AML - Challenge 2: Anomaly detection

Francesco Dente, Michele Ferrero, Niccolò Zanieri

May 2024

1 Introduction

DCASE is a challenge that the *IEEE Signal processing society* created to stimulate the research in the sound scenes recognition field. As described in the challenge's original website[1], being able to extract correct information from sound recordings of complex environments has huge potentials in a variety of applications. The focus of our work is on the application of machine learning techniques to accomplish detection of anomalous functioning of machines based on the sounds they produce. Particularly, we analyze how well unsupervised learning algorithms like *Autoencoders*, *GMMs* and *Diffusion models* to accomplish the above described task.

2 Data Analysis

2.1 Dataset

The dataset we worked on is completely composed of **.wav** files, each one of them containing a 10 seconds long sound recording of a machine, in particular a slide rail either functioning correctly or displaying anomalous behavior. Each audio file is associated with its machine ID, specifically we have three different machine IDs in our dataset (0, 2, 4).

The whole dataset is split in two parts: a training dataset containing only recordings of machines correctly working and a test dataset, the latter containing a mixture of anomalous sound recordings and normal ones.

2.2 Data Preprocessing

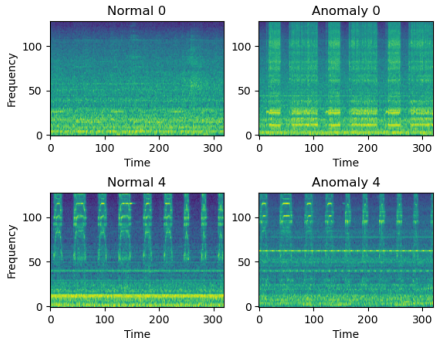
We now present the main preprocessing techniques we used in our work, more detailed information is described inside Section 3.

2.2.1 Spectrogram

In order to feed our samples to the models we decided to transform the time-domain representation given by the **.wav** files into a *spectrogram* representation. This transformation extracts frequency information from the signal and represents the sound with a matrix that associates time and frequency coordinates to an amplitude value which is represented via the Decibel scale. Using *spectrograms* leads to a smaller representation of data that is more suitable for models training, namely, it requires around one fourth of the features needed for the time-domain representation. We preferred this to the Fourier transform because, as described by *Tran and Lundgren*[7], using the latter would lead to a loss of frequency information in the time domain.

Listening to a few audio samples for each machine ID, the difference between anomaly and normal sounds is quite clear, but there may be some other frequencies not well perceived by humans that could help our models in the anomaly detection task. Therefore, we explore the use of both Mel[1] and Hz scale in our spectrogram to see which gives better results.

We will use this technique for our *Dense autoencoder*.



[1] the Mel scale provides our models with sound information similar to what a human would perceive as described in Section IV.B of [6]

Figure 1: Mel spectrograms representing a normal and an anomaly sample for two different machine IDs

2.2.2 Spectrogram augmentation

When dealing with audio data, a good practice to help models avoid overfitting on the training dataset is spectrogram augmentation.
In loading data for training we make sure to have some of the spectrogram's values randomly replaced with the mean value over the whole matrix. This practice is not applied when processing validation and test samples to avoid compromising the sample representation.

2.2.3 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) are a set of features commonly used in automatic speech and audio processing. The basic idea of MFCCs is to take the result of the Fourier transform of the logarithm of the magnitude spectrum of a signal to provide a representation of the rate of change in different spectral bands, combined with the use of the Mel scale. In this way, we can compactly represent the important characteristics of the sound and capture the perceptual properties relevant to human hearing. We will use this technique with *GMMs* in order to avoid feeding our model with high-dimensional data such as spectrograms.

3 Model selection

Now we will analyze the performance of different machine learning models for our task by using **AUC** as a metric to evaluate them on the test set. In this way, we can assess how our models are generally able to separate anomalies from normal sound, regardless of the chosen threshold.

We will explore three possible approaches for solving our task:

- **Deep approaches** using a *Dense Autoencoder* and *U-Net*
- **Non deep approach** using *Gaussian Mixture Models*

[2]MSE between input and output

3.1 Training procedure

Since the main challenge of this task is to detect unknown anomalous sounds under the condition that only normal sound samples have been provided as training data, we need to be careful in the training procedure to avoid **data leakage** from the test set containing also anomalous sounds. We split our training dataset of normal sounds into training and validation (80/20), maintaining the same proportions of each machine ID sound. We use the validation dataset to check for overfitting and to tune the hyperparameters of our models. However, we cannot use **AUC** on the test set because we only have access to normal data, so we have to use different metrics for each of our approaches. We then retrain our chosen model on both training and validation and evaluate it on the test set to assess its performance on the anomaly detection task.

3.2 Dense Autoencoder

Autoencoders [3] are a class of unsupervised neural networks that aim to encode input data into a lower-dimensional space and then reconstruct the input from this compressed representation. The network is designed to minimize the difference between the input and its reconstructed output. We assume that our model will be able to approximate a function $f(x)$ describing the distribution of normal data from our training data. We will then use the **reconstruction error**[2] between input and output data as a score to detect whether a sample is an anomaly or not.

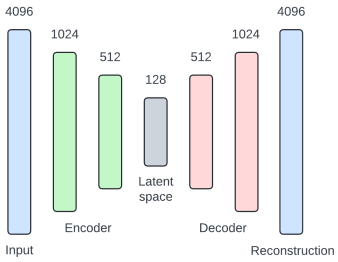


Figure 2: Proposed Dense AE architecture

3.2.1 Sub-windows splitting

In order to use the spectrograms inside our model we perform some further manipulations

on them at training and testing time:

- At **training time**, we divide the spectrogram of each sample into 10 non-overlapping windows of the same size and we use each one of them as a single sample for our model. By doing this, we manage to reduce the dimensionality of our samples while maintaining a good resolution in both the time and frequency dimensions of the spectrograms. We believe that 10 windows is a reasonable choice to maintain enough information of temporal context in each frame for our anomaly detection task. This procedure also increases our training set, which may help the training of our autoencoder.

- At **test time**, we divide the spectrogram of each sample into 10 non-overlapping windows and then take the maximum reconstruction error as the score for detecting the anomaly. This way, if the anomaly is located in a specific time interval in the 10 seconds of audio, we expect to have a higher reconstruction error than if we were to use the whole 10 seconds spectrogram as input.

3.2.2 Model validation

We use the reconstruction error[3] computed over the validation set to choose the size of the latent space and the scale for the spectrogram (whether Mel or Hz)[4] . Our assumption is that the more accurate the autoencoder is at reconstructing normal data (i.e., the lower the loss of our model), the higher the reconstruction error will be on anomalous data, thus increasing the **AUC** on our test set [2].

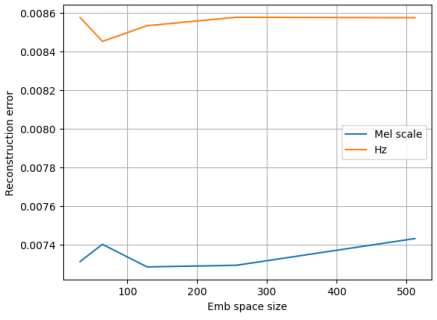


Figure 3: Reconstruction error obtained with different hyper parameters

Given the results we choose **Mel Scale** and a **latent space** of dimension **128** as best configuration for our model. We obtain a noteworthy score of **0.862 AUC** on test, meaning that our assumptions were effective.

3.3 GMM

A *Gaussian Mixture Model* [4] represents a probability distribution as a mixture of multiple Gaussian distributions. Each Gaussian component in the mixture represents a cluster of data points with similar characteristics.

We expect anomalous data to have low probability of being generated by the model with respect to normal points and we use the negative log-likelihood of a sample as the anomaly score.

3.3.1 Preprocessing pipeline

GMMs are very powerful methods for estimating data distributions, but we have to pay attention to the dimensionality of our data in order to correctly estimate the parameters of our model (especially the covariance matrices). Therefore, we do not use the *MFCCs* directly but we extract interesting features in order to obtain a more compact representation of the data (D=102), more suitable for our model. Particularly, together with **mean and variance of *MFCCs*** we use:

- **Delta and Delta-Delta coefficients**: to capture the temporal dynamics of the *MFCCs* (velocity and acceleration)

- **Spectral features**: we use *Centroid*, *Bandwidth*, *Contrast* and *Rolloff*.

- **Temporal Features**: we use *Zero Crossing Rate* and *Root Mean Square Energy*.

Also, considering the number of training samples available (2000), we expect to have a good estimation of the full covariance matrices only for a small number of components, while the use of tied and diagonal covariances may allow us to use a larger number of components[5] without overfitting the training data(this behaviour can be clearly seen in Figure 4).

[3]To make the metric comparable between the Hz and Mel scales, we first rescale our data to the [0,1] range.
[4]Results obtained after training the model for 50 epochs for each combination
[5]$O(KD^2), O(D^2), O(KD)$ free parameters respectively for full covariance, tied, diagonal approach, with K as the number of components.

It's important to note that we also denoise the data before extracting the *MFCCs* using **spectral gating**[6] and normalize it after obtaining the compact representation.

3.3.2 Model validation

We use the log-likelihood computed over the validation set to choose the number of components and the structure of the covariance matrix for our model. Again, our assumption is that the better the GMM estimates our distribution, the higher the anomaly score will be on anomalous data, thus improving the **AUC** on our test set. Furthermore, since initializations in the *EM algorithm* used in *GMMs* are random, to obtain reproducibility without setting a random state, we set the n_init parameter of scikit GaussianMixture to 10. By doing so, 10 initializations are performed and only the best results are retained.

We choose as the best configuration a GMM with **4 components** using **full covariance** matrices, which achieves a remarkable **AUC** score of **0.938** on the test set, meaning that our previous choices were effective.

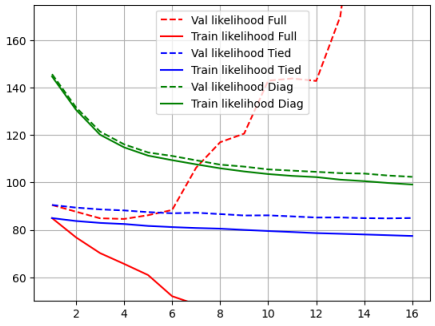


Figure 4: Negative log-likelihoods over training and evaluation dataset with different number of components

3.4 U-Net

As a last approach we explored diffusion modeling using a smaller version of the U-Net architecture [5] trained as image denoiser. Our aim is to let it specialize in reconstructing the noised images from the original distribution $p(x)$ and struggle at reconstructing images from different distributions (anomalies).

We use normal samples with Gaussian noise as input data after using the sub-window splitting described in section 3.2.1, and the *MSE* as loss for training the model.

We will use once again the *MSE* between input and output as an anomaly score.

With this method we get an **AUC** score of **0.742** on the test set, which is not that interesting compared to the other already described architectures.

3.4.1 Possible improvements

Now, let's explore several strategies that we believe could enhance the performance of this model:

- Given the high computing resources required by the model, we had to use a smaller version that could fit our GPU constraints. Relaxing these constraints would probably lead to better results.
- Try different types of noise added to the normal samples during training [8]

Even if our results are not very promising for this method, it might be worth it to go further with a more accurate research in the field.

3.5 Model Choice

	Dense AE	**GMM FC-4**
AUC test	0.867	0.938

We now compare the *AUC* of our two best models, and we see that the non deep model performs much better than the dense autoencoder. This is probably due to the small amount of training data available and the limited computing power available that prevent us to try many deeper architectures and to do more fine-grained hyperparameter tuning.

Since we cannot make a reasonable choice between the two models using metrics computed with only normal data, we choose **GMM FC-4**, which has a higher **AUC**, as the best model.

3.6 Post-evaluation analysis

We can now look at our previous choices to see if we obtained optimal, near-optimal, or suboptimal results by also exploiting information in the test set. For the sake of time (training deep architectures is time-consuming), we will only look at our choices for the GMM model, which turned out to be the best one.

[6] To take into account the possible environmental noise

By performing validation based on the *AUC* metric, we found out that using a lower number of features (D=36), particularly by eliminating the **Spectral Rolloff**, **Delta** and **Delta-Delta coefficients**, we are able to obtain an impressive score of **0.958 AUC**, using a GMM with **9 components** and **full covariance matrices**. This means that the dropped features did not carry any useful information for our task, but were only making it hard for the model to correctly estimate the parameters of full covariance matrices when having a higher number of components, as we can see in figure 5.

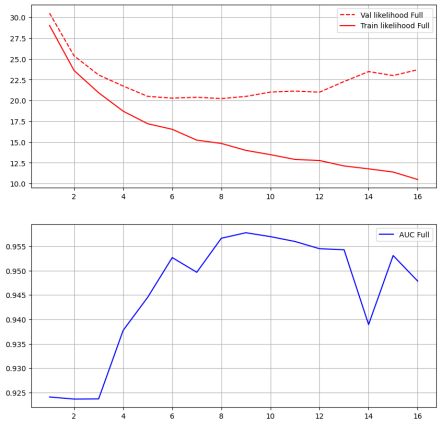


Figure 5: Negative loglikelihoods over evaluation dataset (above) and *AUC* on test set (below) for full covariance GMM with different number of components

4 Conclusions

In this report, we have explored various machine learning models for the task of anomaly detection in machine sounds.

We evaluated three different approaches: a Dense Autoencoder, a Gaussian Mixture Model and a U-Net architecture. Among these, the GMM with full covariance matrices and four components performed the best, achieving an AUC score of 0.938 on the test set. This model outperformed the Dense Autoencoder, which scored 0.867 AUC, and the U-Net, which had a 0.742 AUC.

We also experienced how not having access to anomalous data can be limiting when it comes to hyper-parameters tuning and model evaluation in an anomaly detection task.

In conclusion, our study demonstrates that GMMs are particularly effective for anomaly detection in audio data, especially when combined with careful preprocessing and feature extraction. The results highlight the potential of non-deep learning methods in scenarios with limited training data and computational resources.

References

[1] DCASE challenge 2020. . Last accessed 23 May 2024.

[2] Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring. Last accessed 29 May 2024.

[3] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[4] Douglas Reynolds. *Gaussian Mixture Models*, pages 659–663. Springer US, Boston, MA, 2009.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[6] Karam Sahoo, Ishan Dutta, Muhammad Fazal Ijaz, Marcin Wozniak, and Pawan Singh. Tlefuzzynet: Fuzzy rank-based ensemble of transfer learning models for emotion recognition from human speeches. *IEEE Access*, PP:1–1, 12 2021.

[7] Thanh Tran and Jan Lundgren. Drill fault diagnosis based on the scalogram and mel spectrogram of sound signals using artificial intelligence. *IEEE Access*, 8:203655–203666, 2020.

[8] Julian Wyatt, Adam Leach, Sebastian M. Schmon, and Chris G. Willcocks. Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 649–655, 2022.