Gaussian components GMM along with 90-dimensional local descriptors, for instance, will yield FV with the final dimension given by $2 * 90 * 32 = 5760$. The principal component analysis (PCA) algorithm can be applied on the raw FV encoding in order to reduce its dimension. The PCA projection matrix can be obtained on the training data. Power normalization followed by L2-normalization are then applied [11, 12, 13, 14]. While power normalization helps to reduce the sparsity of the descriptors, L2-normalization aids in improving prediction performance. We utilize a component-wise power normalization with $a = 0.5$ as:

$$f(x) = \text{sign}(x)|x|^a. \tag{3}$$

For UASD task, we train six GMMs, one for each machine type, and then extract fisher vectors from all samples of a machine type using the GMM of that machine type. No dimension reduction is performed but the power normalization followed by L2-normalization is applied.

4. Approaches adopted for UASD Task

4.1. VAE & Conditional VAE

Variational autoencoders (VAEs) have been shown to outperform conventional autoencoders in anomaly detection [16]. Here we used a β Variational AutoEncoder (β-VAE, [17]) with a general architecture similar to the baseline autoencoder. The encoder and decoder are composed of 4 fully connected layers with ReLU activation, batch normalisation and skip connection. But in contrast to the autoencoder, the decoder input is not connected to the encoder output; its input is an embedding sampled from a Gaussian distribution with diagonal covariance $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$. Each parameter $\mu_\theta, \sigma_\theta$ is obtained by applying a dense linear layer to the encoder output. The loss function also differs from the baseline autoencoder. In addition to the reconstruction loss, a divergence term is added:

$$\beta \cdot \mathrm{D}_{KL}[\mathcal{N}(\mu_\theta, \sigma_\theta^2) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})] \tag{4}$$

where D_{KL} is the Kullback-Leibler divergence between the embedding distribution and a zero-mean, unit-variance Gaussian prior, and β is a weight that is used to strike a balance between reconstruction error and divergence from the prior. This additional term acts as a regularizer in learning, to encourage better generalization.

The encoder input and decoder output layers have the same dimension as the original input. All hidden layers have dimension 256, except the embedding (bottleneck) layer which has size 8. The total number of trainable parameters is 733,840. The model is trained for 100 epochs, using an Adam optimizer with default values. A separate model is trained for each machine type.

For Conditional Variational AutoEncoder (CVAE, [18]) we use here a β-VAE which takes two inputs: features and machine label. An embedding layer is applied to the categorical label and its output is concatenated with the features before being fed to the encoder input layer. One version uses only machine type labels and the other one uses a combination of machine type and machine id. A larger, single model is trained with data pooled from all machines, with hidden layers of dimension 384, bottleneck of size 8, machine label embedding of size 128, for a total of 1,545,100 trainable parameters, for 50 epochs.

4.2. Dense AE & LSTM AE

The baseline model (Dense AE) provided by the organizers, is a simple autoencoder based on 9 block layers of a Dense layer,

followed by Batch Normalization, then a Relu Activation. With a final output dense layer of the same original input dimension. This AE is trained to minimize the reconstruction error of the normal training data.

Using the same baseline features, we implemented an LSTM-based AE to take into account the sequential nature of the audio data. This LSTM AE is based on an encoder and a decoder, both of which comprise 2 layers of LSTM cells, with frames as the number of timesteps. The total number of trainable parameters is 299384.

4.3. GMM-based UASD

On the top of fisher encoding-based global representations, we use a Gaussian Mixture Model (GMM) classifier for attaining anomaly score of an observed sound. Six GMMs are first trained, one for each machine type, using the fisher vectors. For a given fisher vector of a specific machine type we compute the anomaly score (weighted log probabilities for each sample) using GMM of that machine type.

4.4. Copy Detection Approach for UASD

In copy detection algorithms [1], we match a new snippet of audio (query audio) with reference audio files to see whether the query audio is a copy of a reference audio. To do this, we create fingerprints for the reference audio and search for the query fingerprints in the reference audio. In doing so, we find a sequence of fingerprints in the reference that match query frames time-synchronously. The match likelihood between the query and the reference increases with the total number of frame-synchronous fingerprint matches between the query and the reference. The maximum count of these frame synchronous matches correspond to the best matching reference to this query, and the matching count acts as the likelihood of the match. Good matches in general correspond to over 10% of the query frames. The best matching reference frame for each query frame is the reference frame with the lowest distance (Euclidean distance, absolute distance etc.).

When we applied the copy detection algorithms to the anomalous sounds detection (ASD), the number of frame synchronous matches were very small, around six or seven, for both anomalous and normal sounds. The reason is that most of the sounds are repeated, and there is no real sequence of distinct sounds as it is for speech. So even the six or seven synchronous matches are probably random matches.

So we varied the copy detection algorithm to create 20 reference templates for each machine (by machine we mean each individual machine: four different fans, four different pumps, etc.) . For each machine we created 20 templates, and the templates were created from only the training data for that machine. We tried 2 different algorithms to create the templates.

In one algorithm, we use the features (MFCC's or filterbank features) from first 20 training files to generate 20 templates. We take the remaining training files from the training data for that machine and compute the best matching frames to these first 20 training files for creating these templates. For example, if there are N training files in the training set for the machine and each training file has M frames, then first 20 are used to create the reference templates. Files 21 through N are used to generate match statistics for matches with these 20 training files. The features of each training and test file are mean and variance normalized before use. If we assume there are F feature frames in each of the training files, then these statistics are generated as follows:

1. for each training file i from tr(1) to tr(20) do;
2. for each training file j from tr(21) to tr(N) do;
3. for each frame k in tr(j) from k=1 to F do;
4. compute d(tr(i), tr(j)(k)) = min dist(tr(j)(k), tr(i)(L)), L=1,...,F;
5. end do; end do;
6. compute mean d(i,k) of (d(tr(i),tr(j)(k), j=21,...,N.) For each frame k
7. mean d(i,k), k=1,...F becomes a reference template i

In other word, as shown above, training files 21 through N are compared with the features of the first 20 training files to create the 20 templates.

For testing, we take each file in the development set (test file) and find the closest matching frames to the training files in the same fashion as during training. For example, for each frame of the test file, we find the distance to the closest matching frame of training file 1. These closest matching scores are then compared to the averaged scores for template one we computed above. There are different ways of coming up with the match between the development file and the 20 reference templates. One way is to sum the differences between all the test scores and the averaged scores of the corresponding frames of the template. The higher the accumulated sum, the worse is the match. So this score is the anomaly score associated with that test file. Another way is to count the number of frames that have test scores higher than the averaged scores in the reference template. The higher this score, the better is the likelihood of an anomalous sample. Another way would be to get the sequence of score differences, compute the mean and variance of this sequence, and find the number of frames that are above one or two standard deviation. The higher this number, the higher the likelihood of anomalous detection.

We tried the above three ways of computing scores for all the machines in the development set. It turned out that associating scores to anomalous behavior was tricky. For example, when a valve is working correctly, it makes loud clicking sounds, while these clicking sounds are weak when the valve is faulty. So higher scores should be associated with correctly working device. Same was true about many other machines. For example, some pumps make much more noise when they are working than when they are faulty. So for each machine, allocation of higher scores to anomaly/normal behavior was on a case by case basis. With proper correction, the results for some of these algorithms are shown in Table 1. In algorithm 1 (Euclidean, above sigma) in this table, we use 40-dimensional MFCC features, Euclidean distance for frame comparison, and for total anomaly score, we first compute the sequence of score differences between the test frames and the reference templates as outlined above. We then find the mean and standard deviation of this sequence. We then sum scores of all the score differences between the test and reference frames that are above mean+standard deviation. This sum becomes the anomaly score. In algorithm 2 (Euclidean, fbank, positive only), we use 40-dim filter-bank features with Euclidean distance for frame comparisons. In this algorithm, we find the number of frames of test which have distances greater than the corresponding average values for all the 20 reference templates. The total count of these anomaly frames becomes the anomaly score. Algorithm 3 (Euclidean, positive only) is same as algorithm 2, but uses 40-dimensional MFCC features. In algorithm 4 (250ms new), we use 250ms window to compute 40-dimensional MFCC features with 125ms frame advance. The

total anomaly score is the difference between the total number of test frame scores above the average values minus the total number of test frame scores below the average values.

alg	TC	TCon	fan	pump	slider	valve
1	0.48	0.5	0.53	0.59	0.61	0.76
2	0.56	0.51	0.5	0.62	0.61	0.78
3	0.55	0.52	0.53	0.61	0.68	0.77
4	0.63	0.56	0.51	0.58	0.66	0.61

Table 1: *Average AUC results for different machines with different algorithms.*

Another algorithm for generating the 20 templates is similar to the previous algorithm, but it also generates stats for each frame. In this algorithm, for example, when we match training example 21 with training example 1, we get the best matching frame for training example 1 with each frame of training example 21, and then generate stats for the frames of training example 1 from all the frames from training example 21 thru training example N that match the training example 1 frames. Here also, the features of each training and test file are mean and variance normalized before use. The pseudo algorithm goes as follows:

1. for each training file i from tr(1) to tr(20) do;
2. for each training file j from tr(21) to tr(N) do;
3. for each frame k in tr(j) from k=1 to F do;
4. compute d(tr(i)(L1), tr(j)(k)) = min dist(tr(j)(k), tr(i)(L)), L=1,...,F;
5. end do; end do;
6. compute mean and standard deviation of d(tr(i)(L1,*) from all frames k of training examples j that match frame L1 of training example i.
7. Each frame L of training examples i gets a mean and a standard deviation that is stored as a template.

In first algorithm, we accumulate statistics for each frame based on matches with frame numbers of training examples 21 thru N. In this algorithm, we accumulate statistics for each frame based on matches with frame numbers of training examples 1 thru 20. So the template has the average and standard deviation of each frame of training example 1 for all such matches with the frames of training examples 21 through N.

During the comparison, We get the closest matching frame of training example 1 for each frame of the test audio from the development set. The template mean for this matching reference frame is then subtracted and normalized by dividing with standard deviation. The resulting scores are summed for all the frames to give the total anomaly score. This is done for all 20 templates. Also different strategies like in the previous algorithm can be applied for generating the anomaly scores. The results for a few different ways of generating anomaly scores are shown in Table 2. The first algorithm (NN-sigma-norm) uses 40-dimensional MFCC features and sums all score differences after normalization with standard deviation as described above. The second algorithm (NN-sigma-norm-1) sums the score differences without normalization with standard deviation. The third algorithm (NN-sigma-norm-fbank) uses filter-bank features with normalization as in the first algorithm.

alg	TC	TCon	fan	pump	slider	valve
1	0.61	0.57	0.52	0.61	0.72	0.73
2	0.59	0.56	0.54	0.60	0.72	0.80
3	0.56	0.51	0.53	0.55	0.59	0.69

Table 2: *Average AUC results for different machines with alternative way of computing reference templates and with three different algorithms for generating anomaly scores.*

If we just look at the best average scores for each machine, then the second set of algorithms give better AUC scores than the first set of algorithms. We also tried to combine the anomaly scores for different algorithms. The combination is by first normalizing the scores for each algorithm, and then combining them either with equal weight or with weights based on AUC values. In either case, we could never improve the AUC values above the best value for each machine. So the best strategy so far for each machine seems to be to pick the algorithm that gives the best AUC or AUC+pAUC value.

5. Ensemble of systems

We tried many different systems with different feature parameters. So the question was which systems to submit. We had a limit of 4 systems to submit. So we prepared these 4 submissions based on the development machine results. The following four systems were submitted:

1. The best overall system based on overall average AUC over all the machines. During training, the AUC was better when trained with dev set alone (instead of dev + eval set training). So this system was only trained on the eval set training data for submission. This is denoted as sub1.
2. The best overall system per machine based on Average AUC per machine. By machine we mean fan, slider, pump, etc. So there could be six different systems involved since there are six machines. Here also, each system was trained on the eval set training data alone. We denote this system as sub2.
3. The best overall system per machine based on average AUC per machine. So the same systems are involved as in the previous submission. However, this time, each system is trained on combined dev and eval training sets. We denote this one as sub3.
4. We combined results from top 3 systems for each machine. So for each machine, the top 3 systems could be different. The combination was by first normalizing the score for each system (subtract average and divide by standard deviation), and then adding the scores. No weighting was applied since we observed that even for the same machine type (like fan), the scores vary a lot for each individual machine in that machine type. When combining the scores for the dev set machines, we achieved better AUC scores for fan, pump, ToyCar and ToyConveyor, while the AUC scores for slider and valve were slightly worse. This combination is denoted as sub4.

The average AUC scores for the dev set for each machine type for the four different submissions are shown in Table 3. In

this Table, the first row shows results for the baseline system. So our progress can be estimated by comparing the corresponding baseline averages for each machine to our submission averages for the dev set.

system	TC	TCon	fan	pump	slider	valve
Baseline	0.78	0.72	0.67	0.73	0.84	0.67
sub1	0.80	0.59	0.72	0.78	0.91	0.96
sub2	0.81	0.75	0.72	0.78	0.94	0.96
sub3	0.81	0.75	0.72	0.78	0.94	0.96
sub4	0.83	0.75	0.72	0.78	0.94	0.95

Table 3: *Average AUC results for the dev set for each machine for each submission. Submissions sub2 and sub3 have the same dev set scores as the submissions differ only in the training set (eval set versus dev+eval set). The first row shows the baseline results. TC and TCon are acronyms for ToyCar and ToyConveyor, respectively.*

It is observed from Table 3 that with all our submissions we were able to achieve better performance than the Baseline on machine types of the development test set with one exception for sub1 which provided worse result on ToyConveyor machine. Our best single system is based on modulation spectrum and is denoted in this table as sub1. As a pre-processing step this system employed speech enhancement for removing the effects of environmental noise. The submitted system sub4 outperformed all the systems reported in Table 3. Systems sub2 and sub3 demonstrated the second best results.

6. Conclusions

In this work, we presented details of our systems developed for the unsupervised anomalous sound detection task of the DCASE 2020 challenge. Detection of anomalous sound in this fashion may be helpful for machine condition monitoring. Various handcrafted local representations from the short-time spectral analysis of sounds and fisher vector encoding - a learned global representations obtained from local representations of sound were employed. In addition to a simple Autoencoder we also employed variational autoencoder and conditional variants of it to compute anomaly scores. Here, we proposed to use copy detection approaches over the local representations of sound for unsupervised anomaly detection. Fisher encoding-based system with a standard GMM classifier provided excellent performance on valve and slider machine types. Our best single system was based on modulation spectrum - based on local acoustic features.

7. Acknowledgements

The authors would like to thank Ministry of Economy and Innovation (MEI) of the Government of Quebec for the continued support.

8. References

[1] V. Gupta, G. Boulianne, and P. Cardinal, "CRIM's content-based audio copy detection system for TRECVID 2009." *Multimed Tools Appl*, vol. 60, pp. 371–387, 2012.

[2] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda *et al.*, "Description and discussion on dcase2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," *arXiv preprint arXiv:2006.05822*, 2020.

[3] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "Toyadmos: A dataset of miniature-machine operating sounds for anomalous sound detection," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 313–317.

[4] H. Purohit, R. Tanabe, K. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "Mimii dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," *arXiv preprint arXiv:1909.09347*, 2019.

[5] M. J. Alam, P. Kenny, V. Gupta, and T. Stafylakis, "Spoofing detection on the asvspoof2015 challenge corpus employing deep neural networks," in *Proc. Odyssey*, 2016, pp. 270–276.

[6] M. J. Alam, P. Kenny, G. Bhattacharya, and T. Stafylakis, "Development of crim system for the automatic speaker verification spoofing and countermeasures challenge 2015," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[7] J. Alam and P. Kenny, "Spoofing detection employing infinite impulse response—constant q transform-based feature representations," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 101–105.

[8] M. J. Alam, G. Bhattacharya, and P. Kenny, "Boosting the performance of spoofing detection systems on replay attacks using q-logarithm domain feature normalization," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 393–398.

[9] M. Sahidullah, T. Kinnunen, and C. Hanilçi, "A comparison of features for synthetic speech detection," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[10] T. Falk, C. Zheng, and W. Chan, "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1766–1774, 2010.

[11] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in neural information processing systems*, 1999, pp. 487–493.

[12] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[13] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3384–3391.

[14] H. Kaya, A. A. Karpov, and A. A. Salah, "Fisher vectors with cascaded normalization for paralinguistic analysis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[15] J. Alam, "On the use of fisher vector encoding for voice spoofing detection." in *13th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI)*, https://doi.org/10.3390/proceedings2019031037, pp. Toledo, Spain, December 2–5, 2019.

[16] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *Proc. ICLR*, 2017, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[18] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-Supervised Learning with Deep Generative Models," in *Proc. NIPS*, 2014, pp. 3581–3589. [Online]. Available: http://arxiv.org/abs/1406.5298