Anomalous Sound Detection

LEONE, Andrea
Politecnico di Torino
Institut EURECOM
leone@eurecom.fr

BONINO, Giulia
Politecnico di Torino
Institut EURECOM
bonino@eurecom.fr

NEPOTE, Luca
Politecnico di Torino
Institut EURECOM
nepote@eurecom.fr

August 29, 2024

Abstract

This project addresses the task of anomalous sound detection, which is part of a bigger challenge for the signal processing community, dealing with the "Detection and Classification of Acoustic Scenes and Events". This is very important in the fourth industrial revolution: using AI-based models makes it possible to automatically detect mechanical failures in a factory.
We experimented different machine learning models, first non-deep models like GMM and One-Class SVM, then a deep-model (AutoEncoder).

1 Introduction

The purpose of this project is to create a model for anomalies detection in an industry's machines. Each machine is assigned an ID to distinguish individuals of the same machine type.
Moreover, because of the difficulty of collecting patterns of anomalous sounds, the training dataset consists of normal samples only, whereas the test data contains also anomalies.

2 Data analysis

2.1 Dataset description

We are provided with a dataset of 10 seconds audio recordings of these machines working correctly. We know the ID of the machine which is recorded in each audio and nothing more. We are also provided with an other smaller and labeled dataset of 10s audio recordings of machines working both correctly and with an anomaly (again we know the ID).
The dataset consists of audio samples for machines with 3 distinct IDs. However, we need our model to be trainable also with other machine IDs: meaning that we must use the same strategy and hyperparameters to train the model for each machine.
We use the first dataset as training dataset for experiments with non-deep models, while we do an 80-20 split to train neural nets. The second dataset is split in 80-20 in both cases and acts as a validation (model selection & tuning) + test dataset (assess generalization capabilites).

2.2 Fourier Transform

As we are dealing with audio signals, we decided to analyze the Fourier transform of these signals and check if we could discriminate the anomalies based on some patterns in the frequencies domain.

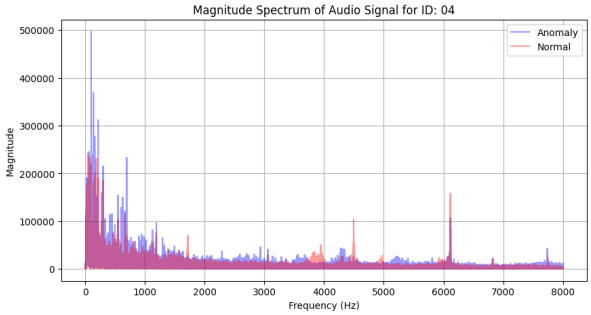

Figure 1: Sample spectrum

From Figure 1 we can see that the spectrum is a bit different in case of anomalies. The main differences consist in some peaks and in the overall magnitude to be higher. Taking into account the mean magnitude of the spectrum and the number of peaks may help with discriminating anomalies.

2.3 Spectrograms

In order to better exploit the properties of the audio signals, we work in the frequency domain using the spectrograms, representing the spectrum of frequencies with the varying of time. In Figure 2 we can observe the difference in the spectrograms of two sound signals from

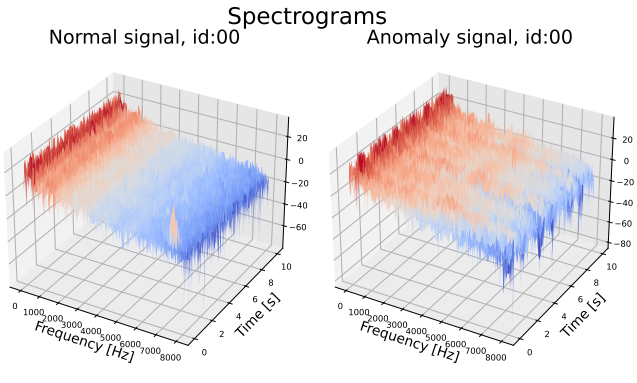

Figure 2: Spectrograms 3D

machines with the same ID: on the left we have a normal signal and on the right we have an anomaly. We can easily note that the spectrogram of the normal signal is more uniform and has less peaks than the corresponding one having the anomaly (this is coherent with our Fourier Transform analysis).

3 Data processing

To use non-deep models which will be discussed in Section 4, we need to perform some data preprocessing to extract relevant features. We process the data separately for each machine ID as follows.

- Downsample the audios by a factor 10
- Calculate the average of the audio signals X_{mean}
- For each sample extract the following 7 features:
  1. Correlation with X_{mean} in the time domain
  2. Euclidean distance of FFT from the FFT of X_{mean}
  3. Euclidean distance of the MFCC components from the MFCC components of X_{mean}
  4. Real part of the difference between the average FFT and the average X_{mean}'s FFT
  5. Imaginary part of the difference between the average FFT and the average X_{mean}'s FFT
  6. Euclidean distance of the pitches from the X_{mean}'s pitches
  7. Number of peaks

These features were selected with a trial and error approach, by plotting how the data was being separated. Figure 3 shows a projection of the data over two of these features, for different machine IDs.
Figure 4 shows the covariance matrix of the transformed

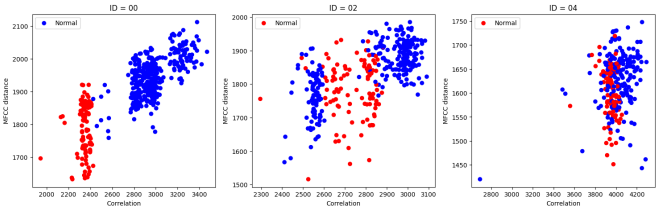

Figure 3: Correlation vs MFCC distance

dataset. Notice that some of the extracted features are correlated, meaning that we could even employ dimensionality reduction techniques to reduce the dataset size.

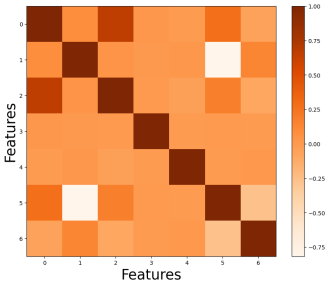

Figure 4: Training Dataset Covariance Matrix

For what concerns the preprocessing of the signals for the AutoEncoder model, we transform all the data into images, converting them into Melspectrograms, i.e. spectrograms that employ the mel-scale: this scale was proposed in 1937 such that equal distances in pitch sounded equally distant to the listener [3].

In both cases (non-deep models and AutoEncoder), the data has then been standardized and normalized.

4 Model selection

In this section we analyze different models that we tested for our purpose and we select the best one. The proposed models are a Gaussian Mixture Model, a One-Class SVM and an Autoencoder.
For evaluating the models we use the *Area Under the (ROC) Curve* score, which gives a metric for how well the model can separate the classes (normal vs. anomaly) regardless of the chosen threshold. The threshold is application-dependent and can be picked in order to favor false positive or false negative predictions.

4.1 Gaussian Mixture Model

Description: We fit a GMM for each machine type ID after pre-processing the data and applying PCA. The dimensionality of PCA is a hyperparameter to tune, while

an other hyperparameter is the number of GMM components. We do not want this last one to depend on the machine type ID, otherwise we would not be able to generalize to the unseen machine IDs which are present in the evaluation dataset. For this reason we fix the number of GMM components to be the same for each GMM (i.e. for each ID).
A grid-search was used to assess the optimal hyperparameters. Then we tried to reduce the number of estimated parameters by also testing the *tied* and *diag* covariance matrix for the GMM components.

Data Preparation: Data has been processed as explained in Section 3 to extract relevant features. We also used PCA to reduce the dimensionality of the dataset, as explained before.

Results: Table 1 reports the AUC score for different hyperparameters choices. We noticed that by changing the seed the results changed by $\pm 2 \cdot 10^{-3}$, so we can assume these results are reliable.

PCA	**GMM components**				
	1	**2**	**4**	**8**	**16**
2	0.942	0.949	0.955	**0.957**	0.957
3	0.933	0.942	0.947	0.949	0.949
4	0.922	0.936	0.939	0.936	0.938
5	0.910	0.928	0.928	0.935	0.933

Table 1: Grid-search results

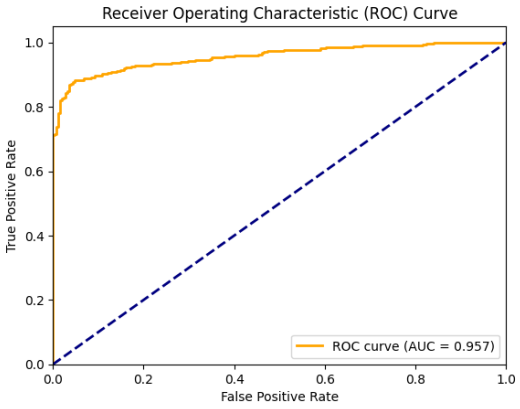

Figure 5: PCA(2), full-GMM-8

PCA(2) is the best regardless of the number of GMM components, while we choose 8 as the optimal number of GMM components even though we get the same AUC with 16 components to avoid overfitting normal outliars (this conclusion has been supported by plotting the means of the GMM).
Since the covariance between the two features obtained after PCA is in the order of 10^{-15}, we can even try to use a *diag*-covariance GMM instead of the *full*-covariance one. This gives us an AUC of 0.954 with PCA(2) and 8 GMM components, which is slightly worse.
With **diag-covariance**, the GMM with 16 components gets a higher score than the model with 8 components, but we prefer to stick to the 8 components GMM for the previously explained reason.

Generalizing the model: Instead of fitting one GMM for each machine type ID, we also tried to fit one GMM for all the data (however, the preprocessing was still done separately for each machine ID since some features differed a lot between distinct IDs). Once more, the best results are obtained by fitting a **full-GMM** with **8 components** and **PCA(2)**, with an AUC score of 0.959.

4.2 One-Class SVM

Description: With the same process explained in Section 4.1 we trained a One-Class SVM (RBF). Since the procedure is the same, we just report the obtained results.

Results: Table 2 shows the optimal hyperparameters configuration in case we want to fit a One-Class SVM for each machine ID. The results are obtained with PCA(2). Results for other values of PCA dimensionality are not reported since they are always worse.

ν	γ		
	10^{-3}	10^{-2}	10^{-1}
0.1	0.944	0.947	**0.964**
0.3	0.940	0.941	0.948
0.5	0.936	0.936	0.936

Table 2: Grid-search results

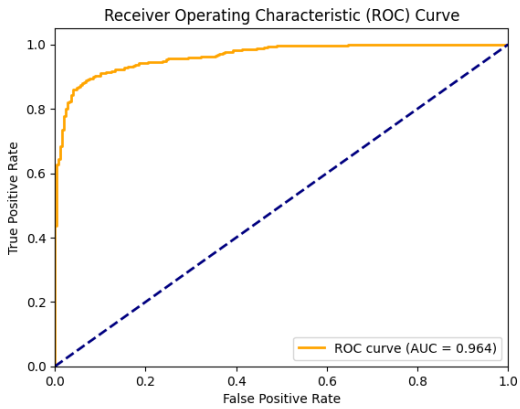

Figure 6: PCA(2), RBF-SVM $\nu = \gamma = 0.1$

By setting $\nu = 0.1$ and $\gamma = 0.1$ we got an AUC score of 0.964, which is even higher than the one obtained with the GMM. Figure 6 shows the ROC curve for such model.

By fitting a One-Class SVM for all machine IDs the results are slightly worse, with an AUC of 0.953 with the same hyperparameters configuration, which is again the best one.

4.3 AutoEncoder

Description: The autoencoder architecture has been proposed to perform anomaly detection tasks in [1] and it has since become very popular. As pictured in Figure 7, with the encoder, we can compress the input normal samples into a latent space, then the decoder has to reconstruct the original image. The model compares the reconstructed image to the original one and classifies anomaly based on the magnitude of the reconstruction loss.

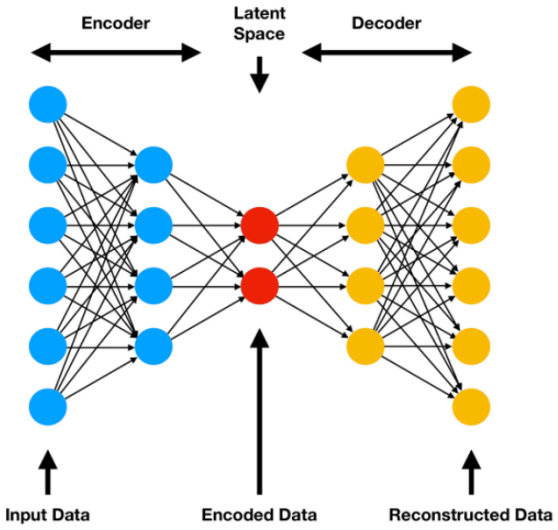

Figure 7: Autoencoder architecture ([2])

We designed an AutoEncoder architecture using 5 Linear layers with ReLU ot Tanh activation maps and a Tanh activation function for the last layer of the decoder. We also experimented with Tanh activations between all the layers, but this change had little impact on the score.

To compute the loss, we use the Mean Squared Error loss, which is suited for reconstruction problems. We set the maximum number of epochs to 200 with early stopping. We split the training dataset into training and validation (80-20 split) in order to supervise the loss and to prevent overfitting. We present the results calculated on the test set, by calculating the AUC score, in Table 3.

Data Preparation: As explained in Section 3, we converted the signals into melspectrograms in log scale and standardized them, subtracting the mean and dividing by the standard deviation. We employed the following parameters for the melspectrogram: $n\_fft = 2048, hop\_length = 512$. We also experimented by increasing the *hop_length* to 1024 but this had no impact on the model.

Results: We trained the model using different learning rates (10^{-3}, 10^{-4}, 10^{-2}, $5 \cdot 10^{-3}$) and compared the resulting losses. The best results were obtained with 10^{-3}. We performed a small search in order to find the best bottleneck size. We tried three different dimensions for the output of the last linear layer, yielding different latent spaces sizes.
Moreover, as said before, we tried different autoencoder architectures, using ReLU and Tanh activation functions between the linear layers of the encoder and of the decoder. We show the results in the following table:

Activation	**Value of z**		
	20	**128**	**256**
ReLU	0.7902	0.7932	0.7977
TanH	0.7973	**0.7982**	0.7967

Table 3: Autoencoder with different bottleneck sizes and activation functions

From the results we can state that the best architecture corresponds to the one having z = 128 and Tanh activation function, reaching AUC = 0.7982. However, all the results are very similar to each other, changing slightly between the different architectures.
We reported also the behaviour of the training and validation loss for the best case in Figure 8: as we can see, the Train and Val losses are decreasing and very close to each other, therefore we can state that the model is not overfitting the training data.

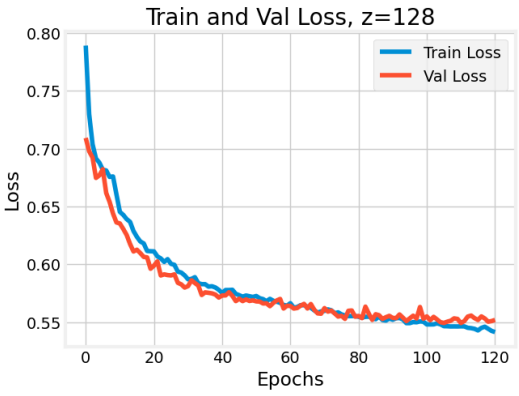

Figure 8: Train and val loss, z=128

In Figure 9 on the left, we can observe the original melspectrograms of two samples (anomaly on the top and normal on the bottom), on the right side we have the images reconstructed by the decoder. We can observe that the reconstruction differs quite a bit from the original in both cases, but the reconstruction of the anomaly sam-

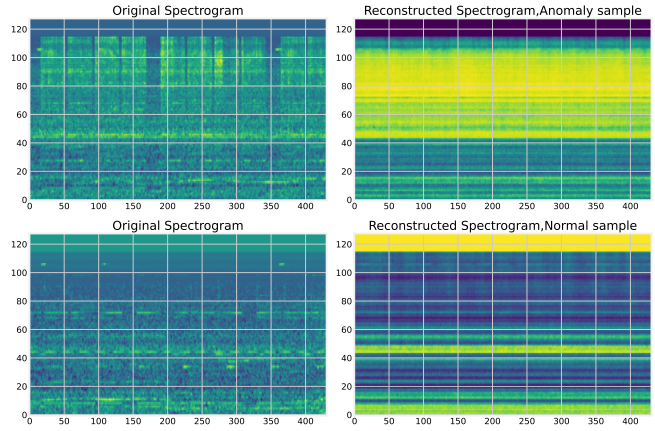

Figure 9: Original and reconstructed mel-spectrogram of a normal sample and an anomaly

ple is evidently less accurate: there are no vertical blue bands, as instead in the original spectrogram.

4.4 Model choice

The non-deep models definitely perform better than the AutoEncoder. Taking this into account, together with the fact that they are much faster to train and lighter to store, we discard the AutoEncoder and focus on the GMM and One-Class SVM with PCA(2).

We think that the One-Class SVM ($\nu = \gamma = 0.1$) with one independent model for each machine ID is the best option to pick in case there are a few machine types to monitor. This solution can also be implemented as a distributed model, which could be useful in case the machines are far away. Given that the models are very light, they can be stored on small embedded calculators near the target machines.

On the other hand, if there is the need to use just one general model because we plan to add more machines in the future, we suggest the full-GMM with 8 components. In this case, data from a new machine should be preprocessed as explained in Section 3 and then fed into the model for the test. This means that before being able to do anomaly detection on the new machine, it will be necessary to collect some audio samples of the machine working in normal conditions. However, this task is easy and fast to do, meaning that this is a very small limitation for our model as long as the number of used machines is small.

5 Test

In this section we report the performances of the chosen models (Section 4.4) on the test set split (20% of labeled data at our disposal).

The One-Class SVM obtains an AUC score of 0.952 on the test set, which is a bit worse than what we found during the validation, but still acceptable.

The GMM obtains an AUC score of 0.967 on the test set, which is even better than what we found at validation time.

6 Conclusions

We dealt with a task of Anomaly Detection on data consisting of sound signals from industrial machines. We encountered the issue of having a training dataset that didn't contain any signals with anomalies, since these can be of varying nature.

We experimented with different techniques involving both non-deep models and neural networks. In the end we picked two candidate models and tested their performance on unseen data to be sure that they generalize.

It turns out that using simple models like a GMM and a One-Class SVM gives better performance with respect to neural networks. This is likely because the size of the dataset at our disposal is small, which makes it difficult to exploit the true power of an AutoEncoder model.

As a farther experiment, we could have tried to increase the size of the dataset with Data Augmentation using a Sliding Window.

References

[1] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018.

[2] GeeksForGeeks. `https://www.geeksforgeeks.org/types-of-autoencoders`.

[3] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190, 1937.