Anomaly Detection - Challenge 2 AML

Alex Argese
Student
EURECOM
Biot, France
alex.argese@eurecom.fr

Cristian Degni
Student
EURECOM
Biot, France
cristian.degni@eurecom.fr

Miriam Lamari
Student
EURECOM
Biot, France
miriam.lamari@eurecom.fr

Enrico Sbuttoni
Student
EURECOM
Biot, France
enrico.sbuttoni@eurecom.fr

***Abstract*—This report presents a comparative analysis of unsupervised machine learning techniques for detecting anomalous sounds in industrial equipment, specifically focusing on the slider machine type. The primary challenge addressed is identifying unknown anomalous sounds in test samples, given only normal sound recordings during training. We describe the dataset, which includes audio recordings from the ToyADMOS and MIMII collections, and detail the preprocessing steps such as audio normalization, feature extraction using log-mel spectrograms. Various models were evaluated, including standard Autoencoders, Variational Autoencoders (VAEs), Convolutional Autoencoders (CAEs), and the PANNs (Pretrained Audio Neural Networks) inference framework. The models were assessed using Area Under the Curve (AUC) as the primary metric to reflect their ability to rank samples by anomaly likelihood. Results highlight the strengths and limitations of reconstruction-based methods versus embedding-based representations, with PANNs showing promising generalization across machine IDs. This work underscores the potential of deep unsupervised methods for robust anomaly detection in real-world, noise-contaminated industrial environments.**

***Index Terms*—Machine Learning, Anomaly Detection, Sound processing, Autoencoder, Pannes Inference**

I. Introduction

Detection of **anomalies** in **industrial equipment** is a critical task for ensuring **operational efficiency** and preventing costly **downtime**. In this challenge, we focus on detecting **anomalous sounds** from a specific type of machine, the **slider**, using **unsupervised machine learning** techniques. The primary goal is to identify **unknown anomalous sounds** in test samples, given only **normal sound recordings** during training. The dataset used for this task consists of audio recordings from the **MIMII** dataset.

II. Dataset

The dataset consists of **audio recordings** in **.wav** format, collected with **eight microphones** placed around a **slider machine**. All recordings are treated as a **single-channel** input from a **fixed microphone**. Each recording is approximately **10 seconds** long and includes both the **operating sound** of the slider machine and **environmental noise**. All signals are **downsampled** to a **sampling rate** of **16 kHz**.

The **training set** contains only **normal sound recordings**, while the **test set** includes both **normal** and **anomalous sounds**.

The dataset is organized as follows:

- **dev/ folder**: contains around **1,000 normal samples** for training and **100–200** samples each of **normal** and **anomalous sounds** for testing.
- **eval/ folder**: includes approximately **400 unlabeled test samples** for evaluation, along with additional **normal samples** that may be used for training.

The **class distribution** within the test set is **unbalanced**:

- **Class Anomalous**: 801 sounds
- **Class Normal**: 300 sounds

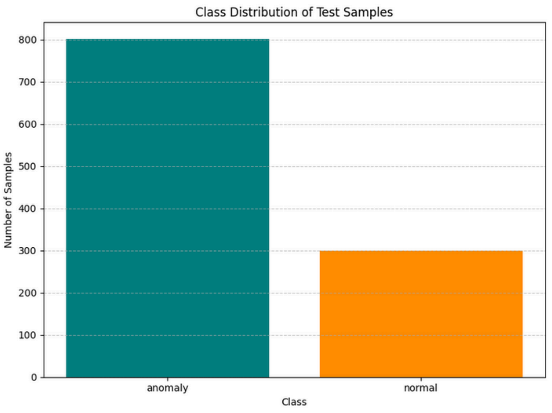


Fig. 1: Test samples class distribution.

III. Preprocessing

A. Mel-spectrogram transformation

Conversion of **waveform** signals into a **mel-spectrogram** representation is a common practice in **audio analysis**, especially suitable for **machine listening** tasks such as **anomaly detection**.

The preprocessing pipeline involves first **loading** the **.wav** audio files using the **Librosa** Python library, followed by

mel-spectrogram extraction using the following parameters:

- **number of FFT components**: defines the number of samples used in each Fourier Transform window;
- **hop length**: determines the step between successive analysis frames;
- **number of mel bands**: sets the resolution of the mel frequency axis.

The resulting **mel-spectrograms** are converted to the **decibel scale** and then **reshaped into flattened vectors** for efficient storage and compatibility with model input formats. The extracted features are **organized into separate folders** for training and testing purposes.

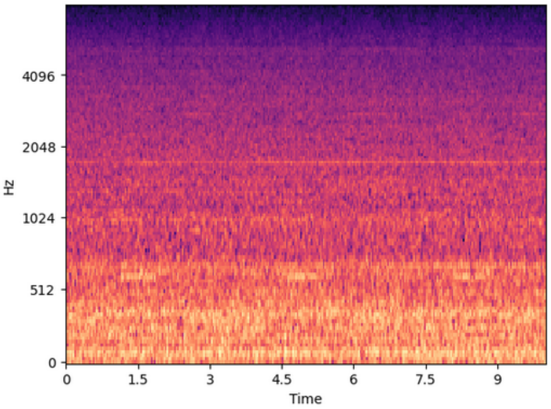


Fig. 2: Example of a mel-spectrogram extracted from a slider machine audio recording.

B. *Feature Normalization*

After converting the **audio signals** into **mel-spectrogram** representations and flattening them into **1D feature vectors**, we applied **feature normalization** to ensure **numerical stability** and improve **model performance**.

We used **z-score standardization**, which transforms each feature to have a **mean of 0** and a **standard deviation of 1**. This is particularly important for models that are **sensitive to feature magnitudes**, such as **support vector machines** and **neural networks**. Without normalization, features with **larger numerical ranges** could **dominate the learning process** or **slow convergence**.

The **normalization parameters** (mean and standard deviation) were computed using the entire set of **normal training samples**. This is consistent with the **anomaly detection** framework, where it is assumed that only **normal examples** are available during training.

IV. Models Evaluated

A. *Fully-Connected Autoencoder*

A **fully-connected Autoencoder** was used to learn **compressed representations** from **tabular input features**. Implemented in **PyTorch**, the model consists of a **dense encoder** that maps input vectors to a **latent space** of dimension **64**, followed by a **symmetric decoder** that reconstructs the original input.

The model is trained using **mean squared error (MSE)** as the reconstruction loss and optimized with the **Adam optimizer** over **30 epochs**. A **learning rate scheduler** is applied to adjust the learning rate dynamically and help prevent overfitting.

TABLE I: Hyperparameter setup for Fully-Connected Autoencoder

Hyperparameter	Ranges / Values	Used
latent_dim	64 (AE), 8 (VAE)	64 / 8
optimizer	Adam	Adam
lr	1e-3	1e-3
β	0.001	0.001

B. *Convolutional Variational Autoencoder*

A **Convolutional Variational Autoencoder (ConvVAE)** was used to model the distribution of **normal Mel spectrograms** extracted from **16kHz audio recordings**. The architecture consists of a **convolutional encoder** that compresses the input into a **latent space** of dimension **8**, **16**, or **32**, and a **decoder** that reconstructs the input from this latent representation. Spectrograms are **standardized** and **zero-padded** to ensure compatibility with the convolutional layers.

The model uses the **reparameterization trick** to allow backpropagation through stochastic sampling, and is trained with a **β-VAE loss**, which combines **MSE reconstruction loss** and **KL divergence**. A grid search over **latent dimensions** and **β values** was performed using **5-fold cross-validation** on normal samples to select the best configuration.

TABLE II: Hyperparameter setup for ConvVAE

Hyperparameter	Ranges / Values	Used
latent_dim	8, 16, 32	32
β	0.01, 0.1, 1.0	0.01

C. *Variational Autoencoder*

A **Fully-Connected Variational Autoencoder (VAE)** was used to learn **compact latent representations** from tabular data. The model is implemented in **PyTorch** and features a **fully-connected encoder** that maps inputs to a **latent space** of dimension **8**, using separate layers to estimate the **mean** and **log-variance**.

A **decoder** reconstructs the input from a latent variable obtained through the **reparameterization trick**. The loss function combines **mean squared error (MSE)** for reconstruction and **KL divergence** for regularization, forming a **β-VAE loss**. The model is trained with the **Adam optimizer**, and a **learning rate scheduler** is used to reduce learning rate when performance plateaus.

TABLE III: Hyperparameter setup for VAE

Hyperparameter	Ranges / Values	Used
latent_dim	8, 16	8
optimizer	Adam	Adam
lr	1e-3	1e-3
β	0.001	0.001

D. PANNs Pretrained Model

The model **Cnn14** is a **pretrained convolutional neural network** from the **PANNs** (Pretrained Audio Neural Networks) family. It was trained on the **AudioSet** dataset and is designed for **general-purpose audio tagging**. The model takes **raw waveform input** and produces a fixed-size **2048-dimensional embedding** that captures **high-level acoustic features**.

Implemented via the **panns_inference** library, the model is used in **inference mode only**, with no additional training or fine-tuning. Audio inputs are **converted to mono**, **resampled to 32kHz**, and passed to the model through the provided **AudioTagging** interface. The resulting embeddings can be used for downstream tasks without modifying the network, leveraging its **pretrained audio representations**.

E. VGGish Pretrained Model

VGGish is a **pretrained convolutional neural network** developed by **Google** and trained on a **large-scale YouTube audio dataset** for **audio classification tasks**. It transforms **raw audio waveforms** into **128-dimensional embeddings** that capture **semantic-level acoustic information**. The model operates on **log-mel spectrograms** computed from audio resampled to **16kHz**, and includes a **post-processing PCA step** for **dimensionality reduction** and **whitening**.

In our pipeline, **VGGish** is used strictly in **inference mode**, without additional **training** or **fine-tuning**. **Audio samples** are **preprocessed** to fit the model's expected input format and passed through the network to obtain **fixed-size embeddings**. These embeddings are then used for **anomaly detection**, leveraging the **rich representational capacity** of the **pretrained model** to characterize **normal audio behavior**.

V. Metric Justification

To evaluate model performance in ranking anomalies, we primarily used the **ROC AUC**, which measures the model's ability to distinguish between **normal** and **anomalous** samples, independently of a decision threshold. This is particularly important in **unsupervised anomaly detection**, where decision boundaries are not known a priori.

While additional classification metrics such as **F1-score**, **precision**, and **recall** were reported after threshold selection, they were not used for **model selection** or hyperparameter tuning.

VI. Results Summary

A. Detailed Results – Fully-Connected Encoder

Prior to implementing the **Variational Autoencoder**, we trained a simple **fully-connected encoder** on the same **normalized tabular Mel spectrograms**. Unlike the **VAE**, this model did not include a **generative decoder**; instead, it focused purely on **compressing input representations**. **Anomaly scores** were computed using the **Mean Squared Error (MSE)** between the **encoded representations** of test samples and the **mean embedding vector** of the training data. Despite its simplicity, this approach achieved a **ROC AUC of 0.7109**, highlighting that even a **basic encoder** can yield **meaningful representations** for **anomaly detection** when combined with a **suitable scoring function**.

B. Detailed Results – Fully-Connected VAE

The **fully-connected Variational Autoencoder** (**VAE**) was trained on **normalized tabular representations** of the Mel spectrograms. Despite effective training convergence (with total loss decreasing from over **10,000** to below **1,000**) the final **ROC AUC** was only **0.3975**. This suggests the learned embeddings were **not sufficiently informative** for distinguishing anomalous samples. The poor performance may stem from the VAE's **limited capacity to model temporal and spatial structure** in spectrogram data.

C. Detailed Results – Convolutional VAE

The **Convolutional VAE** (**ConvVAE**) was trained directly on **2D Mel spectrograms**, using convolutional layers to better capture **local spatial structure**. A grid search over **latent dimensions** and **β values** identified the optimal setting as latent_dim=32, β=0.01, yielding a **cross-validated loss** of **33689.0**. Final evaluation on the test set produced a **ROC AUC** of **0.7890**, with an **F1-score** of **0.8575**, **accuracy** of **0.7947**, and **precision** of **0.8662**. These results confirm the model's ability to **accurately reconstruct normal patterns** while effectively distinguishing **anomalous events**.

D. Detailed Results – PANNs Embedding + Mahalanobis Distance

The **PANNs-based model** used a **pretrained Cnn14 architecture** to extract **2048-dimensional embeddings** from raw waveforms. An anomaly score was computed using the **Mahalanobis distance** from the mean of normal embeddings. This method achieved the best overall result, with a **ROC AUC** of **0.9311**, significantly outperforming the reconstruction-based approaches. The strong performance confirms the effectiveness of **pretrained audio representations** for capturing **semantic structure** in acoustic scenes.

E. Detailed Results – VGGish Embedding + Mahalanobis Distance

The **VGGish model** was used to generate **128-dimensional embeddings** from the audio inputs. **Anomaly detection** was performed by computing the **Mahalanobis distance** between each test embedding and the distribution of embeddings obtained from **normal training data**. This method achieved a **ROC AUC of 0.8513**, confirming the effectiveness of **transfer learning** and **pretrained audio representations** for detecting **anomalous acoustic events**. While not as performant as **PANNs**, the results show that **VGGish embeddings** still capture **relevant structure** in the data, providing a strong balance between **efficiency** and **accuracy**.

F. Metric Selection Rationale

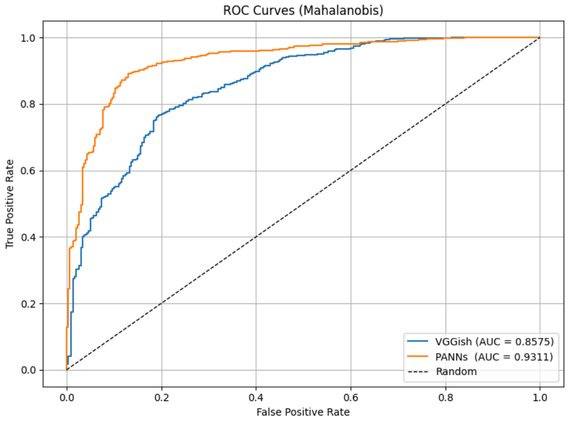
For both **VGGish** and **PANNs-based models**, we experimented with different **anomaly scoring methods**, including **Euclidean** and **cosine distances**. However, we observed that using the **Mahalanobis distance** consistently yielded significantly better results—improving **ROC AUC scores** by approximately **10 percentage points**. This improvement is likely due to **Mahalanobis distance** accounting for the **covariance structure** of the **embedding space**, making it more sensitive to **deviations from the normal distribution** in **high-dimensional representations**.

TABLE IV: Performance Comparison of Models on the Test Set

Model	ROC AUC
Fully-Connected Encoder	0.7109
Fully-Connected VAE	0.3975
Convolutional VAE	0.7890
PANNs + Mahalanobis	0.9311
VGGish + Mahalanobis	0.8513

VII. Model Selected

Although the **Convolutional VAE** performed well and achieved high **precision** and **F1-score** on the held-out test set, the **PANNs embedding method** achieved a **substantially higher ROC AUC** of **0.9311**. Given the **unsupervised nature** of the task and the emphasis on **ranking anomaly likelihood**, **ROC AUC** was prioritized as the primary evaluation metric.



Therefore, we selected the **PANNs + Mahalanobis** method as the **final model**, due to its superior **generalization**, **semantic awareness**, and **robustness to noise**. This approach is also **computationally efficient**, requiring no retraining and leveraging **powerful pretrained audio features**.

VIII. Inference on Unlabeled Test Set

We applied both the **Conv-VAE** and the **PANNs + Mahalanobis** pipeline to the **unlabeled evaluation set** provided. The goal was to rank test samples by **anomaly score** and identify the most suspicious examples.

A. Conv-VAE

The **Convolutional Variational Autoencoder** computes an **anomaly score** based on the **reconstruction error** between the input spectrogram and its reconstruction. A manually selected threshold (derived from the validation ROC curve) is applied to label samples as normal or anomalous.

While the model can flag anomalous instances, it suffers from:

- **sensitivity to audio distortions**,
- potential overfitting to training noise patterns,
- and **limited generalization** to unseen anomalies.

This makes its inference results less reliable compared to embedding-based methods.

B. PANNs + Mahalanobis

The **PANNs model** provides **pretrained 2048-dimensional embeddings** for each audio file. By modeling the distribution of normal embeddings and computing the **Mahalanobis distance**, we obtain robust anomaly scores without retraining. This method:

- generalizes well across machine conditions,
- is **insensitive to minor signal variations**,
- and produces **well-calibrated anomaly rankings**.

IX. Conclusion and Next Steps

In this challenge, we compared three unsupervised approaches for **anomalous sound detection** on the **slider machine** using audio recordings from the MIMII dataset. Our analysis revealed that while **fully-connected VAEs** were limited by their inability to capture spectro-temporal patterns, **Convolutional VAEs** significantly improved anomaly detection by leveraging local structure in Mel spectrograms.

However, the best results were obtained by the **PANNs-based model** with **Mahalanobis distance**, which achieved a **ROC AUC of 0.9311** without requiring retraining. This underscores the value of **pretrained semantic audio representations** for generalization in real-world noisy environments.

As future work, we propose exploring **semi-supervised training** using pseudo-labeling strategies, experimenting with **attention-based models** applying **temporal models** (e.g., LSTMs or Transformers) to capture longer-term dependencies in machine sounds, and evaluating the use of **domain adaptation techniques** to improve robustness across different machine types or noise conditions.

X. References

This report is inspired by the DCASE challenge and its application to real-world industrial environments, as described in:

DCASE Challenge Task 2 (2020), **Unsupervised Anomalous Sound Detection for Machine Condition Monitoring**. The MIMII Dataset: Koizumi et al., **MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation**, 2019. The ToyADMOS Dataset: Purohit et al., **ToyADMOS: A Dataset of Miniature-Machine Operating Sounds for Anomalous Sound Detection**, 2019.