AML 2025 Group 42 - Challenge 2: Anomaly Detection

AHAMAN ULLAH Efaz
EURECOM
Efaz.Ahaman-Ullah@eurecom.fr

CHAUVIER Tom
EURECOM
Tom.Chauvier@eurecom.fr

MEKKI Romain
EURECOM
Romain.Mekki@eurecom.fr

SARR Serigne
EURECOM
Serigne.Sarr@eurecom.fr

Abstract

In this work, we replicate a well-known anomalous sound detection competition: the DCASE Challenge. Constrained to training our models solely on data labeled as 'normal', we adopted unsupervised and self-supervised methods to tackle the challenge. We experimented with Principal Component Analysis (PCA) for its ability to grasp discrete latent variables, Isolation tree for its intuitive behavior in isolating anomalous samples and a Masked Autoencoder based on a vision transformer to try to go further.

1 Introduction

Anomalous Sound Detection (ASD) aims to identify abnormal machine behavior by analyzing sound. It plays a crucial role in modern factory automation, enabling continuous monitoring and fast detection of failures.

In this challenge, the goal is to detect anomalous sounds using only samples of normal sounds as training data. This setup reflects real-world constraints, as anomalous events are rare and highly diverse, making it nearly impossible to build a comprehensive labeled dataset of anomalies.

We focus on one of the six machine types from the original challenge: the slide rail from the MIMII dataset. We have samples from 3 different machines of this type.

Each sample is a 10-second audio clip that includes realistic environmental noise.

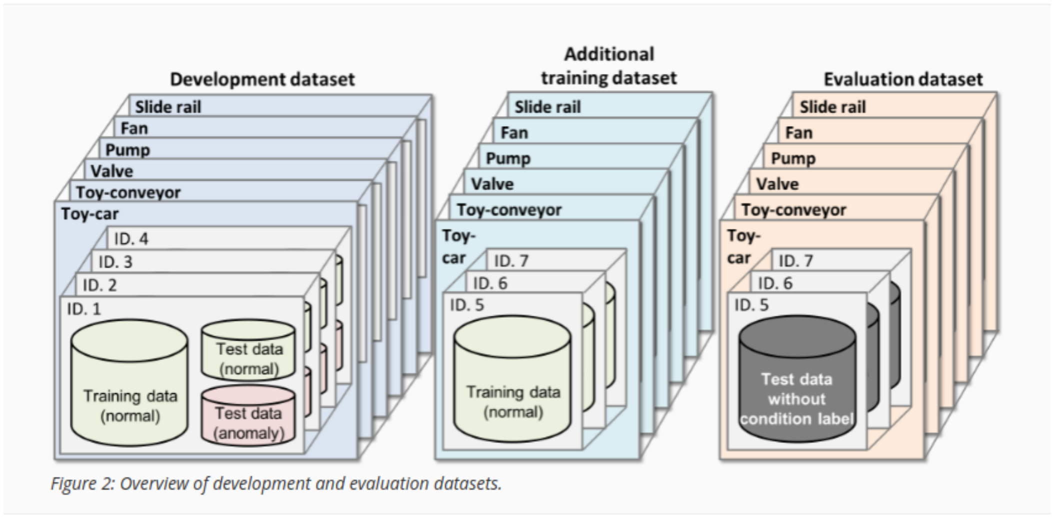

Figure 1: Overview of the development and evaluation dataset

2 Data Preparation

2.1 Dataset construction

2.1.1 Splitting

For processing our audio data, we used the train/test splits already available from the development dataset (later called "Dev set"). The training set contains 2370 normal audio samples while the test set contains a mixture of normal and anomalous samples from machines ’00’, ’02’ and ’04’, for a total of 1101 samples.

2.1.2 Features

The features extracted are described in the appropriate part for each trials we did.

2.2 Preprocessing

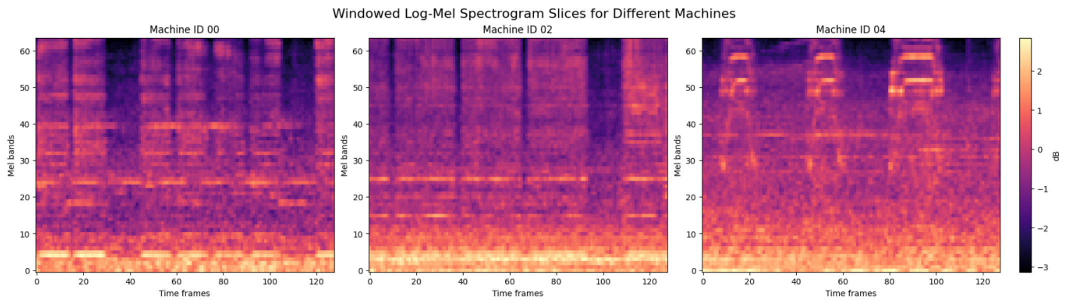

Figure 2: log-mel Spectrogram comparison per machine

We chose to rely on mel spectrograms for two of our solutions (namely the MAE and the PCA) concerning the preprocessing of the audio data. For each audio, we computed its log-mel spectrogram using 64 mel bands, a sampling rate of 16kHz and a maximum frequency of 8kHz. Furthermore, we normalized the log-mel spectrograms to zero mean and unit variance, and applied a sliding window of size 128 and a hop size (overlapping between windows) of 64. For the isolation forest, we used the audio files as they were, extracting meaningful features for each sample as described later.

2.3 Data Augmentation

For the log mel spectrograms, as we chose to use a sliding window over the log mel spectrograms, we had more resulting samples than if using solely the spectrogram. Overall the audios had about 3 slices - which are windows corresponding to images of width 128 and height 64 - per spectrogram.

3 Models and Performances

3.1 Solution 1: PCA

For a first simple approach, we decided to use Principal Component Analysis (PCA). We want to learn a low-dimensional representation of the training data and evaluate a new sample on how well they match the representation.

We expect to be able to represent machine 00 and 02 sounds quite effectively with few components because of the simple shape of their log-mel spectrogram. Machine 04 sounds might be harder to represent as its shape is more complicated and less consistent.

3.1.1 Implementation Details

We used directly the Mel spectrogram of the samples for our model.

for each machine, We chose the number of components by plotting the explained variance by number of components, and choosing the elbow of the curve. This allows us to have enough components to represent most of the feature, without overfitting.

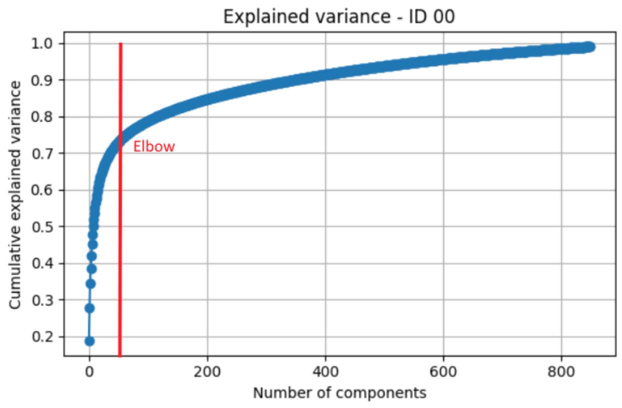

Figure 3: Explained variance by number of components, with elbow value

On the Dev set, we chose the following number of components for each machine:

ID 00: 50
ID 02: 80
ID 04: 130

After fitting the model, we compute the reconstruction error, and apply a sigmoid to obtain a confidence score.

3.1.2 Results and Observations

We achieved interesting results with PCA, with extremely good scores for machine 00 and, while less impressive, also good scores for machine 02 and 04.

We obtained the following AUC-ROC scores with our component number value:

Machine 00: AUC = 0.982
Machine 02: AUC = 0.804
Machine 04: AUC = 0.772

First, we evaluate our number of components choices. To do this, we plotted the AUC-ROC score by number of components for each machine.

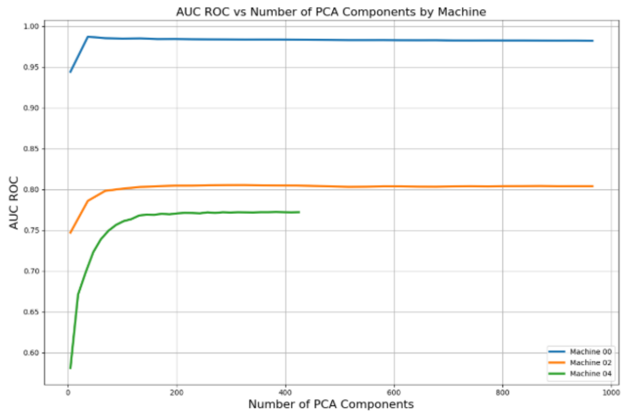

Figure 4: AUC-ROC score vs number of components

We observed that the AUC-ROC value plateaus after our chosen elbow value. This shows that our choice was good, as a bigger number would overfit, and less is not enough to properly represent the data. We can interpret the needed number of components as a "complexity" metric: Because machine 00 requires fewer component to be properly represented than machine 04, its sound pattern are less complex (at least from a PCA perspective). This supports our hypothesis on the machine pattern.

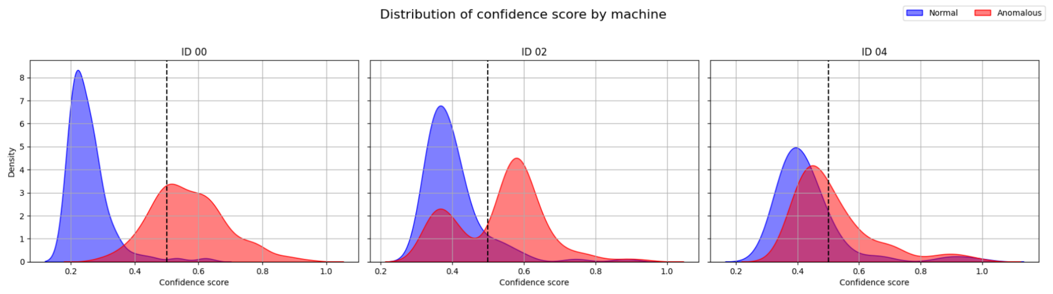

Figure 5: Confidence score distribution with PCA by machine

The distributions teach us the following:

- The machine 00 base sound is very consistent, and its malfunction produces very distinct noise from it.
- The machine 02 has some malfunction which sound are indistinguishable from its normal state, and other than can be pretty accurately separated.
- The machine 04 sound are less consistent for a linear representation, and its malfunction produce sounds very close to its normal state.

3.2 Solution 2: Isolation Forest

For our second approach, we implemented an unsupervised anomaly detection method based on the Isolation Forest algorithm, with the hope that it would better separate anomalous and normal data for machines 02 and 04. Indeed, this model isolates anomalies instead of profiling normal data, making it well-suited for detecting unusual patterns in machine sounds without requiring labeled anomalies during training.

We hypothesize that handcrafted features such as energy entropy, RMS energy, spectral centroid, and tempo, among others, provide a compact and interpretable representation of machine operating sounds. Since these features summarize core spectral and temporal characteristics of the audio, they allow the Isolation Forest to efficiently detect deviations without relying on deep representations.

3.2.1 Implementation Details

To extract meaningful descriptors from audio, we used the following set of features computed on each .wav file:

Feature	Description
Spectral Centroid	"Center of mass" of the spectrum (brightness)
Root Mean Square (RMS) energy	Magnitude of short-term signal amplitude
Energy entropy	Quantifies signal unpredictability
Spectral bandwidth	Fundamental frequency
Spectral flatness	Measures noise-likeness of the spectrum
Onset Strength	Detection of percussive events or note beginnings
Tempo / BPM	Estimated tempo (beats per minute)

Table 1: Table of features

To obtain those final components of the feature vector, we followed our intuition after listening to the audio tracks. Initially, the MFCCs were considered, but excluding them produced better results.

These 7-dimensional feature vectors were computed using librosa, and then used to train an **Isolation Forest** on the *normal* training data of the slider machine.

We set the model parameters to:

- contamination='auto' – automatically estimate the expected proportion of anomalies
- random_state=42 – to ensure reproducibility

The model was trained on the extracted feature vectors from the normal training samples and then evaluated on both normal and anomalous test samples.

3.2.2 Results and Observations

The Isolation Forest performed well in separating anomalous from normal samples on the test set. While the handcrafted feature space is low-dimensional (only 7 features), it encapsulates key statistical and perceptual aspects of the machine's sound behavior.

Using handcrafted features with Isolation Forest yielded the following AUC-ROC scores on the dev set:

- **Machine ID 00**: AUC = **0.90**
- **Machine ID 02**: AUC = **0.72**
- **Machine ID 04**: AUC = **0.69**

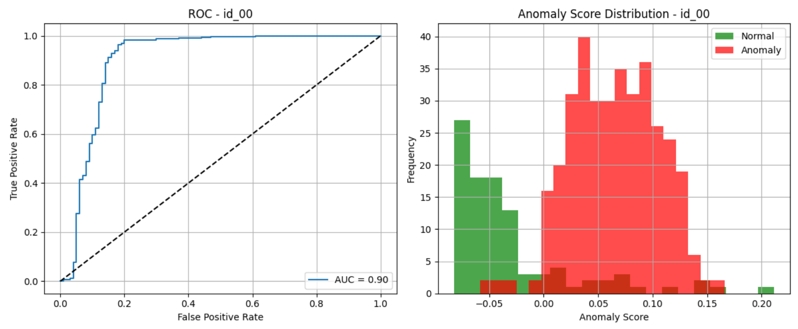

Figure 6: ID 00

The model performs best on machine 00, which has consistent acoustic patterns and clear anomalies.

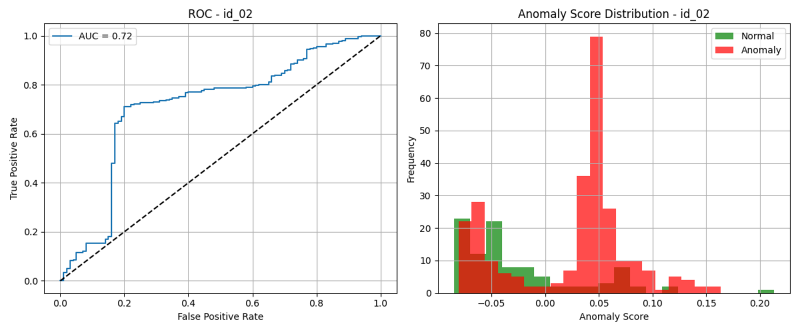

Figure 7: ID 02

Machine 02 shows moderate performance, with some anomalies resembling normal behavior.

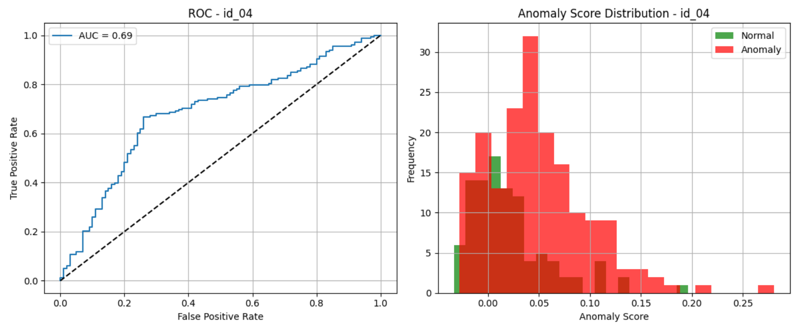

Figure 8: ID 04

Machine 04 is the most challenging, likely due to its complex and less distinctive sound profile.

Score distributions confirm this trend: machine 00 shows strong separation, while 02 and especially 04 have more overlap between normal and anomalous samples.

These results highlight that while Isolation Forest can detect clear deviations using simple features, more expressive models may be needed for acoustically complex machines. It could be that important features for machines 02 and 04 are not taken into account properly.

3.3 Solution 3: Masked Autoencoder (MAE)

Our third solution involves employing a masked autoencoder built on a vision transformer backbone as in the MAE paper [1]. The intuition behind is that as we are constrained to train only on normal data, we want our model to learn how to reconstruct a normal image masked with a specific **mask_ratio** for the anomalous samples to naturally have a higher reconstruction error than the normal ones, as the model would only reconstruct normal samples (the only ones it has been trained with) with fidelity.

3.3.1 Architecture details

Using the log-mel spectrogram slices as inputs (images of size **img_size** $= (64, 128)$, our model starts by "splitting" them into fixed size patches using a convolutional layer (kernel size and stride equals **patch_size**). Each patch is then projected into an embedding space of **embed_dim** dimensions and augmented with learnable positional encodings. Furthermore, a proportion of **mask_ratio** patches are masked out, and the remaining are processed through a vision transformer encoder of depth **d** having **n_head** attention heads, to learn contextual relationships between visible patches. A simple decoder then makes use of the positional embeddings and the contextual processing from the encoder to generate predictions for each missing patch. The training loss (Mean Squared Error) is computed only on the masked patches, for the model to learn more robust representations.

3.3.2 Training difficulty

Due to the high computational cost of transformers, we were unable to experiment with a deep transformer encoder while having a large embedding dimension, many attention heads and small patches all in one without exceeding our computational capabilities. We thus sacrificed the embedding dimension to have more depth, while keeping a relatively large patch size. Furthermore, observing the loss reaching a plateau of approximately 0.9 when using a mask ratio of 0.75, we decreased the mask ratio down to 0.5 for the reconstruction task to be easier. With that, the model's loss still plateau at 0.5 in 20-30 epochs. We used the AdamW optimizer with various learning rates and weight decay, until finding that 1×10^{-5} and 3×10^{-2} were some fitting choices.

Thus the retained parameters of our model were the following :

- img_size=(64, 128)
- patch_size=24
- embed_dim=512
- depth=16
- num_heads=8
- mask_ratio=0.5

3.3.3 Results and model performances

The model performs rather poorly compared to our two first trials, but it can be explained. First, we assume that that training a transformer encoder from scratch requires a lot of data for it to correctly grasp the contextual relation in our patches, given the number of Q K V learnables parameters for 8 heads and 16 layers, plus the positional embeddings to learn. A more seducing idea would have been to start from a pretrained model, however we were restrained by the assumption that images of spectrograms were unusual for vision transformers, thus wanting to train from scratch. We believe that the model could have had better performances even trained from scratch if we were to increase the embedding dimension along with the depth and the masking ratio (ultimately training the model for a longer time), and maybe successively fine tune our base model to explore the loss landscape little by little (by reducing the learning rate). Below are the obtained AUC scores :

[1]Masked Autoencoders Are Scalable Vision Learners, He and al., FAIR (Facebook AI Research)

Machine 00: AUC = 0.72
Machine 02: AUC = 0.63
Machine 04: AUC = 0.50

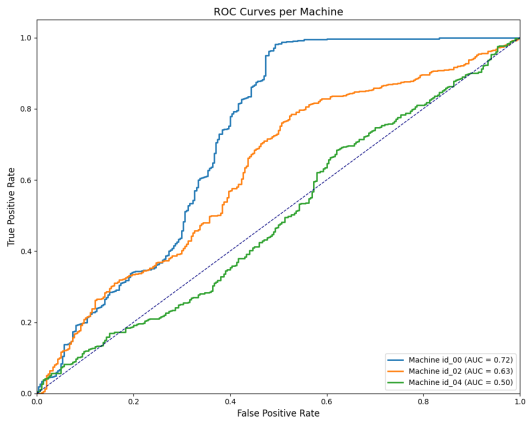

Figure 9: ROC Curves per machines

We can see that the model is no better than random guessing to distinguish samples from machine 4. Looking more precisely on the reconstruction error density for machine 4 samples we observe that the anomalous and normal densities are very similar.

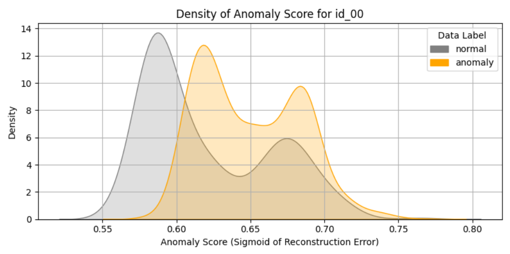

Figure 10: Reconstruction error density for machine 00

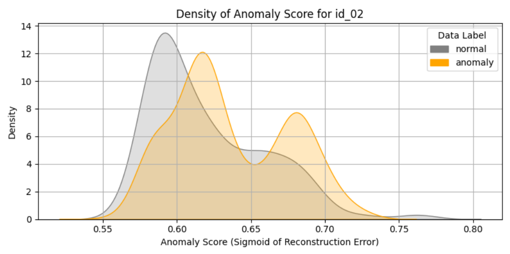

Figure 11: Reconstruction error density for machine 01

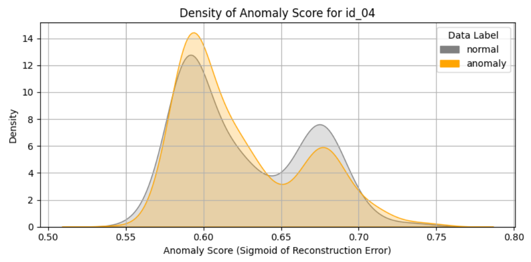

Figure 12: Reconstruction error density for machine 04

The former plot can suggest that our log-mel spectrograms are very similar between the normal and the anomalous data for machine 4. To correct that, can try to reduce the patch size conjointly with increasing our window size for the transformer encoder to grasp more meaningful relations between the patches of a normal sample, thus leading to a stronger representation. Unfortunately we ran out of time and resources to try this for now.

4 Conclusion

In conclusion, our first and simple unsupervised learning PCA solution achieves very good performance for machine 00 and 02, and is clearly better than random for machine 04. Isolation forest is still a valid solution while being even simpler given the small amount of features it uses. Our last solution (MAE-ViT) could have been promising but we lacked the time and resources to train it properly. Further efforts can be put on machine 4 spectrograms and in the MAE hyperparameter tuning to learn a better representation of the normal samples.